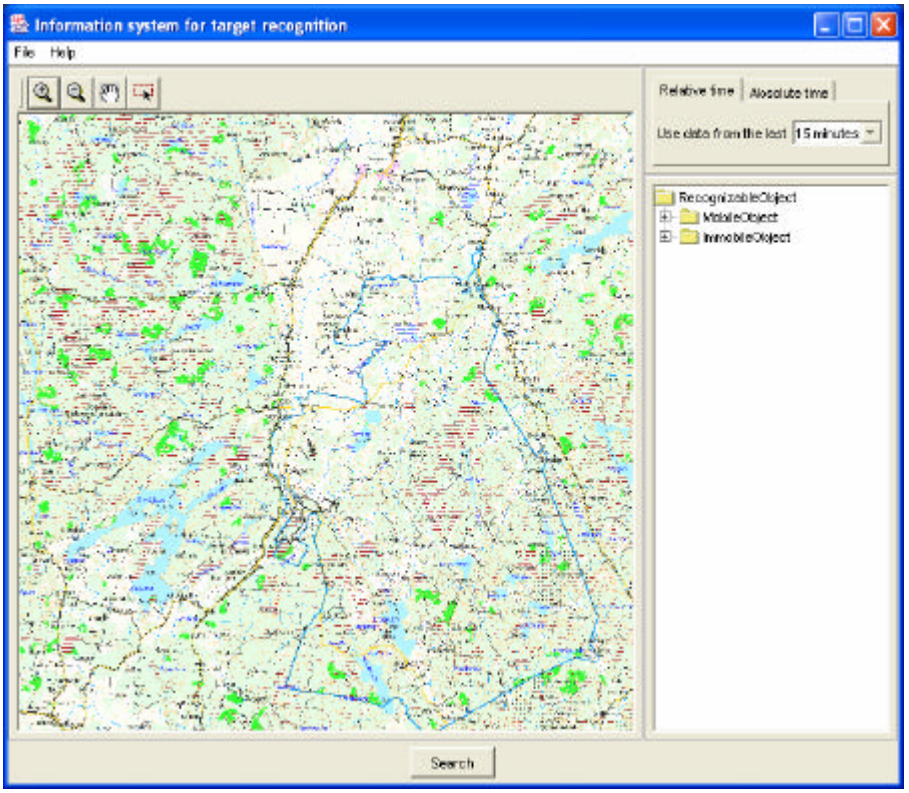
Ett frågebaserat beslutsstödssystem för nätverksbaserade ledningssystem

Erland Jungert, Martin Folkesson,
Jörgen Fransson, Tobias Horney,
Fredrik Lantz, Karin Silvervarg



FOI-R--1787--SE
ISSN 1650-1942

Ledningssystem
Användarrapport

September 2005

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1350 anställda varav ungefär 950 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömningen av olika typer av hot, system för ledning och hantering av kriser, skydd mot hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
Ledningssystem
Box 1165
581 11 Linköping

Tel: 013-37 80 00
Fax: 013-37 81 00

www.foi.se

Ett frågebaserat beslutsstödssystem för nätverksbaserade ledningssystem

FOI-R--1787--SE
ISSN 1650-1942

Ledningssystem
Användarrapport

September 2005

Innehållsförteckning

1. Inledning

Beslutsstödssystem[1] för bland annat militära och krisrelaterade tillämpningar integrerade i nätverksbaserade ledningssystem måste i de flesta fall kunna samla in, analysera, fusionera, hantera, lagra och till sist också kunna visualisera en mängd olika typer av sensordata. I dessa system kan sensorer vara av en mängd olika typer. Ur användarens perspektiv är det inte önskvärt att behöva ha kunskaper om dessa sensorer, deras data eller hur dessa data skall hanteras eftersom de flesta slutanvändare inte besitter den nödvändiga tekniska kompetensen som krävs för detta. Den naturliga konsekvensen av detta blir att dessa beslutsstödssystem, som i sig ofta utgör olika former av informationssystem eller *tjänster*, måste vara oberoende av vilka sensorer som används av systemet och hur sensordata hanteras. Vidare måste dessa tjänster också vara anpassade till de krav som för övrigt ställs på de ledningssystem i vilka de kommer att vara integrerade. Bland dessa krav kan nämnas anpassning till det nätverksbaserade försvaret (NBF), tjänsteanpassade och interoperabla, dvs enkelt utbytbara, map aktuella sensortyper.

Syftet med detta arbete har varit att utveckla ett informationssystem omfattande ett antal beslutsstöd (tjänster) vilket kan integreras i ett nätverksbaserat ledningssystem. Detta informationssystem skall låta användaren ställa frågor i avsikt att samla in information om pågående aktiviteter i ett specificerat geografiskt område av intresse för användaren. Frågor skall kunna ställas riktade mot historiska data i en databas eller mot data som strömmar in från de olika datakällorna under en given tidsperiod där sensorer och andra observatörer övervakar den aktuella terrängen. Datakällorna kommer i första hand att utgöras av sensorer men även andra typer, t ex textmeddelanden från olika observatörer kan tänkas förekomma. Informationssystemet skall således uppfattas som ett system som tillhandahåller ett antal tjänster, vilka kan leverera olika efterfrågade data som återspeglar skeendet i den terräng systemet övervakar.

Den mest omfattande tjänsten i det informationssystem som beskrivs här utgörs av ett frågespråk, som i första hand användas för att ställa frågor om olika förekommande markmål, deras relationer, egenskaper och status (t ex läge). Detta frågespråk är avsett främst för multipla sensordatakällor där data ofta är heterogena. För övrigt omfattar informationsystemet två andra

1. Detta arbete ingår i ett forskningsprojekt vid FOI kallat informationssystem för markspaning (IS-MS) och som finansierats av Försvarsmakten.

tjänster; en för situationsanalys och en för bestäming av bl a framkomligheten i terrängen med hjälp av digitala kartor och med en högupplösande terrängmodell.

Denna rapport innehåller också särtryck av ett antal publikationer som skrivits och publiserats under projektets gång. Dessa särtryck återfinns som ett antal olika appendix vilka refereras till i rapporten för att ge läsaren en möjlighet till fördjupad förståelse av de olika delarna av arbetet i projektet.

2. Problemformulering

Huvudproblemet i detta arbete har varit att utveckla ett informationssystem [1] för markspaning. Informationssystemet skall innefatta ett frågespråk med förmåga att detektera och känna igen markmål med hjälp av data från multipla sensortjänster.

I arbetet har också ingått studier för att utveckla en lämplig nätverksanpassad systemarkitektur baserad på intelligenta agenter för att kunna allokera aktuella sensortjänster i ett nätverk och analysera de data som dessa tjänster levererar. Ansatsen till systemarkitektur är baserad på ett distribuerat ontologiskt kunskapssystem.

En väsentlig problemställning vid utveckling av frågespråket har varit att bestämma hur metodik för fusion av multipla sensordata skall vara utformad. Vidare har funnits krav på metoder för visuell interaktion på olika nivåer med frågespråket. Tjänster för situationsanalys och terrängframkomlighet i högupplösande terrängmodell har också fokuserats. En ytterligare målsättning har varit att demonstrera markspaningsegenskaperna genom att integrera systemet med MOSARTs [2] simuleringsmiljö.

Enligt förutsättningarna skall informationssystemet utnyttjas som ett beslutsstöd i NBF-miljö som stöd för spaning i godtycklig geografisk omgivning med hjälp av ett avancerat operatörsstöd och multipla sensorer. Informationssystemet skall betraktas som en lagringsplats för olika tjänster med avsikt att ge stöd för bl a markspaning, situationsanalys och underrättelseverksamhet. Systemet skall dessutom kunna användas för:

- måldetektering och måligenkänning med fusion av data från multipla sensorer,

- stöd för generering av en aktuell lägesbild.

3. Scenario

Vid FOI (genom medverkan av 5[1] olika projekt) har ett senario utvecklats för att demonstrera förmågor hos olika beslutsstöd. Det aktuella scenariot går under namnet scenario för slaget vid Stora Vredski klint (1455) och syftar på möjliga händelser i internationella uppdrag som i detta scenario utgörs av ett FN-uppdrag. I scenariot beskrivs ett händelseförlopp där man med hjälp av bl a ΣQL och ett antal sensormodeller och sensormanagementteknik samlar in information om det pågående händelseförloppet (som simuleras i MOSART) för att skapa en lägesbild utifrån vilken ledningen för FN-trupperna skall kunna fatta adekvata beslut om insatser mot dels aktivisterna i området samt dels lokalbefolkningens demonstrationer med anledning av 550-årssjubileet av slaget. En kortfattad beskrivning av scenariot och dess förutsättningar utgör följande beskrivning:

Bakgrund
I landet Crisendo finns två etniskt olika befolkningsgrupper A-folk och B-folk. A-folk är i minoritet med ett antal enklaver. I den aktuella enklaven ligger berget Stora Wredskiklint som var platsen för ett slag mellan de båda folken den 12 november 1455 och då B-folk besegrade A-folk i grunden. Sedan dess firas denna seger bland B-folk årligen genom att deltagarna marscherar in mot en punkt söder om klinten. Under senare år har firandet varit mycket begränsat och främst genomförts av en mindre grupp nationalister. Berget Stora Wredskiklint ligger i en av A-folkenklaverna. Denna enklav skyddas av FN-trupper. En halv FN-pluton (OP) finns grupperad i en postering vid en punkt nära inmarschområdet till klinten. Detta år har dock nationalisterna lyckats få gehör bland ett stort antal anhängare. B-folks aktiviteter startar tidigt på morgonen den 12 november 2005 för firandet av 550-årsjubiléet.

Syfte och slutmål
I detta scenario är syftet dels att samla in information som återger läget och presentera detta dels att fatta beslut om hantering av situationen. Målet är således att ge beslutsfattarna ett underlag för att fatta beslut om vilka åtgärder som skall vidtas efterhand för att följa och styra händelseutvecklingen med hänsyn till de aktiviteter som har registrerats. Det senare, dvs att bestämma vilka beslut som skall tas, ligger dock utanför uppdraget.

Miljö
Scenariot kommer att simuleras i simuleringshjälpmedlet MOSART som utvecklats vid FOI. Till hjälp för verksamheten kommer i det att finnas ett antal sensormodeller implementerade i MOSART. Två olika beslutsstödshjälpmedel finns också anslutna; dessa är dels frågespråket ΣQL dels ett system för sensorstyrning.

1. Förutom IS-MS har IAM, SEMARK, MOSART och TMDI projekten medverkat.

Tillgängliga FN-resurser
Förläggningsplats för FN-kompaniet är 2 km söder om Tingsberget.
Vid kompaniet finns en tillgänglig UAV med en IR-sensor och förmåga att lägga ut 6 små marksensornät varje flygning. Totalt finns 10 marksensornät tillgängliga.
Ett sådant nät har batterier för 24 timmars drift.

För närvarande är en halv pluton (Op) grupperad nära klinten och dess inmarschområde strax väster om Ströplahult.
Ytterligare två stationära sensornätverk ligger utplacerade vid (Eveborg) och (Älgmyra).

Vid behov kan också ett flygplan med en CARABAS (som är ett radarsystem utvecklad vid FOI) rekvireras.

Begränsningar
Efter stora vägen norrut har vägen förstörts vid Smedstorp. Området är också minerat så att det är omöjligt att ta sig fram här. Detta gäller både fordon och människor

Utgångspunkten för de frågor som skall ställas med hjälp av frågespråket är de befintliga mark-sensornäten samt tillgången till CARABAS-systemet. Senare i scenariot kommer ytterligare sensorer att bli tillgängliga men med det angivna utgångsläget kan man tidigt börja ställa frågor för att bygga upp den aktuella lägesbilden. Vi har här valt att formulera frågorna som text för att ge en ökad förståelse av frågespråkets förmåga, dvs den visuella tekniken används inte här.

Eftersom man har misstankar om förekomsten av vissa aktiviteter bör den första frågan bli:

Finns det några mobila objekt i området vid den angivna tiden?

Med *området* menas i detta sammanhang det aktuella intresseområdet. Genom att *mobila objekt* efterfrågas innebär detta att man i första hand är intresserad av att *detektera* rörelser.

Som tillägg till denna fråga bör också läggas villkoret:

på väg

Detta tilläggsvillkor är nödvändigt eftersom man vill specificera att det främst är fordon som efterfrågas.

Eftersom man också misstänker att det har upprättats en eller flera vägspärrar i området bör man också specificera en tilläggsfråga:

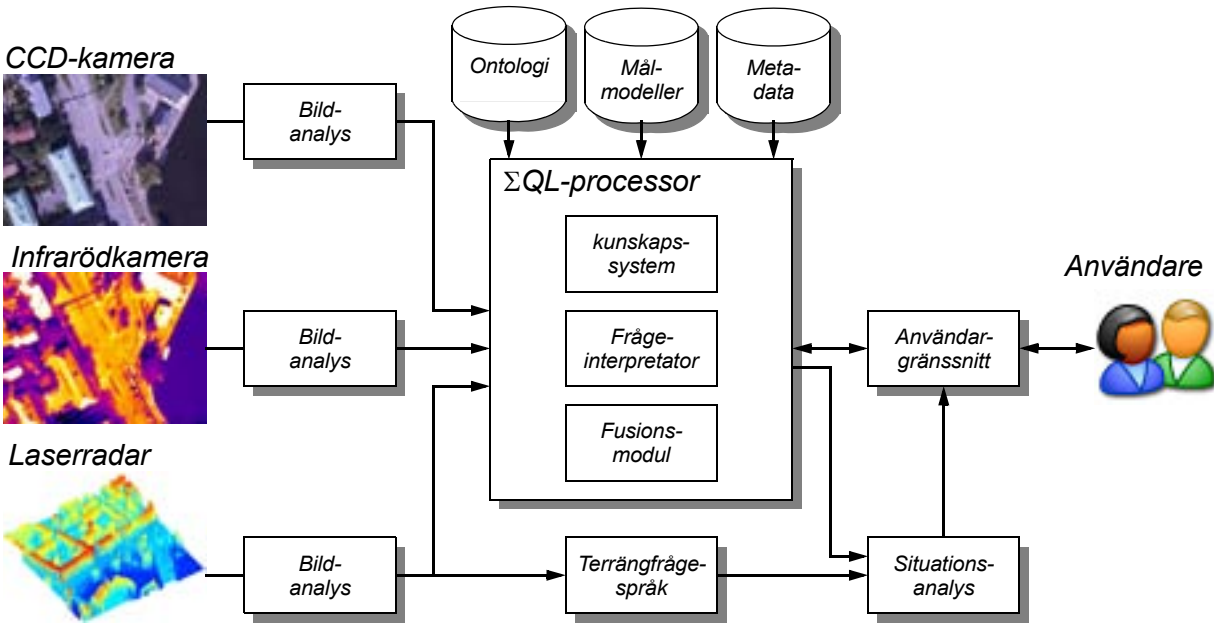
Finns det några konstaterade fordon i området vid den angivna tiden?

Till denna fråga kan fogas en följdfråga som grundar sig på misstanken att vägspärrar finns upprättade i området. Denna följdfråga kan formuleras som:

Förekommer det några vägspärrar i området?

Speciellt i detta sammanhang har vi valt att definiera en vägspärr som ett större fordon, t ex en lastbil, som är placerad tvärs över vägen. Självklart finnas det andra typer av definitioner på vägspärrar.

Med det angivna scenariot kan sedan ytterligare frågor ställas när man har skickat upp den tillgängliga UAV:n. Dessa frågor har i huvudsak samma karaktär som de ovan angivna.



Figur 1. Översikt över informationssystemet.

4. Informationssystemet

En översikt av informationssystemet återfinns i Figur 1 med exempel på några olika möjliga sensorer. De olika modulerna i översikten kommer att diskuteras vidare nedan. Delarna finns också beskrivna ur mera tekniska synvinklar i de efterföljande appendixen. Emellertid kommer inte metoderna för sensordataanalys att diskuteras i denna rapport eftersom detta till största delen ligger utanför det nuvarande projektet. Delar av det arbete som berör sensordataanalysen, finns beskrivna i [1] och i Appendix A och C.

4.1 Frågespråket

Frågespråket ΣQL kan ur ett användarperspektiv karaktäriseras enligt följande:

- Interoperabelt m a p sensortyper.

- Sensordataoberoende.

- NBF anpassat genom agentarkitektur.

- Tjänsteorienterat med förmåga att producera efterfrågad information.

- Inkluderar visuellt användargränssnitt för rumsliga/temporala frågor.

- Har förmåga att automatiskt fusionera registrerade sensordata.

- Dynamisk sensorallokering.

Med interoperabilitet avses här förmågan att enkelt kunna anpassa ett system till olika sensortyper där sensordata kan vara inbördes heterogena. Kravet på interoperabilitet har uppstått genom att Försvarsmakten har blivit engagerad i ett stort antal internationella operationer där samarbete och utbyte av data med andra länders försvarsmakter och med olika försvarsorganisationer, såsom t ex NATO måste kunna ske på ett enkelt sätt. Sensordataoberoende är besläktat med dataoberoende i traditionella databassystem. Metodiken används här också för att göra det enkelt för användaren *att hantera stora datamängder* genererade av multipla sensorer, *minska*

arbetsbördan för användarna, för att *öka tilltron till systemet* förutom att stödja interoperabiliteten. Vidare kan sensorer allokeras till systemet på ett dynamiskt vis med hänsyn till om väder eller ljusförhållanden förändras över tiden. Allt detta kommer att beskrivas ytterligare i avsnitt 4.1.2 nedan.En fördjupad beskrivning av frågespråket återfinns i Appendix B och F, där det senare arbetet också beskriver hur mer komplexa frågor kan ställas till ΣQL.

Tjänstekonceptet har sin grund i att traditionella programstrukturer har visat sig vara olämpliga när man i nätverk behöver svara mot de krav som ställs på olika ledningssystem eftersom olika aktörer har olika roller och därför också olika krav på de data som behövs i den aktuella arbetssituationen. Därför har istället begreppet tjänst introducerats. En tjänst är en abstraktion som beskriver hur producenter kan åstadkomma nytta för konsumenten utan att man i detalj behöver beskriva hur detta realiseras. Nyttan av en tjänst åstadkoms genom att en producent levererar en prestation, vilken ger en effekt hos/för konsumenten. I ΣQL består tjänsten av att frågespråket levererar efterfrågade data på ett sensordataoberoende sätt, i enkla termer som kan förstås även av användare som inte är IT-experter.

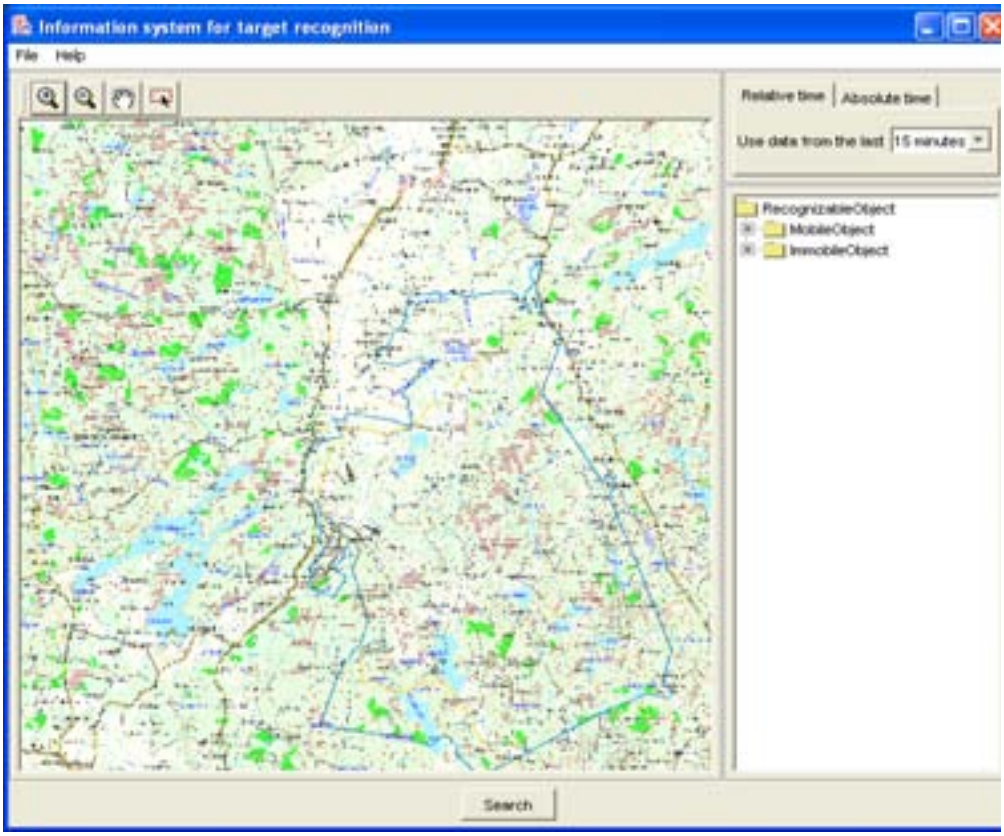
4.1.1 VisualΣQL

Tyngdpunkten vid konstruktionen av det visuella gränssnittet till frågespråket har varit att användaren ska beskriva sina behov utifrån termer som är naturliga för denne och inte genom en direkt koppling till sensorerna. Därför är gränssnittet utformat så att användaren beskriver *var, när* och *vad,* se Figur 2. *Var* är en enkel begränsning i rummet och det har realiserats så att användaren markerar ett område på en karta. Vilken tid som är aktuell väljer användaren antingen genom att välja start och sluttid eller genom att specificera tiden i absoluta termer, t ex de senaste 30 minuterna. Eftersom detta är ett system som fokuserar på markspaning får användaren själv välja vad han söker efter genom att välja mellan möjliga objekt i en lista. I de enkla fallen räcker detta för att ställa en grundläggande fråga mot systemet.

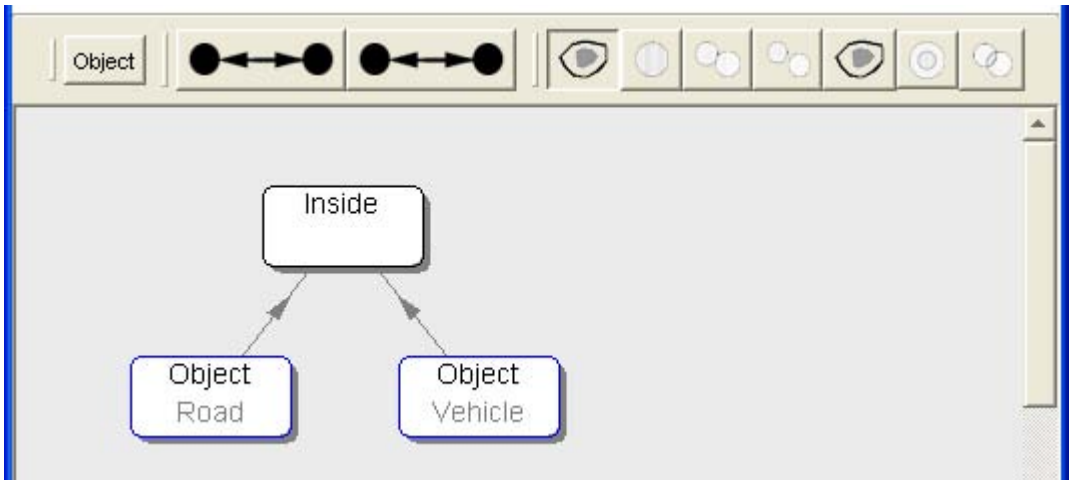
I det mer avancerade fallet har användaren möjlighet att sätta begränsningar på resultatet. Dessa begränsningar formuleras genom att relationer grafiskt kopplas till olika objekt. Relationerna kan vara begränsningar i rum, tid, egenskaper eller kombinationer av dessa, se Tabell 1. Relationerna finns grafiskt organiserade i olika paletter som användaren väljer från och ritar upp objekten och relationerna, se Figur 3.

Tabell 1: Exempel på relationer

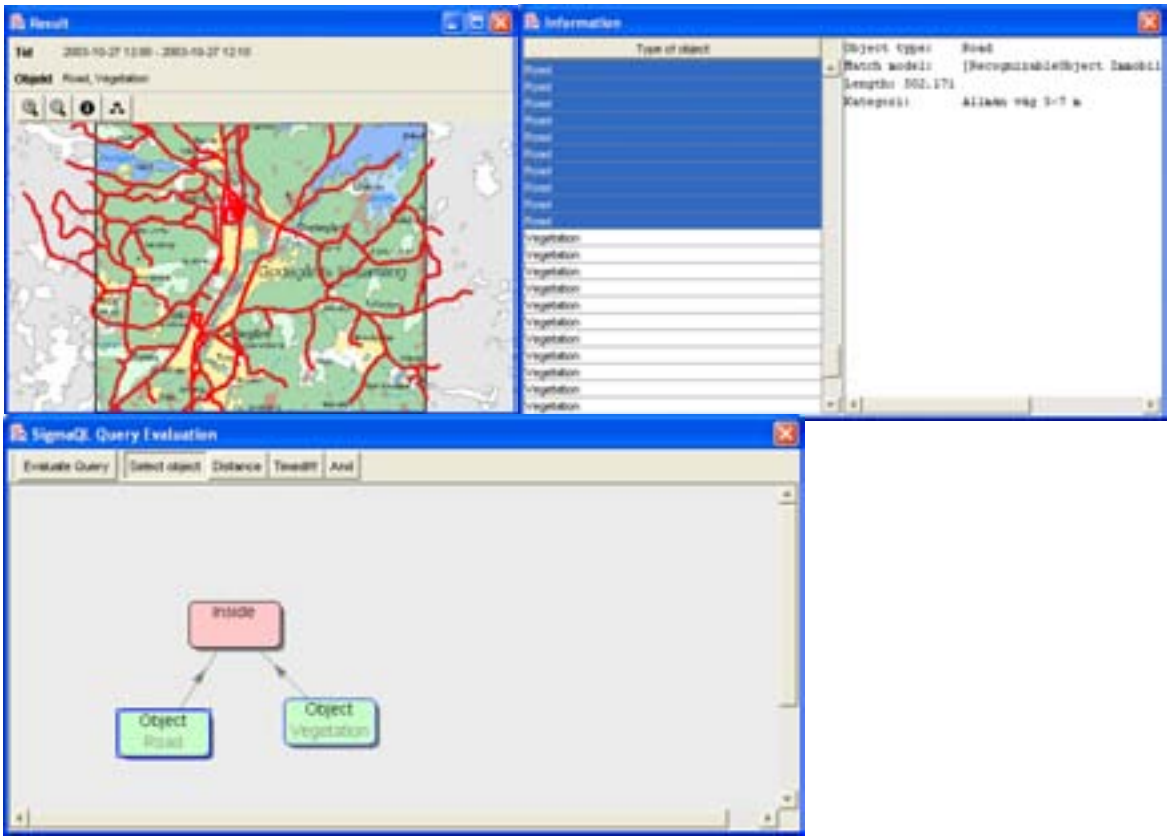
Rum	inuti avstånd mindre än 50 meter norr om bakom
Tid	före mellan klockan 13 och klockan 15 mindre än 1 timme emellan
Egenskap	namnet är Linköping hastigheten är större än 50 km/h färgen är röd
Kombinationer	Objekten ska mötas (vara vid samma tid vid samma plats)



Figur 2. Exempel på standardfråga med aktuellt intresse område, tid och objekttyp.



Figur 3. Fråga med tilläggsvillkor.



Figur 4. Illustration till resultatpresentation.

Resultatet av en sökning kan presenteras på tre sätt; genom en karta, en tabell och genom frågeevaluering, Figur 4. På kartan ritas objekten helt enkelt ut på den kända positionen. I tabellen finns en lista på alla de funna objekten med möjlighet att också se deras olika egenskaper. På kartan och i tabellen presenteras de objekt som uppfyller rums-, tids- och typbegränsningarna. I frågeevalueringen får man se de grafiskt uppritade begränsningarna. När användaren väljer någon av rutorna i grafen så väljs motsvarande objekt ut i listan och på kartan ritas de med annan färg. På så sätt kan användaren enkelt se vilka objekt som besvarar de olika delarna av den ställda frågan.

Aspekter som berör osäkerheter i indata har också behandlats i detta arbete vilket finns redovisat i Appendix C och D.

4.1.2 Sensorintegration

Frågesystemet har en mekanism för att automatiskt välja ut lämpliga sensordata för att besvara en fråga. Systemet kan dessutom välja ut lämpliga måldetektions- och måligenkänningsalgoritmer för att applicera på aktuella sensordata. Detta bygger på ett ontologiskt kunskapssystem som presenteras i Appendix E samt i [4]. Eftersom det är enkelt att lägga till information om nya sensortyper och målextraktionsalgoritmer i kunskapssystemet kan nya sensorer och algoritmer enkelt kopplas in så att de kan användas för att besvara frågor i frågespråket.

För att frågespråket ska kunna tillgodogöra sig data från en viss typ av sensor krävs förutom att information om sensortypen läggs in i kunskapssystemet att meta-data och sensordata från sensorn lagras i den databas frågespråket använder när den letar efter data att använda för att besvara en fråga. Över tiden behöver således sensorns täckningsområden, sensordata och/eller ev måldetektioner/observationer/målspår lagras i denna databas.

Den föreslagna agentarkitekturen för att hitta sensor- och algoritmresurser förstärker frågespråkets förmåga att hantera olika sorters datakällor. Om denna arkitektur införs behöver inte sensordata och måldetektioner/observationer/målspår lagras i frågespråkets egen databas utan agenterna letar då reda på rätt information i sensornätet för att besvara en fråga. Informationen som behövs för att besvara frågan kan således vara lagrad var som helst i sensornätet. Agentarkitekturen beskrivs övergripande nedan och mer i detalj i Appendix H.

4.1.3 Fusion

Fusionstanken i frågespråket är ursprungligen att fusionera olika delresultat från olika datakällor. Detta görs i ett moment, före presentationen för användaren. Generellt antas dessa delresultat vara kvalitativa. Fusionsfunktionaliteten som implementerats utvecklades dock för en specifik tillämpning, *igenkänning av markmål* [1]. De olika igenkänningsresultaten var kvantitativa, således mer informativa. Detta ökar förutsättningarna för ett lyckat fusionerat resultat.

Figur 5. Misstänkta måls dimensioner respektive orientering skattas i lågt upplösta data. Bilderna visar två olika tolkningar, med avseende på målets orientering. Fusionsmetoden bevarar båda dessa, för värdering i högupplösta data. På så sätt ökar chansen att hitta optimal matchning mellan data och målmodeller.

För att ytterligare öka förutsättningarna för hög kvalité på slutresultatet bör fusionsmetodiken *känna till* den bakomliggande igenkänningsprocessen. Det visade sig att igenkänning kunde genomföras i två steg. I det första skattades målets attribut (egenskaper resp. tillstånd, såsom längd resp. orienteringsriktning). Detta kan ses som algoritmens första *tolkning* av bilden. Tolkningar genom olika algoritmer bör typiskt vara likartade, eller distinkt olika - kvalitativt sett. Till exempel är det rimligt att tänka sig att olika algoritmers skattning av orienteringsriktning antingen är likartade, eller skiljer sig åt med ett halvt varv. Som exempel, se Figur 5.

I det andra steget matchades målet mot lagrade modeller, med de skattade attributen som utgångspunkt. Detta innebar finjusteringar av de skattade attributvärdena. Optimeringen genererade det kvantitativa resultatet - ett mått på graden av matchning mellan målet och de olika modellerna. Detta steg i igenkänningen kan ses som *värdering* av attributskattningen, dvs värdering av tolkningen.

Det observerades att värderingen av attributskattning inte beror på vilken algoritm som genererat skattningen. Således kan algoritmerna låtas distribuera tolkningar mellan varandra utan risk för s k *data incest* [3]. Syftet är att öka chansen att hitta den bästa kombinationen av tolkning och algoritm/data.

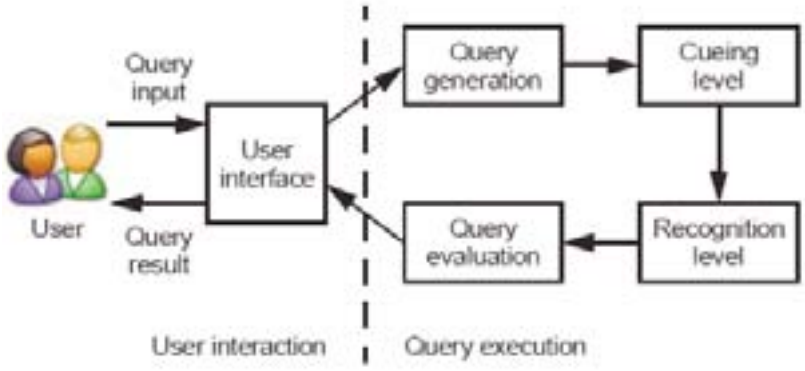
Distributionen av tolkningar/skattningar är att betrakta som en första del av fusionsmetoden i denna tillämpning för frågespråket.

I fusionsmetodens andra del sorteras alla gjorda värderingar enligt de kvantitativa resultaten från respektive matchning. Dessa matchningsvärden tolkas så att värden från olika algoritmer antas vara direkt jämförbara.

För de objektmodeller som matchats av mer än en algoritm sparas endast resultatet med det högsta matchningsvärdet. Det högsta anses alltså vara det mest korrekta. Detta grundar sig på antagandet att matchningsvärden inte kan vara *förrädiskt höga*. Däremot kan höga brusnivåer i vissa data, samt dåliga upptagningsförhållanden för vissa sensorer vid det aktuella tillfället, resultera i *för låga* matchningsvärden från vissa algoritmer och data. En grundlig genomgång av fusionsmetodiken ges i Appendix G.

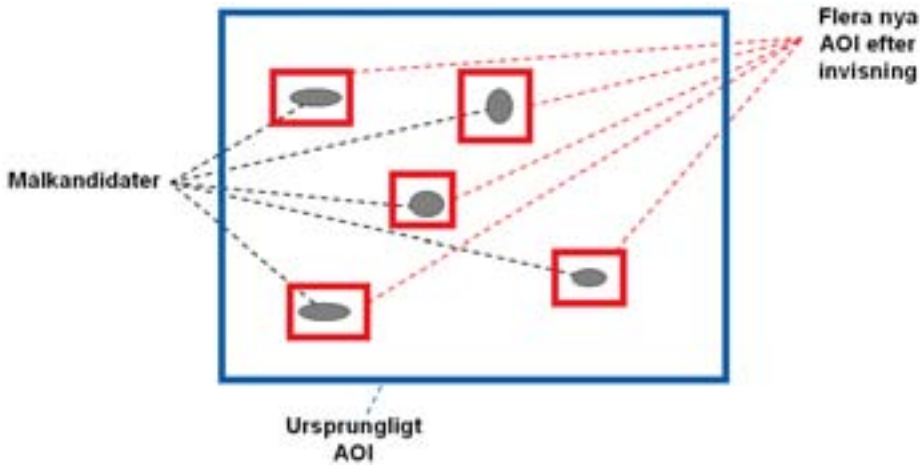
4.1.4 Frågeexekvering

Frågeexekveringen, dvs allt som händer mellan det att systemet tagit emot en fråga från en användare till dess att svaret på frågan presenteras, presenteras övergripande i Figur 6. Mer detaljer om frågeexekveringen finns i Appendix E.



Figur 6. Översikt av frågeexekveringsprocessen.

De två grundläggande nivåerna i frågeexekveringen är invisningsnivån och igenkänningsnivån. Den första av dessa är invisningsnivån, se Figur 7, där området användaren är intresserad av (AOI, dvs *Area Of Interest*) kan vara stort och tidsintervallet användaren är intresserad av kan vara långt. Med invisning i detta sammanhang menas att hitta målkandidater och deras positioner. Med mål menas de typer av objekt en användare frågar efter; det kan vara vilka typer av objekt som helst som systemet har resurser (sensorer, algoritmer och/eller databaser) för att kunna hitta.



Figur 7. Översikt av invisningsnivån - nya AOI genereras där målkandidater hittas.

Om flera olika detektionsalgoritmer och/eller sensorer använts genereras en ny målkandidat för varje målkandidat som de enskilda sensorerna/algoritmerna genererat. Det går också att klustra målkandidater så att de målkandidater som ligger nära varandra slås ihop till en målkandidat innan dessa skickas vidare till nästa nivå i processen.

På invisningsnivån har hittills använts simulerade marksensornät med akustiska sensorer och simulerad CARABAS i scenarion av typen övervakning av markgående fordon i terräng.

Figur 8. Översikt av igenkänningsnivån - klassificering av målkandidater.

I den andra nivån, igenkänningsnivån, se Figur 8, granskas närmare de potentiella målen från invisningsnivån. Igenkänningsnivån är indelad i två huvudsakliga steg, attributestimering och modellmatchning. I attributestimeringen används algoritmer som relativt snabbt kan skatta attribut, t ex position, orientering och storlek på målkandidaterna. Vissa målkandidater kan också avfärdas i detta steg om det visar sig att sensordata som används inte tyder på att det förekommer några mål på den angivna positionen.

Eftersom flera olika attributestimeringsalgoritmer (för samma eller olika sensordata) med fördel kan exekveras parallellt krävs fusion av hypoteserna med avseende på position, orientering, storlek, etc innan de går vidare till nästa steg i processen. Detta kallas attributfusion och sker genom en form av klustringsmetodik som beskrivs övergripande på annat ställe i detta dokument och mer detaljerat i Appendix G.

Ett antal olika attributestimeringsalgoritmer som estimerar position, orientering och storlek på mål i sensordata från IR- och CCD-kameror samt från skannande laserradar har testats. Vissa algoritmer fungerar på data från flera olika sensordatatyper medan andra bara fungerar på en viss typ av sensordata, se vidare Appendix A.

I modellmatchningssteget används algoritmer som matchar sensordata som innehåller målkandidaterna i tidigare steg mot modeller av målen för att försöka klassificera den aktuella måltypen. Eftersom det är ett mycket beräkningstungt problem att matcha sensordata mot alla modeller i ett modellbibliotek i alla olika konfigurationer (skalningar, orienteringar, artikulationer, etc) är det mycket värdefullt om attributestimeringssteget kan ge en indikation på hur stora målen är (avfärdar genast alla kandidater i målbiblioteket som har annan storlek), hur de är orienterade (mycket bättre starthypotes kan användas för hur modellen ska vridas när den ska matchas mot sensordata) etc kan sökrymden vid modellmatchningen kraftigt reduceras vilket snabbar upp måligenkänningen. I detta steg kan många felaktiga hypoteser om målkandidater avfärdas och målkandidater som klassas som mål kan ges en närmare klassificering beroende på vilken modell från modellbiblioteket som passat bäst vid matchningen.

Eftersom flera olika modellmatchningsalgoritmer (på samma eller olika sensordata) kan exekveras parallellt krävs fusion av hypoteserna om målets typ och andra attributvärden innan re-

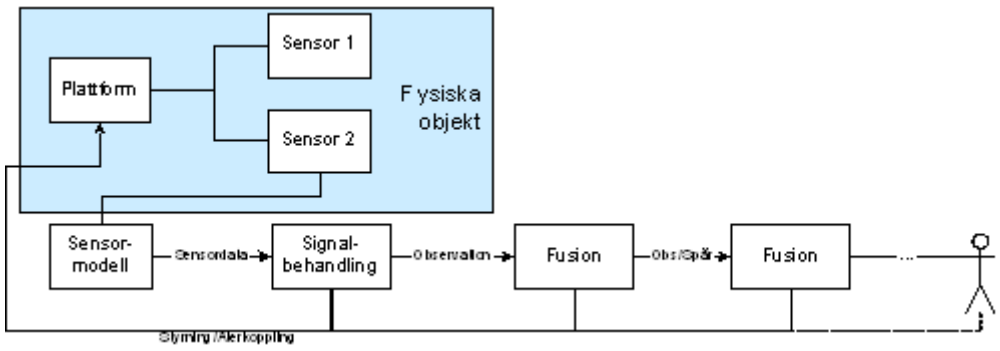
sultatet går vidare till nästa steg i processen. Detta kallar vi matchningsfusion och sker med en form av röstningsmetodik som beskrivs mer detaljerat i Appendix G.

När systemet fått fram klassificerade mål med skattade attribut (position, orientering, storlek, artikulering, etc) evalueras dessa mot de krav på attributvärden och spatiala och temporala relationer mellan objekt som användaren av frågespråket specificerat i sin fråga. När detta är klart är resultatet klart att presenteras.

4.1.5 Arkitektur

Problemet som behandlas i detta avsnitt är huruvida frågespråket ΣQL automatiskt kan hitta lämpliga datakällor på nätet och hur systemet kan få åtkomst till data från dessa källor.

Uppkopplade på nätet finns olika sensorplattformar som kan vara allt från en soldat till en UAV, ett markgående spaningsfordon, ett fartyg på havet eller en fast monterad stolpe på vilken man kan montera en eller flera sensorer. Sensorer kan i sin tur vara allt från ögonen eller öronen på en soldat till en radar eller en elektrooptisk sensor i en UAV.



Figur 9. Beskrivning av informationsflöde från sensor till användare.

Hela konceptet överförs enkelt från den militära världen till exempelvis civila katastrofsituationer där en konsistent lägesbild av ett område önskas.

I Figur 9 ges en schematisk beskrivning av hur informationsflödet från sensor till användare kan se ut.

För att få hög robusthet krävs att systemet är distribuerat och att viktig information finns lagrad på flera ställen i nätet (redundans). Eftersom mängden sensordata i ett sådant här system är mycket stor krävs att data behandlas så nära källan som möjligt och att förädlad information kommuniceras mellan de olika noderna i nätet.

Det gäller alltså att på ett intelligent sätt överföra minimala datamängder med så stort informationsinnehåll som möjligt vid så få och lämpliga tidpunkter som möjligt. Intelligenta agenter lämpar sig väl för denna typ av autonomt uppträdande. Exempelvis kan de förhandla med andra agenter (som har andra mål än agenten själv) om vad som är lämpligt att göra i olika situationer.

Sensorer sitter på plattformar och levererar sensordata. Med sensordata menas i detta sammanhang data från en sensor innan någon måldetektion har genomförts. Exempel på sensordata är således en IR-bild, en radarplott, en SAR-bild och en punktskur från en laserradar.

Sensordata i sig är inte mycket värda. För att få ut något värde ur sensordata måste man extrahera information ur datamängden, exempelvis genom måldetektion. Med måldetektion menas här att man i sensordata försöker detektera ett eller flera objekt som man är intresserad av att presentera i lägesbilden. Användaren av ett informationssystem ska inte behöva bry sig om vilka plattformar, sensorer och databaser som finns tillgängliga för att besvara hans fråga. Systemet ska istället automatiskt hitta lämpliga datakällor (sensorer, databaser, etc) med den information som behövs för att besvara en fråga. För att lösa problemet föreslås en arkitektur baserad på intelligenta agenter, med två typer av agenter. Den första typen är en resursallokeringsagent och den andra en dataåtkomstagent.

När en användare ställer en fråga skapas en resursallokeringsagent. Denna agent är sedan ansvarig för att leta upp och hålla reda på de resurser som behövs för att besvara frågan över tiden. Resurser som agenten letar upp och håller reda på är sensordata, algoritmer för behandling av sensordata (måldetektionsalgoritmer etc), observationer och målspår som kan behövas för att besvara frågan. Dessa resurser finns tillgängliga antingen i databaser eller i sensorplattformar, men är i båda fallen åtkomliga via nätet.

När resursallokeringsagenten har allokerat resurser för att besvara frågan skapas en dataåtkomstagent för varje sensordata-, observations- och målspårsresurs. Om resursen är sensordata kommer dataåtkomstagenten att ta med sig information om vilka algoritmer som ska appliceras för att få lämpliga observationer. En lämplig sensordatabearbetningsnod väljs (en bearbetningsnod med lämpliga prestanda och ledig beräkningskraft så nära datakällan som möjligt) av agenten och algoritmerna appliceras på aktuella data. Observera att tillgång till data om sensordata (metadata), bl a. täckningsområden för insamlade data, är av central betydelse för att resurserna i nätet ska kunna utnyttjas på ett bra sätt.

En mer omfattande och detaljerad beskrivning av agentkonceptet presenteras i Appendix H.

4.2 Terränganalys

Analys av terrängen är en naturlig del i ett beslutsstöd för markspaning. Terrängen är central vid t.ex. analys av skydd, synfält eller framkomlighet. För att terrängdata ska vara användbart i ett frågespråk är det nödvändigt att systemet kan tolka terrängen i termer som användaren själv kan tolka och resonera om på ett naturligt sätt. Ett led i processen att tolka terrängen består i att klassificera ytan i olika meningsfulla delar, d.v.s att detektera relevanta terrängobjekt, som t ex kullar eller diken. De terrängdata som använts i IS-MS kommer från en flygburen, skannande laserradar (TopEye) med en mycket hög upplösning (≤ 0.5 m). Dessa data kan användas till att detektera väsentliga terrängobjekt av typ diken och kullar. För att terränganalys ska kunna ske i nära realtid måste effektiva metoder för detektion av objekten utvecklas. Detta är ett komplext problem eftersom en mycket stor mängd data måste lagras, bearbetas, distribueras och visualiseras, speciellt då hög upplösning krävs. Vidare krävs också metoder för reduktion av data, där de relevanta delarna kan behållas, men de mindre viktiga delarna tas bort [5]. En metod för att uppnå dessa mål baserad på bestämning av symboliska ytelement har utvecklats, Appendix I. Ytelementen används sedan i en effektiv matchningsprocess för att detektera väsentliga terrängobjekt. Den symboliska strukturen är en blandning av en reguljär och en irreguljär datastruktur där vissa irreguljärt distribuerade, viktiga höjddatapunkter behålls. Den huvudsakliga fördelen med denna struktur är att den tillåter en effektiv matchning mot användarens frågor om beskrivna terrängobjekt. Förutom att vara en lämplig struktur för att användas i ett frågespråk kan strukturen också användas som ett index till en höjdmodell. Detta skall ses som ett komplement till en höjdmodell i hög upplösning och inte som en ersättning.

Framkomlighetsanalys i terräng innebär analys och visualisering av möjligheten att ta sig fram i ett godtyckligt terrängavsnitt med ett givet fordon av någon typ, se vidare Appendix J. Slutmålet är att generera ett eller flera förslag på sätt att ta sig från en position till en annan. Dessa förslag ska vara de bästa ur någon given synvinkel, t.ex. de kortaste eller de säkraste. Framkomlighetsanalys är ett komplext problem som beror av många faktorer, men som också har många olika tillämpningar. Förutom att vara ett stöd till militära beslutsfattare vid planering kan också t.ex. utryckningsfordon vid skogsbränder vara i behov av en framkomlighetsanalys.Tyvärr är det underlag som finns i allmänhet inte tillräckligt för att framkomlighetsanalys ska kunna göras med en adekvat noggrannhet. Inte minst saknas 3D-data i tillräckligt hög upplösning men även data, om t.ex. jordartstyper med tillräcklig upplösning saknas.

Inom ramen för detta projekt har tre arbeten [6], [7] samt Appendix J utförts där några av ovanstående problem adresseras. Utgående från de terrängobjekt som detekterats i 3D-data samt data från Fastighetskartan [8] konstrueras ett nätverk av terrängobjekt med olika framkomlighetsegenskaper. Hantering av fullständiga terrängobjekt istället för rena höjddatapunkter gör analysen snabbare och gör att olika genererade förslag skiljer sig på ett meningsfullt sätt. De olika objekten förknippas sedan med olika kostnader beroende på svårigheten att ta sig igenom objektet. Kostnaden beräknas genom att en mängd olika faktorer vägs samman i en kvalitativ ansats. Metodens viktigaste egenskaper är att vara robust mot ofullkomligheter i data, vilket väsentligen är beroende av stora osäkerheter i ursprungsdata samt att data i många fall saknas. Sökning i nätverket sker med en kombination av Dijkstra's algoritm [9] och A*-algoritmen [9].

4.3 Situationsanalys

Det problem som behandlas i anslutning till situationsanalys berör associering av observationer av vägbundna fordon och kan ses som ett generellt stöd för lägesuppfattning. Detta omfattar en metod för associering av observationer av vägbundna fordon [10]. Med associering menas här att avgöra vilka observationer som härrör från samma fordon. Det finns flera olika syften med att associera observationer på detta sätt. Rent allmänt kan man säga att associering, om den är genomförbar, ger färre målspår och därmed en mindre komplex *lägesbild* att analysera och förstå. En fråga man vill kunna besvara är exempelvis hur många fordon som har observerats. Associering av observationer är ett sätt att göra detta. Med associering ökar också möjligheterna till en visualisering som ger en förståelse av vad som har genererat observationerna. En målsätt-

ning är att ge användaren en ungefärlig uppfattning om vilka rörelsemönster som finns i situationen.

Ursprunget till den här beskrivna metodutvecklingen återfinns i Appendix K, som beskriver ett generellt ramverk för situationsanalys. Delar av de idéer som där förs fram har sedan legat till grund för det fortsatta arbetet och kan sägas vara en vägledande vision för vad vi vill åstadkomma på längre sikt. Det som primärt bör tillföras vid en fortsatt utveckling av metoden är hantering av grupper av fordon.

Den utvecklade metoden för associering av observationer har två huvudkomponenter:

- skattning av lokal associationssannolikhet mellan varje par av observationer.

- global optimering för att finna den kombination av associationer som ger störst total sannolikhet.

Skattningen av lokal associationssannolikhet mellan ett par av observationer baseras på följande faktorer:

- Tidsdifferensen mellan observationerna.

- Längden på den kortaste färdvägen mellan observationsplatserna.

- En sannolikhetsfördelning för fordonets förväntade genomsnittliga hastighet.

- Eventuell typklassificering av observationerna.

- Graden av rationellt vägval som en färdväg mellan observationerna skulle innebära.

- En apriorifördelning av sannolikheten för olika grader av rationellt vägval.

- Skattad fordonstäthet.

Med rationellt vägval avses att vald färdväg inte utgör en lång omväg. Att en omväg måste ha tagits kan ibland konstateras genom att studera vägnätets topografi i kombination med den historia i form av målspår som finns för observationerna.

När alla parvisa lokala associationssannolikheter har beräknats görs en global optimering där observationerna delas upp i grupper. Varje sådan grupp representerar observationer av ett och samma fordon. Optimeringens målfunktion är

$$V = \prod_{i>j} P_{ij}\delta_{ij} + (1 - P_{ij})(1 - \delta_{ij})$$

där

P_{ij} är associationssannolikheten mellan observation i och j

δ_{ij} = 1 om observation i och j är associerade, 0 annars

Optimeringsmetoden bygger på Genetiska Algoritmer (GA) [11], vilket är en stokastisk sökmetod inspirerad av evolutionsteorin. GA bygger på att man har en uppsättning lösningar som i ett antal cykler genomgår slumpmässiga förändringar. I varje sådan cykel väljs de bästa ut för att generera en ny uppsättning lösningar som är slumpmässigt modifierade versioner av de tidigare. Metoden terminerar efter ett förutbestämt antal cykler, då den bästa lösningen som har genererats så långt returneras.

För internt bruk har en visualiseringsfunktion som utnyttjar färgkodning utvecklats, se Figur 10 Denna ger emellertid en svåröverskådlig bild när antalet observationer växer och ger inte heller någon uppfattning om tidsdimensionen.

Arbete pågår med att utveckla en metodik för att spela upp resultatet i simulerad tid. Där man har haft målföljning ser man en fordonssymbol som rör sig enligt målspåret. I den tidslucka som finns mellan två associerade målspår rör sig en annan symbol längs en hypotetisk färdväg mellan målspåren.

.

Figur 10. Målspår över en tidsperiod i stadsmiljö. Målspåren har anpassats till vägnätet för tydligare visualisering. Målspår som bedöms härröra från samma objekt har samma färg.

5. Sammanfattning

Det informationssystem för markspaning som beskrivits här är primärt avsett för att integreras med ett nätverksanpassat ledningssystem. Informationssystemet kan ses som ett beslutsstöd eller lagringsplats för olika tjänster som kan allokeras av enskilda användare. För närvande uppgår dessa tjänster till tre men kan efter behov utökas. Dessa tre tjänster utgörs av ett frågespråk med ett antal underliggande sensordatatjänster, situationsanalys samt en metod för generering av högupplösande terrängmodeller.

Frågespråket är för närvarande den mest komplexa tjänsten med förmåga att besvara dels enkla frågor, som inte kräver någon djupare kunskap om hur frågespråk används, dels mer komplexa frågor som kräver en mer omfattande kunskap. ΣQL kan hantera osäker information och kan dessutom ge stöd för underhåll av en lägesbild. Frågor kan ställas med hjälp av ett kraftfullt visuellt användargränssnitt. Frågespråket har också till uppgift att ge stöd till andra tjänster, t ex stöd för situationsanalys etc.

Frågespråket är också försett med andra egenskaper bland vilka kan nämnas ett effektivt redskap för fusion och sensorinteroperabilitet. Dessa egenskaper stöds båda av det ontologiska kunskapssystemet. Syftet med denna teknik, som beskriver data på ett hierarkiskt strukturerat sätt, är att också ge stöd vid frågegenereringen, vilket sker genom att peka på lämliga objekt i den ontologiska objektstrukturen.

Systemet kan allokera tjänster på olika nivåer, dvs direkt av användaren eller indirekt via direktallokerade tjänster. Exempel på tjänster av den senare typen är sensortjänster. När det gäller förmågan att allokera tjänster så sker detta på ett användaroberoende sätt. Detta sensordataoberoende gör det möjligt för användaren att hantera systemet utan att ha kunskap om sensorerna eller de data som dessa producerar.

Sensorer genererar data som är mer eller mindre osäkra, vilket beror på sensorernas egenskaper men också på väderleks- och ljusförhållanden. I detta arbete har vi kunnat visa att osäker information kan hanteras på ett robust sätt, se bl a [1] men också Appendix C och D. Osäkerhet i data påverkar också svaren på frågorna som ställs. Detta indikeras genom att systemet förser svaren med trolighetsmått (eng. belief values) med normaliserade värden mellan 0 och 1. Exempel på detta är att en T32 har observerats med ett trolighetsmått på 0.80. Detta bör tolkas som att man kan ha en relativt hög tilltro till svaret.

Andra tillämpningar av detta system kan förutom militära också vara civila, t ex krisledning. Att hantera stora datamängder vid krisledning är av central betydelse och speciellt i sådana sammanhang där behov finns av att ge stöd för databrytning (eng. data mining). Frågespråket i detta informationssystem lämpar sig väl för denna problematik.

För metodiken för framkomlighet kan konstateras att det fortfarande krävs mycket arbete med att ta fram underlaget för framkomlighetsanalysen, många datakällor och nya analysmetoder behövs. Inte minst måste metoder för fusion av olika terrängdatakällor, i syfte att bestämma relevanta terrängobjekt, utvecklas. Själva analysmetoden måste också vara robust för att kunna hantera de fall då data saknas, är osäker eller av varierande typ.

Hur pass väl den utvecklade metoden för associering av observationer av vägbundna fordon fungerar beror i huvudsak på förhållandet mellan fordonstäthet och observationstäthet. Den fungerar speciellt bra för estimering av antalet fordon i relativt gles trafik. Bra klassificeringsinformation får samma effekt som minskad fordonstäthet, vilket innebär att bra klassificering är ett sätt att hantera större fordonstäthet. För framtiden är en utvidgning av metoden för att associera observationer av grupper av fordon intressant. Dels är uppträdandet i grupper vanligt i många sammanhang, dels ger detta möjlighet att utnyttja gruppernas sammansättning för beräkning av likhetsmått mellan grupper, vilket skulle ge en ännu bättre förutsättning för associering.

Referenser

[1] E. Jungert et al., *From Sensor to Decision*, FOI user report, FOI-R--1041--SE, December 2003.

[2] M. Tyskeng, *MOSART - Instructions for use*, FOI-R--1098-SE, dec 2003.

[3] E. Sviestins, *synergetic partnership*. Militärteknisk tidskrift 1:14-19, 2003.

[4] T. Horney, *Design of an ontological knowledge structure for a query language for multiple data sources*, FOI-rapport, 2002.

[5] F. Lantz, E. Jungert, *Dual Aspects of a Multi-Resolution Grid-Based Terrain Data Model with Supplementary Irregular Points*, Proceedings of the 3rd International Conference on Information Fusion, Paris, France, July 10-13, 2000.

[6] S. Edlund, *Driveability Analysis*, Linköpings Universitet, LITH-IDA-EX-04/031-SE, 2004,(FOI rapport FOI-R--1241--SE, maj, 2004).

[7] K. Gustafsson, J. Hägerstrand, *Development of a Neighbourhood Graph for trafficability Analysis*, Linköpings Universitet, LITH-IDA-EX-05/047--SE, (FOI rapport FOI-R--1698--SE, juni, 2005).

[8] Lantmäteriet, /Produktbeskrivning: GSD-Fastighetskartan i Shape och MapInfo-format/, http://www.lm.se/gsd/fastighetskartan/fastshmi.pdf, August 2004, visited Jan 15, 2005.

[9] V.M.Jiménez, A. Marzal, J. Monné, *A Comparison of Two Exact Algorithms for Finding the N-Best Sentence Hypotheses in Continuous Speech Recognition*, From the Proceedings of

the 4th European Conference on Speech Communication and Technology, EUROSPEECH-95, Madrid, 1995, pp. 1071-1074.

[10] J. Fransson, *Situationsanalys: Associering av observationer av vägbundna fordon*, FOI-memo 1366, Juni, 2005.

[11] E. Falkenauer, "Genetic Algorithms and Grouping Problems", Wiley, ISBN 0471 971502, 1998.

Projektpublikationer, 2004-2005

Tidskiftsartiklar

S.-K. Chang, E. Jungert and X. Li, *A Progressive Query Language and Interactive Reasoner for Information Fusion,* accepterad för publicering i the Journal of Information Fusion, Elsevier.

S.-K. Chang, G. Costagliola, E. Jungert, F. Orciuoli, *Querying Distributed Multimedia Databases Data Sources in Information Fusion Applications*, IEEE Trans. on Multimedia, Vol. 6, No. 5, October 2004, 687-702.

Bokkapitel

J. Alfredson, G. Derefledt, E. Jungert, *Kunskapsrepresentation för utveckling av gemensam lägesförståelse i närverk* (in Swedish)*, in Samhällsförsvar - nya hot och ökat internationellt engagemang*, G. Derefeldt, H. Friman (Eds.), Utrikespolitiska institutet, Stockholm 2004, pp 177-192.

Konferensbidrag

K. Silvervarg, E. Jungert, *Uncertain topological relations for mobile point objects in terrain,* Distributed multimedia Systems, Banff, Canada, Sept 5-7, 2005.

F. Lantz, E. Jungert, *Context Fusion for Driveability Analysis*, Proceedings of the international conference on Information Fusion (Fusion'05), Philadelphia, PA, July 25-29, 2005.

K. Silvervarg, E. Jungert, *A Visual Query Language for Uncertain Spatial and Temporal data*, Visual Information System, Amsterdam, July 9, 2005.

K. Silvervarg, E. Jungert, *Visual specification of spatial/temporal queries in a sensor data independent information system*, Proceedings of the Workshop on Visual Language and Computing (VLC'04), San Francisco, California, September 8-10, 2004.

S.-K. Chang, G. Costagliola, E. Jungert, G. Casella, T. Horney, X. Li, *An Architecture for Interactive Query Refinement in Sensor-based Information Fusion Systems*, proceedings of the

Workshop on Visual Information Systems (VIS'04), San Francisco, California, September 8-10, 2004.

S.-K. Chang, E. Jungert, *Iterative Information Fusion using a Reasoner for Objects with Uniformative Belief Values*, Proceedings of the 7th international Conference on Information Fusion (Fusion'04), Stockholm, Sweden, June 28- july 1, 2004.

T. Horney, J. Ahlberg, E. Jungert, M. Folkesson, K. Silvervarg, F. Lantz, J. Franssson, C. Grönwall, L. Klasén, M. Ulvklo, *An Information System for target recognition*, Proceedings of the SPIE conference on defense and security, Orlando, Florida, April 12-16, 2004, Vol. 5434, pp 163-175.

Examensarbeten

S. Edlund, *Driveability Analysis*, Linköpings Universitet, LITH-IDA-EX-04/031-SE, 2004,(FOI rapport FOI-R--1241--SE, maj, 2004).

K. Gustafsson, J. Hägerstrand, *Development of a Neighbourhood Graph for trafficability Analysis*, Linköpings Universitet, LITH-IDA-EX-05/047--SE, (FOI rapport FOI-R--1698--SE, juni, 2005).

FOI-rapporter

M. Folkesson, C. Grönwall, E. Jungert, *A Fusion Framework for coarse-to-fine Target Recognition*, FOI-rapport, FOI-R--xxxx--SE, September, 2005.

FOI-Memo

T. Horney, E. Jungert, *Agent architecture for a query language in NVD-Environment* (in Swedish), FOI-memo 1025, September, 2004.

E. Jungert m fl, *Verksamheten inom Försvarsmaktsprojektet Informationssystem för Markspaning (IS-MS),* FOI-memo 1126, December 2004.

J. Fransson, *Situationsanlys: Associering av observationer av vägbundna fordon*, FOI-memo 1366, juni, 2005

E. Jungert, T. Horney, P. Follo, *Integration av CARABAS med frågrespråket ΣQL för detektering av markfordon,* FOI-memo 1430, september 2005.

Särtryck av utvalda publikationer

Appendix A

An Information System for Target Recognition

SPIE, Aerosense, Orlando, FA, April 12-16, 2004.

Horney, T., Ahlberg, J., Jungert, E., Folkesson, M., Silvervarg, K., Lantz, F., Franssson, J., Grönwall, C., Klasén, L., Ulvklo, M.

An Information System for Target Recognition

Tobias Horney[1], Jörgen Ahlberg[2], Erland Jungert[1], Martin Folkesson[1], Karin Silvervarg[1], Fredrik Lantz[1], Jörgen Fransson[1], Christina Grönwall[3], Lena Klasén[3], Morgan Ulvklo[2]

1) Dept. of Data and Information Fusion, Div. of Command and Control Systems
2) Dept. of IR Systems, Div. of Sensor Technology
3) Dept. of Laser Systems, Div. of Sensor Technology

Swedish Defence Research Agency (FOI)
SE-581 11 Linköping, Sweden

{tobho, jorahl, jungert, marfol, karin, flantz, jorfra, stina, lena, morgan}@foi.se

Abstract

We present an approach to a general decision support system. The aim is to cover the complete process for automatic target recognition, from sensor data to the user interface. The approach is based on a query-based information system, and include tasks like feature extraction from sensor data, data association, data fusion and situation analysis.

Currently, we are working with data from laser radar, infrared cameras, and visual cameras, studying target recognition from cooperating sensors on one or several platforms. The sensors are typically airborne and at low altitude.

The processing of sensor data is performed in two steps. First, several attributes are estimated from the (unknown but detected) target. The attributes include orientation, size, speed, temperature etc. These estimates are used to select the models of interest in the matching step, where the target is matched with a number of target models, returning a likelihood value for each model. Several methods and sensor data types are used in both steps.

The user communicates with the system via a visual user interface, where, for instance, the user can mark an area on a map and ask for hostile vehicles in the chosen area. The user input is converted to a query in ΣQL, a query language developed for this type of applications, and an ontological system decides which algorithms should be invoked and which sensor data should be used. The output from the sensors is fused by a fusion module and answers are given back to the user. The user does not need to have any detailed technical knowledge about the sensors (or which sensors that are available), and new sensors and algorithms can easily be plugged in to the system.

Keywords: Automatic target recognition, multisensor fusion, query languages

1 Introduction

This paper describes an information system for recognition of ground targets, mainly various types of military vehicles. Our aim is to cover the complete process of target recognition, from the sensors to the decision support in a command and control system. Applications are surveillance and intelligence in a network centric defence. The main goals of our project have been to:

- Develop signal- and image processing methods for target recognition;
- Demonstrate improved target recognition using data fusion;
- Develop an information system for target recognition based on a query language, with a powerful visual user interface, for heterogeneous sensor data sources;
- Demonstrate how the decision support tool in the information system can support situation analysis.

The outline of this paper is as follows. A background to this kind of information systems is given in the remaining of this section. An overview of our system is given in Section 2, and its processing of user queries is described in Section 3. The sensor data analysis, i.e., the target recognition algorithms, is briefly described in Section 4, and how the output is fused is described in Section 5. The supporting modules for terrain analysis and situation awareness are prestented in Section 6 and Section 7. Finally, our implementation is treated in Section 8 and concluding remarks given in Section 9.

1.1 Background

The next generation decision support tools for situation and impact analysis [1] will generally require input from a large number of sensors located on different platforms. These decision support tools will also be integrated in a communication network enabling network centric warfare applications. Systems designed for this type of applications, i.e., basically command and control, will obviously become very complex. Hence, they will either require users that are well trained in using complex technical information systems *or* efforts must be made into the design of *usable systems* [2]. The latter alternative is clearly preferable since most users will place a higher trust, or confidence, in the system at the same time as they will be able to focus their efforts on their primary tasks. Such a system requires capabilities to select sensors and algorithms for sensor data analysis without any user interference. Systems with this capacity are said to be *sensor data independent* [3] and help in avoiding a situation where the users of an information system need a deep understanding of the sensors, when certain sensor types are usable, how to analyse the sensor data, etc. Sensor data independence is basi-

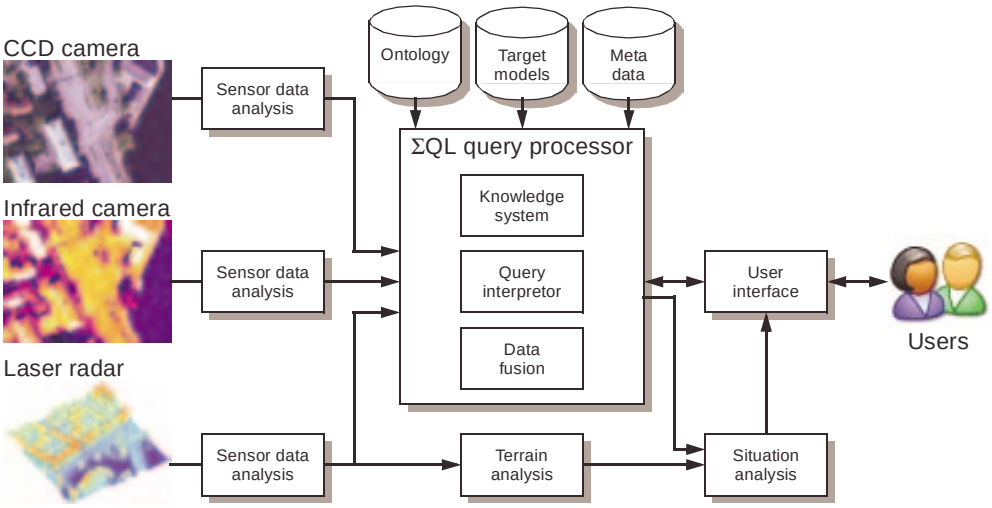


Figure 1: System overview.

cally similar to data independence in database design, where it was first introduced to allow modifications of the physical databases without affecting the application programs [15]. This was a powerful innovation in database design and generally in information technology. Sensor data independence can from a practical viewpoint be carried out by means of an ontology combined with an ontological knowledge-base [4, 5]. A nice aspect of this solution is that new sensor types and sensor data analysis algorithms can be integrated simply by updating the knowledge-base. The user of the information system does not even necessarily need to be informed.

Another important aspect when designing systems for command and control applications, and where sensors are the primary input data sources, is that the users should be allowed to define relevant application oriented goals [6]. This should enable the system to acquire the information needed to solve the problem associated with the given goal. Consequently, a system of this type must be *goal driven* or, as will be seen subsequently, *query driven*.

Various means can be used to accomplish the user-defined goals. In this work a query language for sensor data, where the different occurring sensor data types are homogeneous with respect to the sensors, is presented. The query language is called ΣQL. ΣQL uses a visual user interface [7] for application of the user defined queries and an ontology with an ontological knowledge-base to achieve sensor data independence. A more thorough description of ΣQL can be found in [8,9].

This paper focuses on the information system as such and does not in detail describe the sensor data analysis algorithms developed and used in the information system. The sensor data analysis used by the information system is more thoroughly presented in [10,18,20,21].

One of the objectives of the information system is to acquire information from the sensors and deliver input to a module for situation assessment [11]. To support the latter, a technique for generation of a symbolic digital terrain model in very high resolution (~ 0.5 m) has been developed. This includes a filtering method for determination of various terrain features [12]. However work has been done on the situation assessment and terrain analysis parts, those are not integrated in the current system implementation.

2 System structure

In this section an overview of the system structure is presented. Also, the user interface, the ΣQL query processor and other core functionality are described. The data fusion portion of the system is described in Section 5.

2.1 Overview

An schematic overview of the system can be found in Figure 1. The system is, from right to left, divided into the visual user interface, the query processor, and the sensor nodes to which the sensor data sources are attached. The sensor nodes include means for target recognition and terrain data analysis. The query processor includes a knowledge system, a query interpretor and a sensor data fusion module. Connected to the query processor is an ontology, a target model library and a database with meta data for all available sensor data.

There is also a situation analysis module which is more independently attached to the query processor. One of the purposes with the query processor is to feed the situation analysis module with relevant information. Below follows descriptions of the different parts of the system.

2.2 User interface

The basic idea of the information system is that the user should not need to have any knowledge about sensors or sensor data. The consequence is that the user interface should be based on concepts related to the user's task. In this case those concepts are area, object, and time, in particular the *area of interest* (AOI), the requested *objects* and the *time interval of interest* (IOI). The user interface for requesting a search of sensor data is split into three parts; one for area of interest, one for object selection, and one for specification of the time interval.

Generally, the user is not interested in data from all locations, so he/she must select the relevant area for the current query. To facilitate the selection of AOI the user interface contains a map, see Figure 2 (left). It is possible to zoom and pan the map to find the desired area of interest. Once the user has determined the correct location he/she simply marks it on the map. Although the user does not need to know anything about sensors, it is nevertheless undesirable to make a query

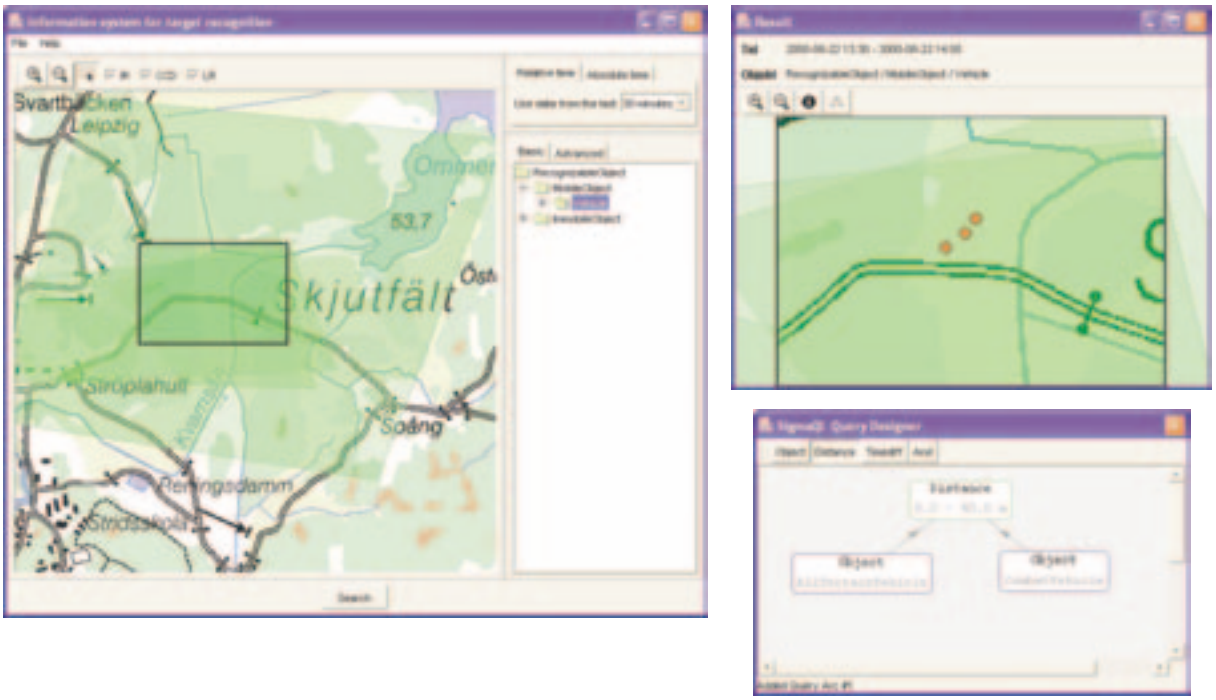


Figure 2: The user interface. Left: Creating of a query. Top right: The result of a query. Bottom right: The advanced query designer.
Map copyright Lantmäteriverket 2001, ref. no. L2002/308.

where no data are available. Consequently, the map is overlayed with sensor coverage information for the selected time interval.

The user also has to specify which kind of objects that are currently of interest. This can be done by selecting the objects from a hierarchical list in the right part of the user interface (Figure 2, left). The user can also use a more advanced tool for the object type selection called the *advanced query designer* (Figure 2, bottom right). In the advanced query designer it is possible to specify object attributes, e.g., length and velocity, but also relations, both spatial and temporal, between objects.

The third part of the user interface concerns the time-constraints on data for a particular search; that is basically the time span of the data. The user can either specify the absolute time for a query that should be repeated over time by giving start and end time, or he can specify the query in relation to *now*, for instance, by only using data collected during the last 30 minutes.

The result of the search is presented in a separate window showing the zoomed-in area of interest (Figure 2, top right). All occurrences of relevant objects recognized in the area by the sensors are marked. At the user's request, additional information from sensor data about each of the objects can be shown.

In most cases a query concerns the occurrences of objects of certain types including a number of attributes. Since the sensors and their platforms always are associated with various uncertainties there are always uncertainties associated with the query result. Clearly, there is no way of avoiding these uncertainties although the result can be improved, e.g. if the sensors are improved. For this reason, means to indicate the level of uncertainty in the query results must be available to allow the users to draw their conclusions regarding how trustworthy the query results are. Such means can, of course, be presented by the system in many ways. In this work we have chosen to represent the uncertainties by means of what are here called *belief values.* A belief value is a normalized value in the interval [0, 1]. The interpretation of such a value is that when it is close to 1 there is a high belief in the received objects and when close to zero the belief is low.

2.3 Knowledge system

Three requirements must be fulfilled to establish sensor data independence. Firstly, the system must be able to select one or more sensors while considering, e.g., the present weather and light conditions. Secondly, proper recognition algorithms must be chosen, i.e., algorithms that support an efficient recognition of the requested targets under the existing conditions. Finally, means to control the sensor data fusion process and to determine the interconnections between the controlling part and the fusion process must be established.

A system designed to support all these characteristics will need a structure that on the basis of the requested targets is able to pick the most appropriate sensor(s) and recognition algorithm(s) and to access, analyse and eventually fuse the information gathered from the sensors.

Selection of appropriate sensors and algorithms

The ontological knowledge-base has been designed to help answering such questions as which sensor data to use under certain circumstances. Also of importance is which recognition and cueing algorithm(s) that should be applied. An algorithm that performs these selections using the ontological knowledge-base has been developed, that is, the knowledge in the ontological knowledge-base is used in conjunction with the knowledge-base rules (described below) to determine which sensors and recognition/cueing algorithms are the most appropriate under the given circumstances, i.e., the actual ΣQL query, the meta data conditions, the external conditions, and the terrain background. This algorithm is called *Algorithm For Finding Appropriate Sensors and Algorithms (AFFAS)* and is described in detail in [4].

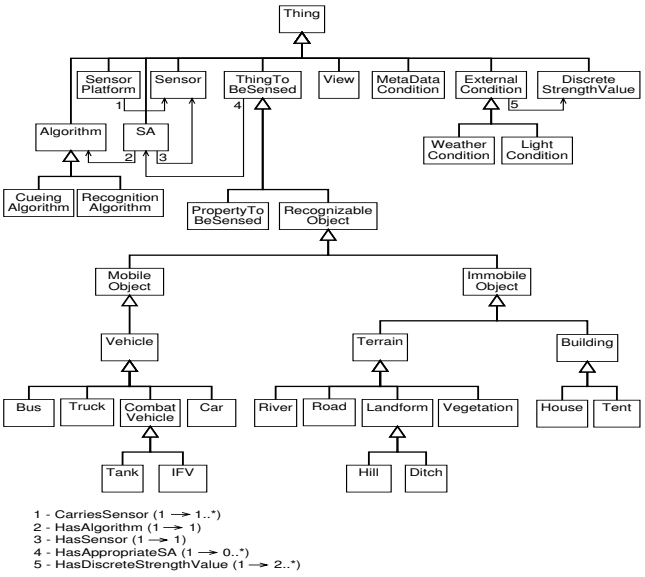


Figure 3: Ontology overview — The knowledge structure.

In the process of deciding upon appropriate sensors and algorithms it is necessary to have rules describing under which conditions certain sensors and algorithms are appropriate. The rules that are used to decide how the impact factors impact the sensors and recognition/cueing algorithms can be written in the following form:

*If an impact factor **x** has the discrete strength value **y** then the impact on the sensor/algorithm **z** has impact strength value **v**.*

Example 1: *If the impact factor **Rain** has the discrete strength value **Gentle** then the impact on recognition algorithm **GeometricFeatureExtraction** has impact strength value **Little**.*

Example 2: *If the impact factor **View** has the discrete strength value **Local** then the impact on the sensor **Standard CCD Sensor** has impact strength value **None**.*

A complete set of rules is needed for the system to function properly. Definitions of impact factor, discrete strength value, and impact strength value are presented in [4]. The definitions are quite straightforward.

2.4 Ontology

The knowledge represented in the ontological knowledgebase is modelled in a hierarchical manner known as the ontology. All concepts in the universe of discourse, the interesting properties of the concepts and the important relations between the concepts are modelled. The hierarchy has the ultimately general concept called *Thing* at the top. All other concepts inherit directly or indirectly from *Thing*. The hierarchy is organized so that more specialized concepts appear further down the inheritance chain. The concepts of this ontology are divided into three major parts, i.e., *things*, *characteristics* and *conditions*. An overview of the ontology is presented in Figure 3. Further details can be found in [4, 5].

The *Things to be Sensed and Recognized* part of the ontology models everything that can be sensed by the sensors and everything that can be recognized by the recognition algorithms or cued by the cueing algorithms. It is represented in the ontology by the *ThingToBeSensed* concept and subconcepts; examples are trucks and tanks.

The *Sensor and Algorithm Characteristics* part of the ontology models the characteristics of the sensors and the recognition and cueing algorithms. This part includes the concepts *Sensor platform, Sensor, Algorithm* (including subconcepts), and *Sensor-Algorithm* (the combination of a sensor and an algorithm).

The *Conditions* part of the ontology models the conditions that have an impact on the appropriateness of the sensors and the recognition/cueing algorithms. The conditions are state conditions describing the state of something, for example how rainy it is. The concepts that make up this part are *View*, *MetaDataCondition, ExternalCondition* (including subconcepts) and *DiscreteStrengthValue.*

Relations are used to model how the concepts in the ontology are related to each other. It is important to note that relations are inherited, meaning that if concept B inherits concept A and concept A has a relation to C, then concept B automatically has that relation to C as well.

2.5 Query interpretor

In the advanced query designer (Figure 2, bottom right), the user can enforce spatial and/or temporal constraints between objects in the query. An example of a query with a spatial and a temporal constraint is: “Show me tanks which are within 50 metres of each other at some point in time inside the area of interest during the time interval of interest”. The query interpretor evaluates which solutions are possible to a certain query given its constraints. In the example above, the system will first search for all tanks in the area of interest during the time interval of interest. After that, the query interpretor will be called to make sure that out of all the tanks the system found only those within 50 metres of another tank (at the same point in time) become part of the result. Thanks to the query interpretor a user can specify, e.g., a certain formation of vehicles instead of just the vehicles taking part in such a formation.

2.6 Target model library

In the process of target recognition, the sensor data are matched to models in the target model library. The models in the library are described by their 3D structure and their appearance. The use of the target model library is described in Section 4.

2.7 Sensor meta data DB

In a network centric warfare environment, where multiple sensors on multiple platforms provide sensor data in real time, it is very important to keep track of what kinds of sensor data are available at what locations at what times. For this purpose, we use a database of meta data from the sensor systems. This database is assumed to be correctly updated at all times, i.e. our system does not today contain the functionality to update the sensor meta data DB on the fly, instead we assume it is correctly updated. An important issue is of course to make sure to get sensor data from the interesting locations at the interesting points in time (i.e., sensor management). For now, our system is not involved in any sensor management activities, this is assumed to be handled elsewhere.

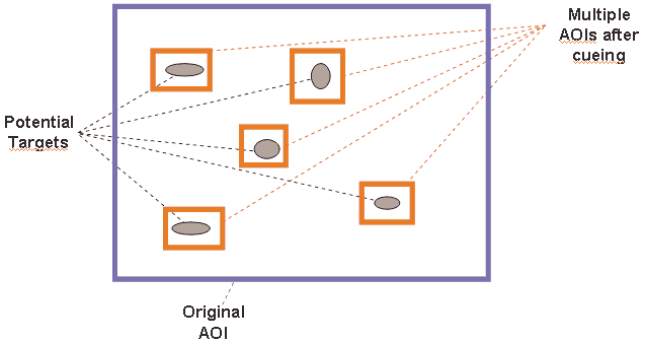


Figure 5: The cueing level — Identifying potential targets.

3 Query execution process

The execution of a query, that is, everything performed between the reception of a query entered by the user and the presentation of the query result, is performed in a process controlled by the ontological system; this includes the control of the data fusion process. An overview of the query execution process is presented in Figure 4. More details can be found in [5]. The two basic levels in the data fusion control process are described below.

The first level is the *cueing level*, see Figure 5, where the area of interest (AOI) can be large and the time interval of interest (IOI) can be long. Cueing in this sense means finding potential target objects (the ones searched for in the query) and indicating the positions of these potential targets. Note that the recognition algorithm can use other types of sensor data that can classify and/or identify the targets found in those positions. Sensor nodes for cueing are currently not integrated in the system.

The second level is the *recognition level* where the recognition process takes place. This is where recognition and possibly identification of the potential targets found in the cueing level is performed. The process includes two major steps: *estimation of the attributes* of potential targets and *matching* of the potential targets to models selected from a library. Recognition algorithms working in the first step are called attribute estimation algorithms, whereas algorithms working in the matching step are called matching algorithms. Recall that the system allows for multiple algorithms to perform both attribute estimation and model matching. Therefore, data fusion takes place both after attribute estimation and after model matching, see Figure 6.

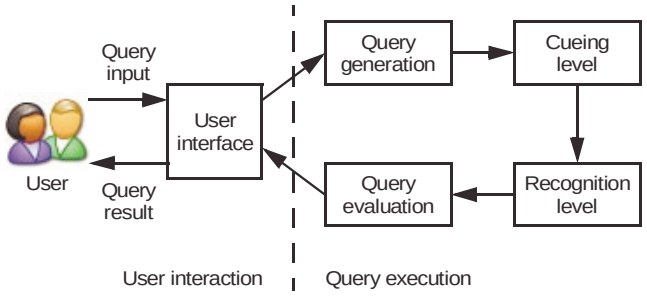


Figure 4: Overview of the query execution process.

When the recognition step has been carried out it is time to create an answer to the query. This is done by the query interpretor which evaluates the logical expressions enforced in the query by the user in the advanced query designer as described in Section 2.5.

4 Sensor data analysis

In this section we describe the target recognition process and the sensor data and algorithms used in that process. We will also show some of the information system's abilities to solve target recognition problems.

4.1 Sensors

The target recognition task is performed by analysing sensor data that is simultaneously recorded by multiple sensors. Currently, data from infrared (IR) cameras, CCD cameras (visual light), and laser radar sensors are used. Examples of data are shown in Figures 7 and 8. Two types of laser radar data sets are included. The first set consists of 3D scatter and reflectance data from an airborne down-looking scanning sensor (Figure 7 c–d). The second set consists of gated reflectance 2D images from a ground-based gated viewing system (GV), sequentially retrieved at different ranges to the target as illustrated in Figure 8. Such a data set provides for 3D reconstruction of the surface structure of a target, which is utilized in the analysis. The sensors are located on the same platform, with the exception of the GV sensor that is placed on a complementary platform. The sensor systems and the data collection in the visual, IR, and 3D/reflectance scatter laser radar are described in [17]. The GV system and the data collection are described [18].

4.2 The target recognition process

The target recognition process (i.e., the recognition level in the query execution process) is performed in four steps; *attribute estimation, attribute fusion* (i.e., fusion of attribute estimations), *model matching* and *model match fusion* (i.e., fusion of matching results), see Figure 6. The attribute estimation and model matching steps are described below, while the fusion is described in Section 5.

Several constraints are applied to the target recognition process in order to keep complexity down and also since training and testing data are very expensive. For example, the number of possible targets is limited to a small set and it is assumed that cueing (detection and coarse localization of the target) is handled separately.

As mentioned, the first step is to *estimate the target attributes* that should a) be used as input to the matching phase, and b) reduce the number of possibilities in the matching process. If there are only a few models and few variations of each model, the total number of matches can be kept small. Examples of target attributes are position (i.e., refined localization), orientation, dimensions, temperature, and colour. The estimated target attributes are returned to the knowledge system, which based on the attributes determines the possible target types and which model matching algorithms to invoke.

Figure 6: The recognition level - Classifying and possibly identifying potential targets.

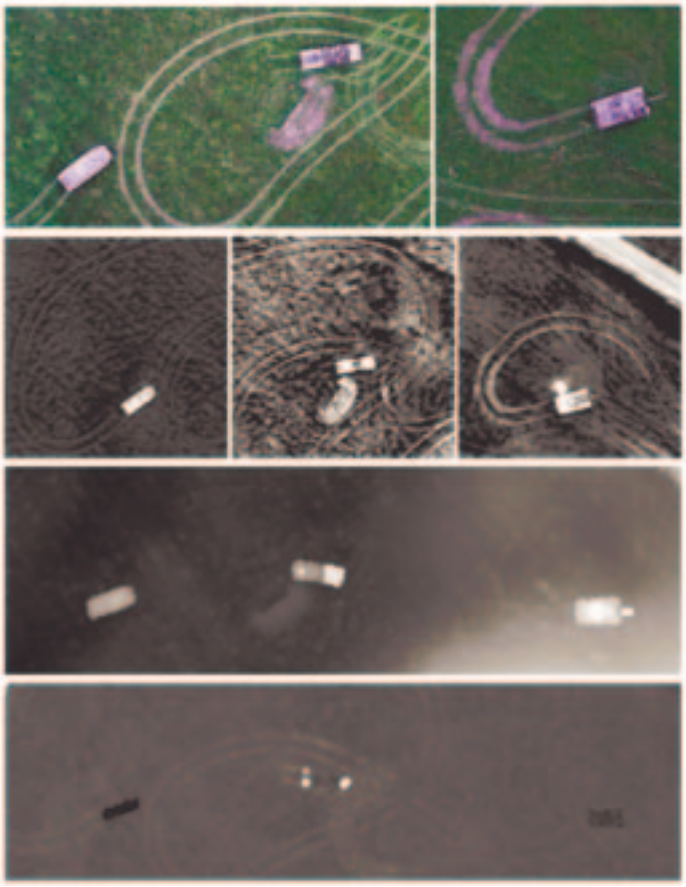


Figure 7: Registrations of an anti-tank gun vehicle BMP70 (left), truck TGB30 (middle) and a tank T72 (right). Top row: Visual senor data (CCD), second row: IR sensor data, third row: laser radar range data, bottom row: laser radar reflectance data.

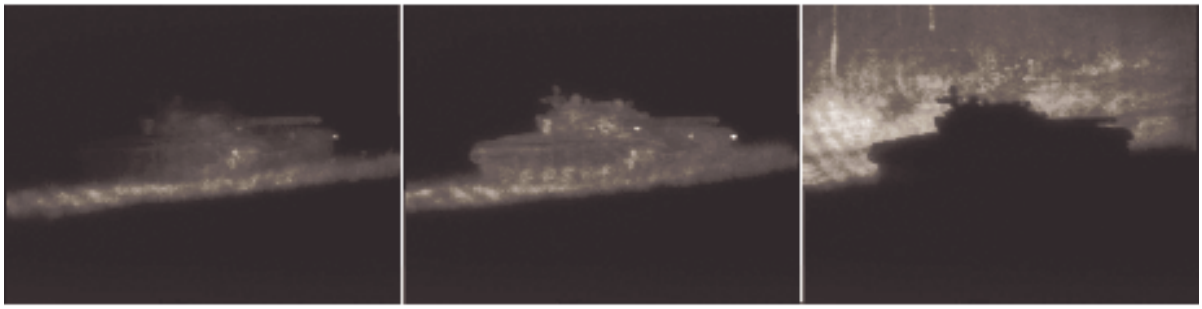


Figure 8: A sequence of gated 2D laser radar reflectance data recorded at varying distances. The target is a T72 main battle tank. Images are retrieved from in front of (left), on (middle) and behind (right) the target. From such sequence a 3D description of a target can be estimated.

The reliability of the attribute values are estimated by the algorithms themselves and can also, in the case when more than one algorithm has been invoked, be estimated by the knowledge system.

In the *model matching step* a common target model library is used, where each model is described by its 3D structure (facet/wireframe models), its appearance (visual or infrared textures), and, in some cases, algorithm/sensor specific attributes (e.g., pre-processed imagery). Based on the operator's query and the estimated attributes, a set of target models are selected for the matching process. Depending on the situation and how reliable the attribute estimations are, one, a few, or all of the available target models can be selected. For example, if the length of the target is estimated to eight metres, target models of small vehicles like cars can be excluded. However, if the length is an uncertain value, some of the smaller targets are included in the set of target models for the matching.

For each selected target model, one or more matching algorithms are invoked. The matching algorithm matches the sensor data to the corresponding target models and a belief value is calculated. This is a value between 0 and 1, where 1 means a perfect match. The belief values from the different algorithms and target models constitute the output from the sensor data analysis module, and they are handed back to the sensor fusion module, see Section 5, for a final decision.

Even though this division in attribute estimation and model matching might be suboptimal from a computer vision point of view (compared to combining the two steps), it has shown to be necessary when integrating the full system.

4.3 Sensor data analysis algorithms

The analysis methods need to be invariant to different sensing conditions, i.e., varying orientation and number of on-target samples. There is also a need to handle intra-class target variations, making the target differ from the pre-stored target model. The intra-variations are, for example, caused by variations in illumination, temperature, and minor shape variations. Due to the different sensor characteristics, differ-

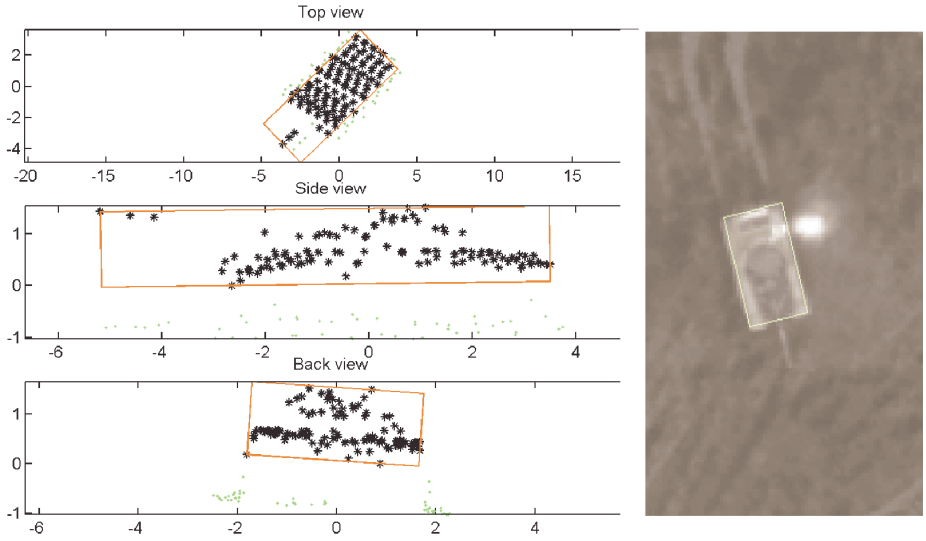


Figure 9: The attribute estimation methods. Estimations are performed on 3D laser radar and IR data of a T72 main battle tank. Left: Attribute estimation based on geometric feature extraction of 3D laser radar data. Each green dot is a background sample and the black stars are samples on the target. The rectangle shows the estimated 3D size and orientation. Axes in metres. Right: Attribute estimation based on active shape modelling, applied on IR data. The rectangle shows the estimated length, width and 2D orientation.

ent algorithms are needed for analysing the images. Moreover, several technical approaches are implemented to analyse the sensor data. There are seven such algorithms implemented, with varying technical approaches and capabilities of handling occlusions, intra-variations and movements. These algorithms are invoked by the knowledge system and can operate simultaneously or sequentially. Moreover, they can work on the same 2D or 3D data set or on different data sets depending on the situation at hand. The algorithms for attribute estimation and matching are of different types. Some are less complex, while others are more sophisticated, in order to handle specific cases like occlusion. In the attribute estimation step, the main criterion for algorithm selection is fast computation, while in the matching step more complex algorithms are also used.

Below, we will first describe the algorithms used for attribute estimation, then the algorithms for model matching in 2D data (images) and 3D data (laser radar range data) are described. The algorithms are very briefly described here.

Attribute estimation

The first problem is that the target is completely unknown, i.e., we know that *something* is there, and we should estimate its attributes. Naturally, we need to make relevant restrictions, for example we can look for man-made objects between three and twelve meters long.

When the input data is 3D range laser data, we can use *geometric feature extraction* [10,19]. Using this approach we approximate a 3D point scatter by one or several rectangles, and we can thus retrieve estimates of the length, width, height, and 3D orientation of the target.

In 2D images, *active shape models* (ASMs) [23] provide a technique for estimating contours around objects belonging to a specific class. Here, we use an ASM to find rectangular-shaped objects in IR images, 3D laser radar data, or laser reflectance data, the latter two resampled to 2D square pixel images. The output from the ASM is a quite precise estimate of the model parameters.

The assumption of rectangular targets holds well when searching for vehicles, provided that the images are in top view (the laser radar data can be resampled to any view).

Generative methods for 2D matching

To match one of the models from the target model library and the image data, and thus produce a belief value for the corresponding target type, the model needs to be adapted to the image in terms of 3D translation and rotation. For this purpose, generative models and methods are popular in the computer vision community, and we have studied two approaches here.

A generative model can, given a parameter set, generate an image that can be compared to the input image. If the images are similar, then the parameters are assumed to be a good estimate of the target parameters, and the similarity measure can be used as a belief value (after appropriate normalization).

Thus, we need search algorithms that minimize the difference between the generated model image and the input image by changing model parameters controlling the model's position, shape, and texture. Since such algorithms are typically greedy, they need a good initial suggestion of the model parameters.

The ontology tells us which model in the target model library to use for matching, and also the camera position relative to the target. Assuming approximately level ground, known target size (from the target model), and estimated attributes (position, orientation) we can initialize the parameters fairly well. Two search methods are implemented:

Using a sparse grid of Gabor probes. This approach is based on multiscale Gabor filters in a sparse grid. The filters represent edges and lines in different orientations and scales. The target model library has been analysed by the filter probes, and the outputs stored in an additional database. The node positions are iteratively refined as to minimize the difference between the filter outputs and the pre-computed filter outputs in the database.

Using active appearance models (AAMs) [22]. This approach has been tried out for adapting 3D models to 2D IR images. Six pose parameters and a turret rotation parameter are iteratively refined to minimize the difference between a cut-out window from the sensor data and the generated model image.

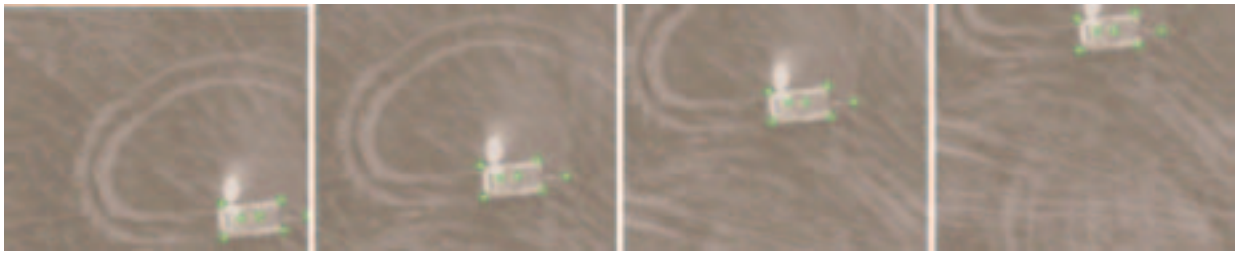


Figure 12: Matching and tracking using model-based reconstruction applied on IR data from Figure 7. The marked points on the object are tracked. In this example the sensor platform is moving and the target is still.

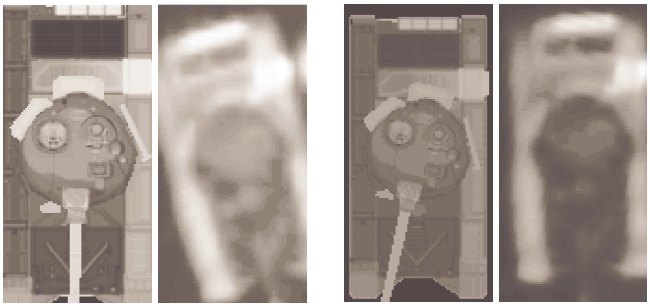


Figure 10: The AAM search. a) The original synthetic model. b) The original window. c) The final model. d) The final window.

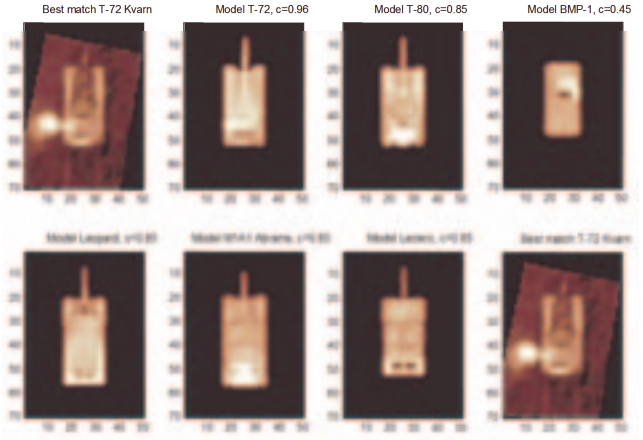


Figure 11: Matching with Gabor probe modelling for IR data in Figure 7. The parameter 'c' indicates the belief value for six different models.

In both cases, the final difference (summed squared error) between model and image is used as belief value and the model parameters are used for refined attribute estimation.

3D matching

Three different approaches for 3D model matching are implemented, suited for three different kinds of sensor data.

For infrared or visual image sequences, *model-based recognition* [20] is used for simultaneous tracking and recognition of targets. By tracking a few critical points of the target, e.g. corners and turret its movements and shape variations can be followed through the image sequence.

For 3D laser radar data, *3D scatter matching* [21] is used to match the sensor data to a 3D model of similar resolution. The squared distance between the points and the facets of the model is calculated as a belief value.

For GV laser radar data, we use a *3D range template matching* algorithm [21]. Based on the estimated attributes, a synthetic range image of the model is generated, from which surface and range boundaries are extracted. These are compared to the surface and boundary images extracted from the GV laser radar, resulting in a belief value.

4.4 Results and examples of system abilities

We describe some of the abilities of the system by three examples. By these examples we show the potential of multi-sensor data and its fusion, and we stress the advantage of combining different analysis methods. The first example forms a "standard" situation and incorporates all data analysis methods**.** The second and third examples show non-trivial analysis problems where no general solution exists today.

Example 1: A target in an open field

This example is considered rather straightforward and simple, as the target in this case is located on a flat, homogenous surface with no disturbing objects in the background. Moreover, the target is neither occluded nor camouflaged. The input data is illustrated in Figure 7. The attribute estimation is performed on 3D laser radar and IR data as shown in Figure 9. Figure 10 shows active appearance modelling (giving a belief value of 0.86 for the target being a T72 tank) and Figure 11 the shows result of model fitting with Gabor probes using six different models. In Figure 12, a result from the model-based reconstruction is shown, working on a sequence of IR images. Figure 14 (left) illustrates how the 3D laser radar data is fitted, using 3D scatter matching, to a low resolution CAD model, and 3D range template matching using GV data is illustrated in Figure 14 (right).

Example 2: Recognition of target variations

In Figure 15, articulation estimation in 3D laser radar data using the geometric feature extraction algorithm is shown.

In IR data it is possible to detect the engine exhaust plume and trails of a ground vehicle. These features are important indicators of target activity. Furthermore, the direction of the plume differs with target type and velocity. In Figure 13, the exhaust plume and vehicle trails from a real IR image of a T-72 is compared to the version generated by the target model library.

Example 3: Recognition of a partly occluded target.

The problem addressed in this last example is more difficult. However, for a ground target recognition system it is necessary to be able to handle (partly) occluded and/or camouflaged targets. To solve the problem, the analysis methods must be capable to perform their task even if parts of the target is not registered. The ability to penetrate sparse objects, such as some categories of vegetation and camouflage, is a key issue to overcome the problem.

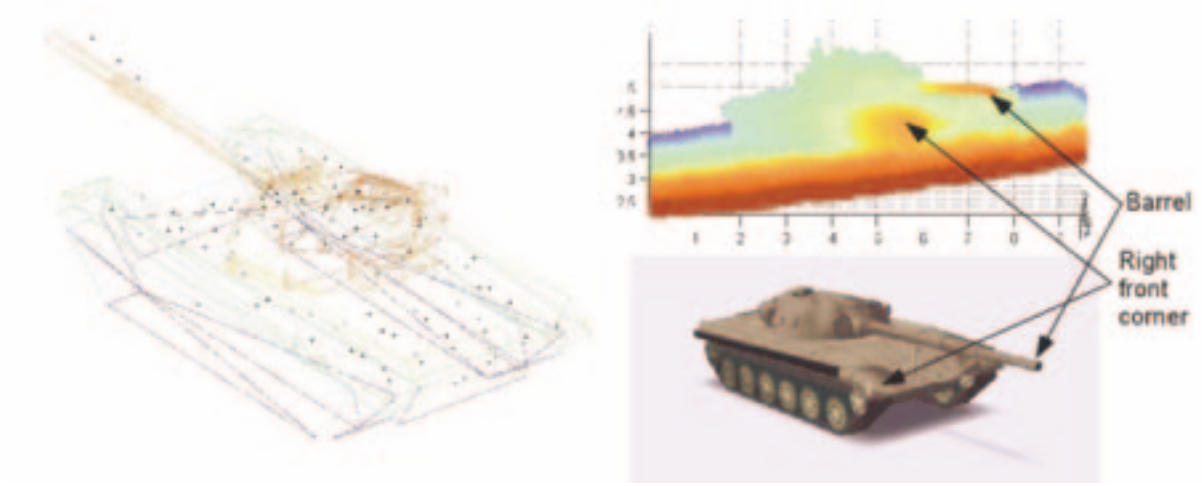


Figure 14: Matching using the target's 3D structure (extracted from laser radar range data). Left: Matching using the 3D scatter matching method. The samples on the target and the facets of the model are shown. The matching is performed on laser radar range data of the T72 shown in Figure 7. Right: Matching using the 3D range template matching method. Top right: Processed sensor data. Bottom right: The model transformed to the same articulation. Matching is performed on data of the T72 of the same type as shown in Figure 8.

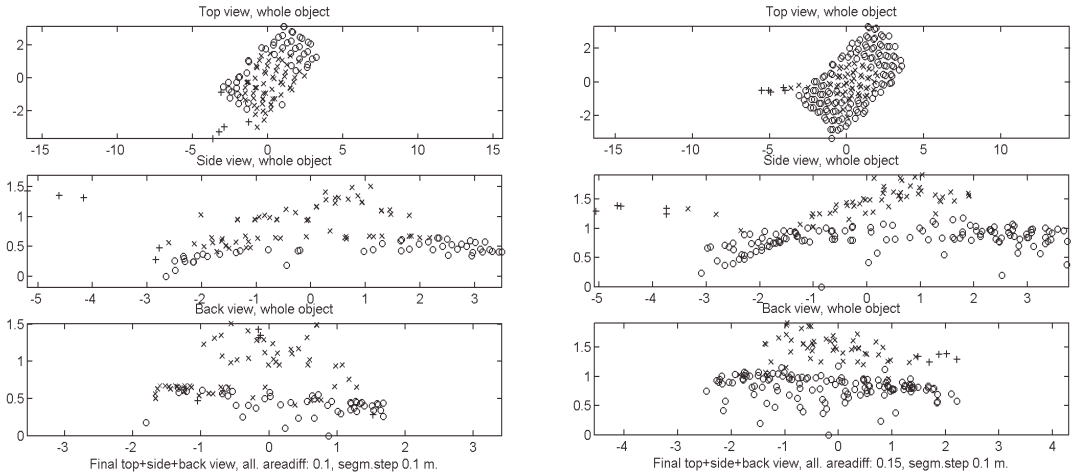


Figure 15: Estimation of barrel articulation and turret extraction of a T72 using the geometric feature extraction algorithm. Note the different positions of the barrel. The circles, x-marks and plus-signs indicate different parts of the object, approximately the barrel, turret and chassis. Axes in metres.

Figure 13: Models for exhaust plume and vehicle tracks are included in the IR-model library. Left: IR image of a T-72. Right: IR model of the T-72 with activated models for exhaust plume and vehicle trails.

The technical approach used in here is to fuse IR and 3D laser radar data. The 3D laser radar data analysis supports the IR data analysis, by penetrating the vegetation and separating objects at different heights, for example a tree and a vehicle. The IR matching methods can then weight the image and focus the analysis on the image regions that have been classified as belonging to a target pixels.

5 Data fusion

There are two fusion processes in the query system. The first one considers the *attribute set estimations* (ASE) which is the output from all the attribute estimation algorithms. The second one considers the matching output from the matching algorithms, i.e. the matching results, which each consists of a refined object state estimation and a belief value. The object's state is described by those attributes in the ASE that are sent to the attribute estimation and matching algorithms[1]. A target's state easily changes over time; examples are *orientation* and *speed.*

5.1 Fusion of attribute estimations

The first step of the ASE fusion consists of identifying sets of similar ASEs. Each set, or cluster, is then replaced with one single, representative ASE, i.e. the fusion result. The aim is to provide the matching process with a condensed number of distinct ASEs. These can then be kept apart in the match-

1. The other attributes of the ASE refer to the object's *properties*. Those values are never sent to the model matching algorithms. Instead, those algorithms are supplied with models consistent with the estimated properties.

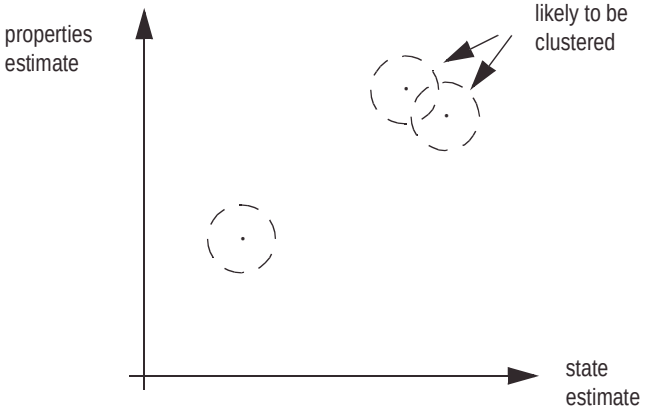


Figure 16: Clustering of similar ASEs. In this example the ASEs are complete and thus represented by dots (with estimated error bounds). In general, ASEs are incomplete and represented by objects of higher dimension. Also, both axes are in reality multidimensional.

ing process, so that the second fusion process can make comparisons between belief values regarding objects in the same state[1]. Since a typical ASE is likely to be incomplete, the problem is similar to a general set of problems called *clustering of incomplete data*. Such problems have been addressed for example in [14]. Here, an Euclidean distance between ASEs has been used as a basis for clustering. Roughly, it states that if the error volumes of a set of ASEs are pairwise non-disjunct, the set forms a cluster.

Figure 16 visualises the first step of the attribute fusion process. Note that the ASEs here are represented by dots (with estimated error circles), which corresponds to *complete* ASEs. Generally, however, the ASEs are *incomplete*. The idealisation has been done here since both the state and the properties really are multidimensional, which complicates the drawing of an incomplete ASE.

Selection of a representative ASE for each cluster is the second step of the first fusion process. Here, the ASE is chosen and should be as complete as possible. The state, especially, needs to be well-defined. Otherwise, the fusion of matching results cannot make the comparisons mentioned above. Two main approaches to obtain a well-defined state estimate have been identified. Which approach will be followed is not yet decided but work to solve this problem is currently continuing. However, the two approaches are:

1. *Fusing* state estimations, i.e. combining values from incomplete estimations in order to form a complete one. This is non-trivial, since the resulting state might be unattainable from different aspects.

2. Supplying the matching algorithms with incomplete estimates, using *them* as input to the attribute estimation algorithms and then rerun the processes. The states from the matching algorithms should be well-defined. The belief values produced in the preliminary run can be fused with a "safe" fusion method. Its result can be taken as a preliminary result from the system.

5.2 Fusion of matching results

The second fusion step deals with the results from the different matching algorithms. Here the belief value is in focus.

1. It is assumed that two matching algorithms using similar state estimates, refine these to essentially the same state estimate. This is why clusters are identified in the attribute estimation fusion.

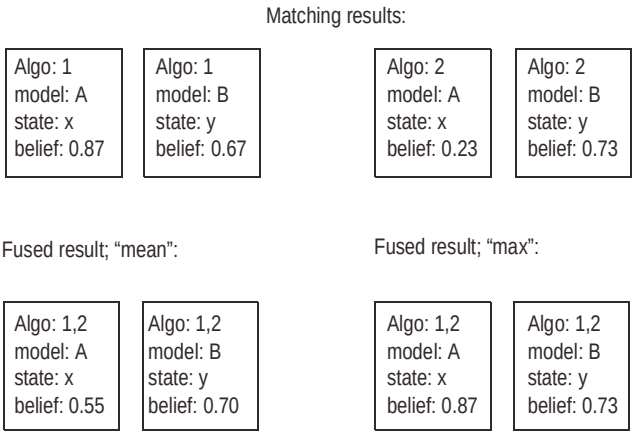


Figure 18: Illustration of the second fusion step, in this case with only two hypotheses and two matching algorithms. Considering mean and maximum belief values for the two hypotheses, respectively, leads to radically different results.

Several simple approaches have been implemented, corresponding to different strategies respectively. For example, one could argue that a hypothesis (model + state) should be heavily supported by *all* matching algorithms to be considered heavily supported. Also, one could argue that high enough support from *one* matching algorithm is enough. Figure 18 exemplifies such a situation. Other desired properties of the fused result call for other strategies as well.

6 Terrain analysis

Most research in 3D terrain modelling is focused on obtaining maximum accuracy and fidelity of the model, while not being constrained with particularly difficult time requirements. In the military domain the purpose is often to perform mission training or to simulate sensors with a very high level of accuracy. In contrast to that, the terrain analysis considered in this project should be viewed as a part of a decision support system. Such a system must be able to answer que-

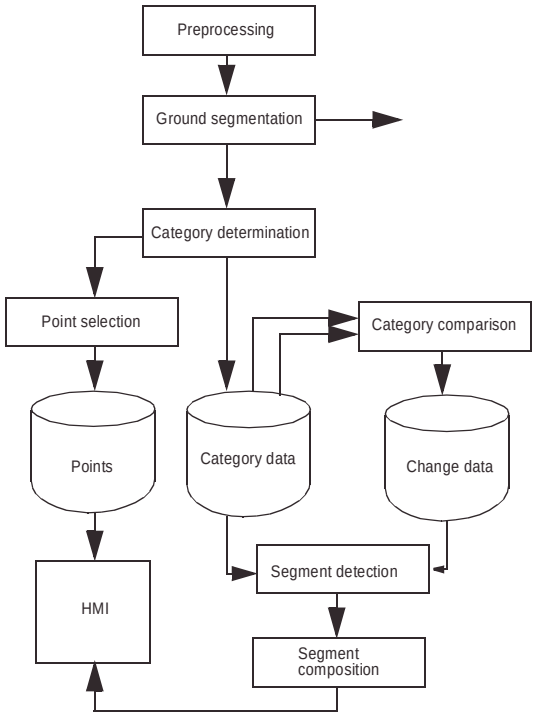


Figure 17: An overview of the most important processing steps and data sets in the terrain analysis system.

ries from the users about the terrain properties, relevant to their goals, with a speed and accuracy that is required by the situation and the mission. Consequently, the designed terrain analysis system must be adaptive and able to respond to queries in close to real time. The system should also automatically permit translation of the queries into an appropriate format. Another important demand is that the system must be able to support multi-step processing, where initial detection and coarse description of the feature of interest is done first and where refined, special purpose methods can be used afterwards to obtain the needed accuracy. To support this demand, methods that reduce the data volumes have been developed. Reduction of the data volumes is also necessary to reduce the search times and to make further processing more efficient. The approach in this work has been to use a symbolic method, as no quantitative algorithm can answer the queries quickly enough. The main result is concerned with detection of important terrain features and can be used for, e.g., construction of driveability maps. A recent overview of this part of the work can be found in [12].

During the last year, most effort has been put into designing and implementing algorithms for detecting changes in the terrain surface, derived from laser radar data collected at different times. Change detection is an important component in a terrain analysis system for two reasons. Firstly, the ground surface changes very slowly over time and to store and access large unchanged quantities of data is clearly detrimental to the system functionality. Considering only the changed parts from a newly acquired data set can speed up processing considerably, which is a necessity in a decision support system. Secondly, the changes that have occurred are potentially very important as they may be a sign of conscious alteration of terrain. An overview of the terrain processing system can be found in Figure 17.

7 Situation awareness

The work in this area has been oriented towards framework design and identification of required algorithmic support. A suggestion for a basic framework for situation analysis (SA) support has been presented in [11]. Although a very general framework, it is based on some assumptions of what the characteristic problems are when looking at sensor observations spread over time and space in a dynamic tactical land-based context. The basic underlying assumption is that of observation *fragmentation* in time and space. This makes association between observations a fundamental and difficult problem, one which cannot be solved by physical laws and statistics alone. To deal with such situations we have to make use of a priori knowledge of typical organization and behaviour of military vehicles and units. Straightforward use of such knowledge in a computer optimization model, to find the most likely interpretation of a given data set, does however raise several questions. Two main concerns are those of a priori knowledge reliability and context dependency. If we ignore these issues, the result will most probably be an unreliable system with a high degree of user distrust.

The conclusion we draw from the reasoning above is that the user must be involved in the interpretation process. The SA system then becomes a tool for user guided exploration of possible/likely interpretations. An optimization model is still at the core of the SA system, but it is now required to be able to find *alternative solutions.* The concept of alternative solutions can be defined as a set of solutions which are sufficiently likely and sufficiently dissimilar from each other to be of interest. The estimation of solution likelihood should be built into the optimization model, but the concept of solution similarity is context dependent and can only be defined by the user. The overall goal is to give the user an overview of interpretation possibilities by presenting a manageable number of solutions where the differences are relevant. This should be done in an iterative way, where the user can become more precise as to what information he is interested in as the optimization process proceeds. This can be viewed as a learning process.

To meet the requirements of a flexible optimization process capable of generating multiple solutions, the paradigm of evolutionary computation (EC) has been investigated [13]. This paradigm has some inherent properties that suit the demands of our SA framework. It has for example been extensively used in the area of multi-objective optimization, where multiple solutions are searched for in parallel. In order to develop an EC-based optimization kernel of our framework for SA, there are two major methodological/algorithmic issues that have been addressed.

The first issue is a *selection of a representation and of operators* for EC for the combinatorial problem of set partitioning. There are some interesting suggestions in the literature. A common property of all these pure EC variants, however, is low computational efficiency. This is primarily due to the strategy in pure EC based systems of blind randomization with feedback, which is very general and flexible but often inferior to heuristic based systems. To remedy this we have adopted the common strategy of hybridization, where heuristic components are integrated into the EC machinery to guide the search process. In this case we use the local object-to-object association values for search guidance. With this approach we can quickly find a good solution in most situations and then continue searching for better or different solutions as time permits. However, the machinery developed still needs to be tested in realistic and dynamic scenarios.

The second major issue is that of *generating multiple solutions* which are “dissimilar”. There are quite simple ways to achieve this but the principal problem is that of low computational efficiency. Some efforts in this area are reported in the literature, but none of which have the generality needed in our framework. This is a crucial part of our suggested SA framework which needs further research.

8 Implementation

A prototype of the information system and its query language has been implemented. The prototype basically includes the following:

- A ΣQL query processor;
- The query processor includes a knowledge-based system which is connected to an ontology; designed to allow sensor data independence from an end-user perspective;
- A data fusion module that fuses sensor data in two steps, i.e to support fusion of attribute estimations as well as fusion of matching results;
- A visual user interface that allows the application of queries in a sensor data independent way;

- Means for visual presentation of query results;
- Application of sensor data from sensors of different types, currently IR, laser radar (scanning and gated viewing) and CCD-camera—the number and types of sensors are extendable;
- Seven algorithms for analysis of the sensor data.

By means of the prototype, queries can be applied and answered. Basically, these queries are concerned with recognition of targets of military type. Some recognition algorithms include initial support for recognition of partly occluded targets.

The terrain data analysis activities show that the development of the high resolution digital terrain model allow for:

- various terrain features to be efficiently identified by means of a filtering technique;
- the terrain model to be used for efficient visualization of the terrain in high resolution.

The activities to finally bring forward a system for situation awareness have not yet come to a break-through although some fundamental results have been demonstrated—the reasons for this are that this part of the project has been going on at a low pace and that the system basically has been built bottom-up, i.e. starting at the sensor data.

9 Conclusions

In the system presented in this paper a user can define queries in a visual user interface using well-known terminology such as area of interest, time interval of interest, and interesting object types. If necessary, spatial and/or temporal constraints between objects in a query can be enforced using the advanced query designer. The system will then automatically come up with answers to the query without user interference in choice of sensors, sensor data analysis, data fusion, etc, thus providing sensor data independence for the end-user of the information system.

The query language is characterized by the sensor data independence capability, which makes it possible even for users that lack technical knowledge on sensors and sensor data to use the system. To support the users even further, the system includes, in addition to sensor data analysis algorithms performing target recognition, means for sensor data fusion. It can deliver answers to the queries so that the uncertainties introduced by the sensors are taken into account by the resulting belief values.

Furthermore, we can expect that information systems in the future can gain access to enormous amounts of sensor data in a network centric warfare context. In this situation, it will be impossible for an information system user to access and analyse the available data on his own. A system which is sensor data independent, such as the one presented here, helps the user with this task, so that he can be focused on his primary tasks.

Thus we believe that ΣQL is a powerful tool that can be used as a decision support tool in a network centric command and control system. We also believe that even inexperienced users can use the system and feel confidence in the results it produces.

References

[1] Hall, D. L. and Llinas J. (Eds.), *Handbook of Multisensor Data Fusion*, CRC Press, New York, 2001.

[2] Nielsen, J., *Usability Engineering*, Morgan Kaufman, New York, 2001.

[3] Jungert, E., Silvervarg, K. and Horney, T., "Ontology driven sensor independence in a query supported C^2-system", *Proceedings of the NATO workshop on Massive Military Data Fusion and Visualization: Users Talk with Developers*, Halden, Norway, September 2002.

[4] Horney, T., *Design of an ontological knowledge structure for a query language for multiple data sources*, FOI-R--0498--SE, Swedish Defence Research Agency (FOI), May 2002.

[5] Horney, T., Jungert, E., Folkesson, M., "An Ontology Controlled Data Fusion Process for Query Language", *Proceedings of the International Conference on Information Fusion 2003 (Fusion'03)*, Cairns, Australia, July 8–11, 2003.

[6] Matheus, C. J., Kokar, M., Baclawski, K., "A Core Ontology for Situation Awareness", *Proceedings of the International Conference on Information Fusion 2003 (Fusion'03)*, Cairns, Australia, July 8–11, 2003.

[7] Silvervarg, K. and Jungert, E., "Aspects of a visual user interface for spatial/temporal queries", *Proceedings of the workshop of Visual Language and Computing*, pp 287–293, Miami, Florida, September 24–26, 2003.

[8] Chang, S.-K., Costagliola, G., Jungert, E. and Orciuoli, F., "Querying Distributed Multimedia Databases and Data Sources for Sensor Data Fusion", accepted for publication in *IEEE Transactions on Multimedia*, 2003.

[9] Chang, S.-K. and Jungert, E., "Query Languages for Multimedia Search", in Lew, M. S. (Ed.), *Principles of Visual Information Retrieval*, pp 199–217, Springer Verlag, Berlin, 2001.

[10] Ahlberg, J., Klasén, L., Grönwall, C., Ulvklo, M., Jungert, E., "Automatic Target Recognition on a Multi-Sensor Platform", *Proceedings of the Swedish Symposium on Image Analysis*, pp 93–96, Stockholm, Sweden, March 6–7, 2003.

[11] Fransson, J., Jungert, E., "Towards a Query Assisted Tool for Situation Assessment", *Proceedings of the International Conference on information Fusion 2002 (Fusion'02)*, Annapolis, Maryland, July 8–11.

[12] Lantz, F., Jungert, E., "Determination of Terrain Features in a Terrain Model from Laser Radar Data", *Proceedings of the Workshop on 3D reconstruction from Airborne Laser scanner an in SAR Data*, pp 193–198, Dreseden, Germany, October 8–10, 2003.

[13] Coello, C. A. C., "A comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques", *International Journal of Knowledge and Information Systems*, 1(3):269–308, August 1999.

[14] Hathaway, R. J., and Bezdek, J. C., "Fuzzy c-means clustering of incomplete data", *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 31(5), October 2001.

[15] Ullman, J. D., *Database and knowledge-base systems*, Vol. 1, pp 11–12, Computer Science Press, Rockville, Maryland, USA, 1988.

[16] Grönwall, C., *Mätningar med flygburet multisensorsystem - Mätrapport från fordonsplatserna i Kvarn och Tullbron*, FOI-D--0060--SE, Swedish Defence Research Agency (FOI), August 2002, (in Swedish).

[17] Steinvall, O., Klasén, L., Carlsson, T., Andersson, P., Larsson, H., Elmquist, M., Henriksson, M., *Grindad avbildning - fördjupad studie*, FOI-R--0991--SE, Swedish Defence Research Agency (FOI), November 2003, (in Swedish).

[18] Carlsson, C., *Vehicle Size and Orientation Estimation using Geometric Fitting*, Thesis no. 840, 2000, LiU-TEK-LIC-2000:36, Dept. of Electrical Engineering, Linköping University, Linköping, Sweden, 2000.

[19] Steinvall O., Carlsson T., Grönwall C., Larsson H., Andersson P., Klasén L., *Laser based 3-D imaging new capabilities for optical sensing*, FOI-R--0856-SE, Swedish Defence Research Agency (FOI), 2003.

[20] Klasén, L., *Image Sequence analysis of Complex Objects – Law Enforcement and Defence Applications*, Linköping Studies in Science and Technology, Dissertation No. 762, 2002, Dept. of Electrical Engineering, Linköping University, Linköping, Sweden, 2002.

[21] Andersson P., *Automatic target recognition from laser radar data. Applications to gated viewing and airborne 3D laser radar*, FOI-R--0829--SE, Swedish Defence Research Agency (FOI), 2003.

[22] Cootes, T. F., Edwards, G. J., Taylor, C. J., "Active Appearance Models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–684, 2001.

[23] Cootes, T. F., Edwards, G. J., Cooper, D. H., Graham, J., "Active Shape Models – 'smart snakes'", *Proceedings of British Machine Vision Conference*, pp 266–275, Leeds, UK, 1992.

Appendix B

Querying Distributed Multimedia Databases and Data Sources for Sensor Data Fusion

IEEE Transactions on Multimedia, Vol. 06, No 5, Oct. 2004.

Chang, S.-K., Costagliola, G., Jungert, E and F. Orciuoli

Querying Distributed Multimedia Databases and Data Sources for Sensor Data Fusion

Shi-Kuo Chang, *Fellow, IEEE*, Gennaro Costagliola, *Member, IEEE*, Erland Jungert, and Francesco Orciuoli

***Abstract*—Sensor data fusion imposes a number of novel requirements on query languages and query processing techniques. A spatial/temporal query language called ΣQL has been proposed to support the retrieval and fusion of multimedia information from multiple sources and databases. In this paper we investigate fusion techniques, multimedia data transformations and ΣQL query processing techniques for sensor data fusion. Fusion techniques including fusion by the merge operation, the detection of moving objects, and the incorporation of belief values, have been developed. An experimental prototype has been implemented and tested to demonstrate the feasibility of these techniques.**

***Index Terms*—Data fusion, distributed database, multimedia database, query language, query processing, sensor data fusion.**

I. Introduction

SENSOR data fusion is an area of increasing importance that requires novel query languages and query processing techniques for the handling of spatial/temporal information. Sensors behave quite differently from traditional database sources. Most sensors are designed to generate information in a temporal sequence. Sensors such as video camera and laser radar also generate large quantities of spatial information. Therefore the query language and query processing techniques must be able to handle sources that can produce large quantities of streaming data within short periods of time.

Another aspect to consider is that user's queries may be modified to include data from more than one sensor and therefore require the fusion of multiple sensor information. In our empirical study we collected information from different type of sensors, including laser radar, infrared video (similar to video but generated at 60 frames/s) and CCD digital camera. In a preliminary analysis of the above described sensor data, it is found that data from a single sensor yields poor results in object recognition. For instance, the target object may be partially hidden by an occluding object such as a tree, rendering certain type of sensor ineffective. Object recognition can be significantly improved if the query is modified to obtain information from another type of sensor, while allowing the target being partially hidden. In other words, one (or more) sensor may serve as a guide to the other sensors by providing status information such as position, time and accuracy, which can be incorporated in multiple views and formulated as constraints in the refined query.

Existing query processing techniques are not designed to handle sensors that produce large quantities of streaming data within short periods of time. With existing query languages such as SQL, it is also difficult to systematically refine the query to deal with information fusion from multiple sensors and distributed databases. To support the retrieval and fusion of multimedia information from multiple sources and distributed databases, a spatial/temporal query language called ΣQL has been proposed [6]. ΣQL is based upon the σ-operator sequence and in practice expressible in a syntax similar to SQL. ΣQL allows a user to specify powerful spatial/temporal queries for both multimedia data sources and multimedia databases, eliminating the need to write separate queries for each. A ΣQL query can be processed in the most effective manner by first selecting the suitable transformations of multimedia data to derive the multimedia static schema, and then processing the query with respect to the selected multimedia static schema.

The main contribution of this paper is to provide a systematic approach consisting of fusion techniques, multimedia data transformations and query processing techniques for sensor data fusion. The paper is organized as follows. Section II presents background and related research. The basic concept of the dual representation of the σ-query is explained in Section III. The usage of the various types of operators is discussed in Section IV. The techniques of sensor data fusion are explained in Section V. Section VI describes the architecture of the system for sensor data fusion. Section VII presents the data model, and Section VIII the detailed syntax of ΣQL and some examples. In Sections IX–XI, we discuss sensor data fusion using the merge operation, the detection of moving objects in a video, and the incorporation of belief values, respectively. Section XII concludes the paper.

II. Background and Related Research

Sensor data fusion posed some special problems. First of all, there is no general solution to the problem of sensor data fusion for an unlimited number of different types of sensors. The problem is usually restricted to a limited number of object types observed from a specific perspective by a limited number of sensors [22]. One such example is to select sensors that are looking only at ground objects, primarily vehicles, from a top view perspective where the sensors are carried by a flying platform such as a helicoptor. By studying this restricted problem in detail, we may be able to understand better how to deal with complex

Manuscript received February 12, 2002; revised December 10, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sankar Basu.

S.-K. Chang is with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, 15260 USA (e-mail: chang@cs.pitt.edu).

G. Costagliola is with the Dipartimento di Matematica ed Informatica, Università di Salerno, Salerno, Italy (e-mail: gencos@unisa.it).

E. Jungert is with the Swedish Defense Research Institute (FOA), SE-172 90 Linköping, Sweden (e-mail: jungert@foi.se).

F. Orciuoli is with the Dipartimento di Ingegneria dell'Informazione e Matematica Applicata, Università di Salerno, Salerno, Italy (e-mail: orciuoli@crmpa.unisa.it).

Digital Object Identifier 10.1109/TMM.2004.834862

queries for sensor data fusion. For a more general view on sensor data fusion, see e.g., Waltz and Llinas [21].

As explained in the preceding section, sensor data fusion requires a query language that supports sensor sources and the systematic modification of queries. In early research on query modification, queries are modified to deal with integrity constraints [19]. In query augmentation, queries are augmented by adding constraints to speed up query processing [8]. In query refinement [20], multiple term queries are refined by dynamically combining precomputed suggestions for single term queries. Recently query refinement technique was applied to content-based retrieval from multimedia databases [3]. In our approach, the refined queries are manually created to deal with the lack of information from a certain source or sources, and therefore not only the constraints can be changed, but also the source(s). This approach has not been considered previously in database query processing because usually the sources are assumed to provide the complete information needed by the queries.

In addition to the related approaches in query modification, there is also recent research work in agent-based techniques that are relevant to our approach. Many mobile agent systems have been developed [1], [2], [16], and recently mobile agent technology is beginning to be applied to information retrieval from multimedia databases [15]. It is conceivable that sensors can be handled by different agents that exchange information and cooperate with each other to achieve fusion. However mobile agents are highly domain-specific and depend on ad-hoc, 'hardwired' programs to implement them. In contrast our approach offers a framework for data transformation and query processing and is applicable to different type of sensors, thus achieving a certain degree of sensor data independence.

III. The Dual Representation of the ΣQL Query Language

As noted in Section I, ΣQL is a spatial/temporal query language for information retrieval from multiple heterogeneous sources and databases. Unlike SQL, which does not explicitly deal with spatial/temporal queries, ΣQL is designed with that purpose in mind. Unlike SQL, which deals only with databases, ΣQL is designed to deal with both static sources (databases) and dynamic sources (sensors), and furthermore these sources are distributed. Its strength is its simplicity: the query language is based upon a single operator—the σ-operator. Yet the concept is natural and can easily be mapped into an SQL-like query language. The σ-query is useful in theoretical investigation, while the SQL-like query language is easy to implement and is a step toward a user-friendly visual query language. An example is illustrated in Fig. 1. The *source* R, also called a *universe*, consists of time slices where each time slice consists of objects with the same time value. To extract three predetermined time slices from the source R, the σ-query in mathematical notation is $\sigma_{\mathrm{t}}\ (\mathrm{t}_1, \mathrm{t}_2, \mathrm{t}_3)\ \mathrm{R}$.

The meaning of the σ-operator in the above query is "select," i.e., we want to select the time dimension and three slices along this dimension. The subscript t in σ_{t} indicates the selection of the time dimension. In the SQL-like language the ΣQL query is expressed as

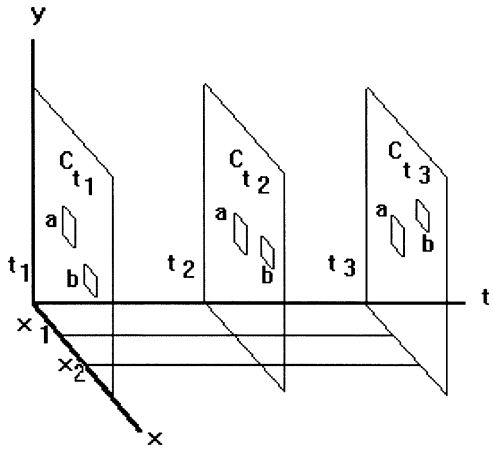


Fig. 1. Example of extracting three time slices from a source.

```
SELECT t
CLUSTER t1, t2, t3
FROM R
```

A new keyword "CLUSTER" is introduced so that the parameters such as t_1, t_2, t_3 for the σ-operator can be listed. The word "CLUSTER" indicates that objects belonging to the same subset of the universe (i.e.,, a cluster) must share some common characteristics (such as having the same time value, being similar to one another, etc.) Clustering is a technique used in pattern classification to form subsets of similar objects. A cluster may have a substructure specified in another (recursive) query. Clustering is a natural concept when dealing with spatial/temporal objects that are specifiable only through similarity to other objects. The mechanism for clustering will be discussed further in Section VIII. The result of a ΣQL query is a string that describes the relationships among the clusters. This string is called a *cluster-string*, which will also be discussed further in Section IV.

The dual representation of ΣQL means that a query can be formulated as an SQL-like query [6] or as a sequence of *generic* operators (the σ-operators introduced above) and *specialized* operators (the ϕ-operators to be discussed in the following section). Translation from one representation to the other is quite straightforward.

The operators may handle both qualitative and quantitative information. Primarily, the operators allow operations on a sensor-data-independent level, i.e., sensor data should be transformed into information structures at high abstraction levels that are sensor independent. To accomplish this, the queries are expressed in terms of operator sequences where the transformations of the sensor data are carried out stepwise by the operators. The operators reduce the dimensions of the multudimensional search space to which each successive operator is applied. Intuitively, the reduced search space is also another *cluster*. Thus, as successive operators are applied, the clusters become more and more refined.

In contrast to this refinement of the search space, the *fusion* and related operators take multiple clusters as input and fuse the information to determine a belief value that may support a certain hypothesis such as whether different observations in the

clusters correspond to the same object. Furthermore the fusion and related operators can handle uncertain information through the use of belief values.

IV. Operator Classes

The operators in ΣQL can be categorized with respect to their functionality. The two main classes are the *transformational operators* (the σ-operators) and the *fusion operators* (the ϕ-operators). In this section the two main operator classes are discussed according to their input, output and functionality.

A. σ-Operators

A σ-operator is defined as an operator to be applied to any multudimensional source of objects in a specified set of intervals along a dimension. The operator projects the source along that dimension to extract clusters [6]. Each cluster contains a set of objects or components whose projected positions fall in one of the given intervals along that dimension. As an example, let us write a σ-expression for extracting the video frame sequences in the time intervals $[t_1 - t_2]$ and $[t_3 - t_4]$ from a video source VideoR. The expression is $\sigma_{\text{time}}([t_1 - t_2], [t_3 - t_4])$ VideoR where VideoR is projected along the time dimension to extract clusters (frames in this case) whose projected positions along the time dimenson are in the specified intervals.

In case of uncertainty, the components of the clusters may be associated with various probabilities or belief values. Input and output data may be of either qualitative or quantitative type, although generally the later type is of main interest. Thus input data will be accessed from either a raw-data source such as a sensor, or from a structured data source such as a database, or from some internal source such as *qualitative strings* that are strings consisting of object descriptions projected along certain dimension(s) [6]. The output data correspond to clusters in relational representations that in practice may be available as qualitative strings of various types [6]. The general formalism can thus be expressed in the following way:

$$\sigma_{\text{dimension}}\left(\langle\text{intervals}\rangle\right)\{\langle\text{source}\rangle|\langle\text{cluster}\rangle|\langle\text{cluster\_strings}\rangle|\ldots\} \Rightarrow \{\langle\text{cluster}\rangle|\langle\text{cluster\_strings}\rangle|\ldots\}.$$

A variety of σ-operators can be defined [11]. Many of these operators are common in most spatial applications. Examples are the determination of various attributes and spatial relations, such as "northwest-of," "to the left of," etc. For simple inputs, these operators can be described as

$$\sigma_{\text{attribute}}\left(\langle\text{attribute} - \text{values}\rangle\right)\langle\text{cluster\_strings}\rangle \Rightarrow \langle\text{relational\_strings\_of\_}(\langle\text{attribute}\rangle\langle\text{object}\rangle\langle\text{attribute value}\rangle)\rangle$$

$$\sigma_{\text{relation}}\left(\langle\text{relational\_values}\rangle\right)\langle\text{cluster\_strings}\rangle \Rightarrow \{(\langle\text{relational}\rangle(\langle\text{relational\_value}\rangle\langle\text{object}\rangle - \text{i}\langle\text{object}\rangle - \text{j}))\}.$$

As an example to find a pair of objects such that the blue object precedes ($<$) the red object along the spatial dimension V, the σ-operator instance is

$$\sigma_{\text{color}}(\text{blue}, \text{red})(\text{V} : \text{A} < \text{B}) = (\text{V} : (\text{color A blue}) < (\text{color B red}))$$

where $(\text{V} : \text{A} < \text{B})$ is a cluster string.

In case of uncertainty the input and output to the σ-operators may include an attribute corresponding to a specific belief value. The σ_{type}- and the σ_{motion}-operators may include this attribute. The σ_{type}-operator is concerned with matching between objects found in a sensor image and objects stored in a library database and where both objects are described in the same terms that may be either qualitative or quantitative. Traditionally matching was regarded as a part of information fusion. Generally, however, the σ_{type}-operator and its result, i.e., a set of objects and their corresponding normalized belief values, can be expressed as follows when the input to the operator is a single cluster:

$$\sigma_{\text{type}}(\text{type\_value})\langle\text{cluster\_strings}\rangle \Rightarrow \langle\text{cluster\_strings\_of\_}(\langle\text{type\_value}\rangle\langle\text{object}\rangle - \text{i}\langle\text{nbv}\rangle - \text{i})\rangle$$

where $\langle nbv\rangle$ corresponds to the normalized belief value that in practice becomes an attribute to the actual object. An instance of this is

$$\sigma_{\text{type}}(\text{car})(\text{U} : \text{A} < \text{B} < \text{C}) = (\text{U} : (\text{car A } 0.7) < (\text{car B } 0.1) < \text{C}).$$

The σ_{motion}-operator can be expressed as follows:

$$\sigma_{\text{motion}}\left(\langle\text{motion\_value}\rangle\right)\{\langle\text{cluster} - \text{sequence}\rangle|\langle\text{cluster}\rangle\} \Rightarrow \{(\langle\text{motion\_value}\rangle\langle\text{object}\rangle - \text{i})\}.$$

For example, to find three moving objects, each preceding the other, along the spatial dimension U, the operator instance is:

$$\sigma_{\text{motion}}(\text{moving})(\text{U} : \text{A} < \text{B} < \text{C}) = (\text{U}(\text{moving A}) < (\text{moving B}) < (\text{moving C})).$$

B. ϕ-Operators

The ϕ-operators are more complex because they are concerned with sensor data fusion. Consequently these operators require more complex expressions as well as input data in different time periods from multiple sensors.

The ϕ_{fusion}-operator performs sensor data fusion from heterogeneous data sources to generate fused objects. Fusion of data from a single sensor in different time periods is also allowed. The output of the ϕ_{fusion}-operator is some kind of high level, qualitative representation of the fused object, and may include object type, attribute values and status values. The output may also include a normalized belief value for each fused object.

The fusion operators rely upon solutions to the association problem [11], which is generally concerned with how to determine whether an object of a certain class observed at one time is the same object observed at a later time. The observations may be made by the same sensor or by different sensors of either the

same or different types. This is a complex problem [11] that normally requires probability-based approaches such as Baysian networks [10] or Dempster–Shafer theory of evidence [23].

The output from the fusion operator may serve as the answer to a query. This result may consist of a list of objects each having a belief value. The object with the highest belief value is the most likely answer to the query and thus should come first in the list. The general description of the fusion operator is therefore

$$\phi_{\text{fusion}}(\text{cluster}|\text{cluster}-\text{string}) \Rightarrow \{(\text{type},\text{obj},\text{nbv})|(\text{type},\text{obj},\text{nbv})-\text{list}\}.$$

Last, but not least, a $\phi_{\text{similarity}}$-operator that takes two images from either the same sensor or two different sensors, transforms them into unified cluster data and then establishes the similarity between the two with respect to their contents, can be described as

$$\phi_{\text{similarity}}(\text{image}-\text{cluster}_1,\text{image}-\text{cluster}_2) \Rightarrow \{(\langle\text{similar}\rangle\text{image}-\text{cluster}_1,\text{image}-\text{cluster}_2,nbv)\}.$$

The similarity operator relies upon qualitative techniques such as the technique described in [7]. Similarity retrieval has for a long time been part of image information retrieval and includes techniques for iconic indexing [4]. However in similarity retrieval the complete content of the images, instead of just single objects, is of concern. Similarity retrieval is also less concerned with the identity of the objects but with sets of objects, and their relationships and positions.

V. Sensor Data Fusion

In sensor data fusion [4], queried objects from the different sensors need be associated to each other in order to determine whether they are the same object registered at different times and at different locations. Tracking is another problem that is closely related to the sensor data fusion problem. In tracking, the problem is to verify whether different object observations represent the same or different objects. Another problem of concern is the uncertainties of the sensor data. Consequently, there is a demand for a well-structured and efficient general framework to deal smoothly with a large number of different query types with heterogeneous input data.

In sensor data fusion, the main issue is how to develop a general framework that can be applied to carry out the fusion process in query processing. The fusion framework that will be applied in this work is based on a method described by Horney *et al.* [25], which makes use of a technique that is quite general and can be applied not only to fusion of sensor data but also to other problems. However this fusion method should be *replaceable* by other methods, which will be tested later. The fusion method is chosen because it is efficient, simple to implement, demonstrates a high degree of generality and fits well into the environment of concern in this research work. Other related approaches to fusion are given by Parker [18] and Klein [14].

For certain sensors the objects can only be determined with respect to their type but rarely with respect to their identity. Therefore classification of the objects is necessary. This is a process that can be carried out in a matching algorithm that should display a result that includes not only the type of the object but a normalized belief value, nbv, associated to the observed object type. A number of attributes and state variables can be extracted from the sensor data where the actual types of attributes depend on the actual sensor types. Among the most important state variables are orientation, type and position, direction of motion, speed and acceleration. Most attributes and state variables, such as position and orientation, may be determined either in quantitative or in qualitative terms. In ΣQL, reasoning is generally performed on qualitative information, basically represented in terms of Symbolic Projection. For this reason, it is an advantage to use qualitative input to the fusion process as well.

TABLE I
Example of Basic Spatial State Values and Their Belief Values and With Their Qualitative Value Sets

attribute	Belief value	value set
Position	$B(A=B \mid pos_B)$	{vl, l, m, h}
Direction	$B(A=B \mid dir_B, pos_B)$	{l, m, h}
Type	B(A)	{l, m, h}

The following probabilities are determined during the fusion process for indication of the belief values in different state variables.

The probability $\mathrm{P(A = B)}$ is the probability that object observation A is the same as object observation B.

$\mathrm{P(A = B|pos_B)}$ is the probability that the observations A and B can be associated to each other given the position of B relative the position of observation A.

$\mathrm{P(A = B|dir_B, pos_B)}$ is the probability associating the observations A and B to each other given the position of B and its direction relative the position of A.

These probabilities can be replaced by a set of *belief values* as can be seen in Table I. Here the belief values are of qualitative type but they may be quantitative as well.

Given some spatial attributes and their probabilities or normalized belief values, as in Table I, the basic means to carry out the fusion process are available. The attributes/state values and their belief values correspond to information obtained from the sensor data as a consequence of the queries. However, the result of this process must clearly be represented in a structure that efficiently supports the fusion process. The structure proposed for completion of the fusion process is to a high degree related to a scheme proposed by Chong *et al.* [7]. The structure here is called a *fusion scheme*. A difference between the two schemes is that the one used by Chong *et al.* is merely concerned with determination of tracks of objects, i.e., both the tracks registered by individual sensors and tracks from the fusion process, which they call a system track. In this work, the scheme is concerned with all information that relates to the objects, i.e., including state variables and attributes.

Generally, the information acquired from the sensors is stored as instance descriptions and eventually forwarded into the fusion composition table (FCT), which serves as input to the fusion routine. FCT is organized as a set of *pages*, one for each observation. These pages are called object instance pages (OIP).

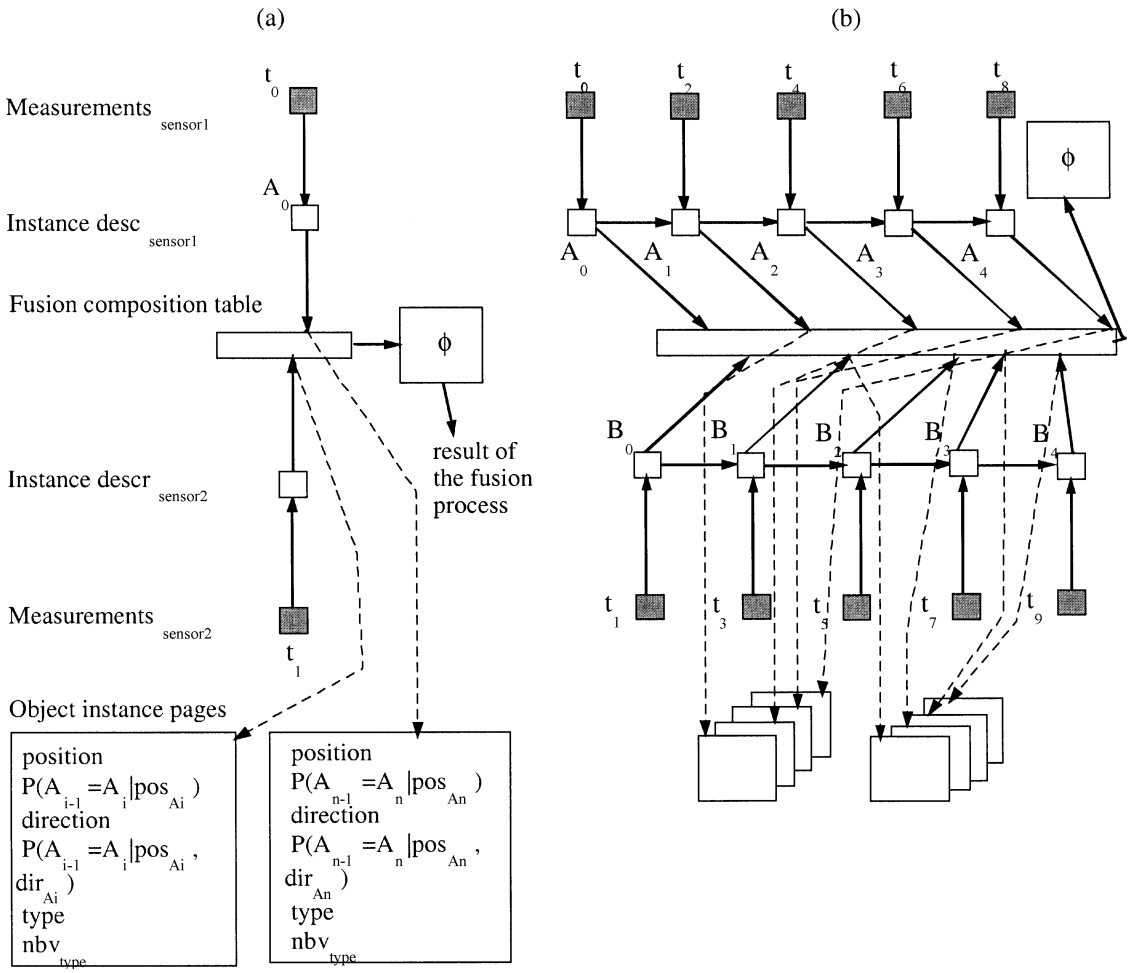


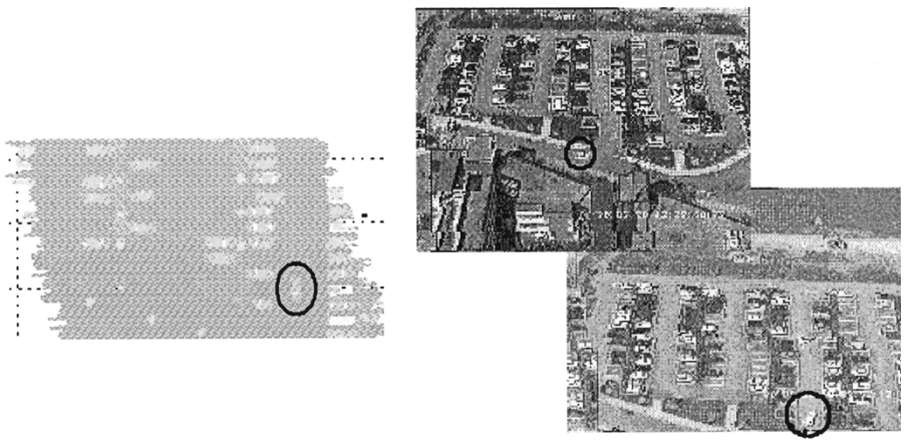
Fig. 2. (a) Flow of the sensor data fusion process for two single instance sensors. (b) Flow of the sensor data fusion process for two sensors with multiple instances.

In some applications it may be convenient to represent these pages in terms of HTML code since that way it may be possible to carry out the fusion process in a distributed environment. The fusion process can now be seen as a process where information acquired from a set of sensors is fed into the query system and its fusion routine. Query execution corresponds, in principle, to a process where a multudimensional search space through projection is reduced to clusters in lower dimensionality. In the extreme, a cluster may correspond to a single object instance, which together with its belief value is fed into the FCT. Eventually the fusion is carried out by the ϕ-operator. This scheme is automatically created by the system. The structure of the scheme may vary depending on the type of sensor information that is under consideration, e.g., image sequences from a video generate sequences of pages while for single images just a single page is generated [see Fig. 2(a) and (b)].

The fusion process proceeds all through the execution of the query feeding the FCT and produces a response to a proposition, i.e., which of the registered objects can be associated to each other. The registered objects are determined by the query interpreter and inserted into the FCT for a fusion step. Since the object instance pages in the FCT contains the actual attribute values and their corresponding qualitative belief values it is feasible for the ϕ-operator to answer propositions like: "*are all the observations in the FCT corresponding to the same object*." The result of the fusion process will tell which object observations that can be associated to each other, although the result, to some degree, exhibits a qualitative uncertainty, which must be evaluated by the user in his final decision process.

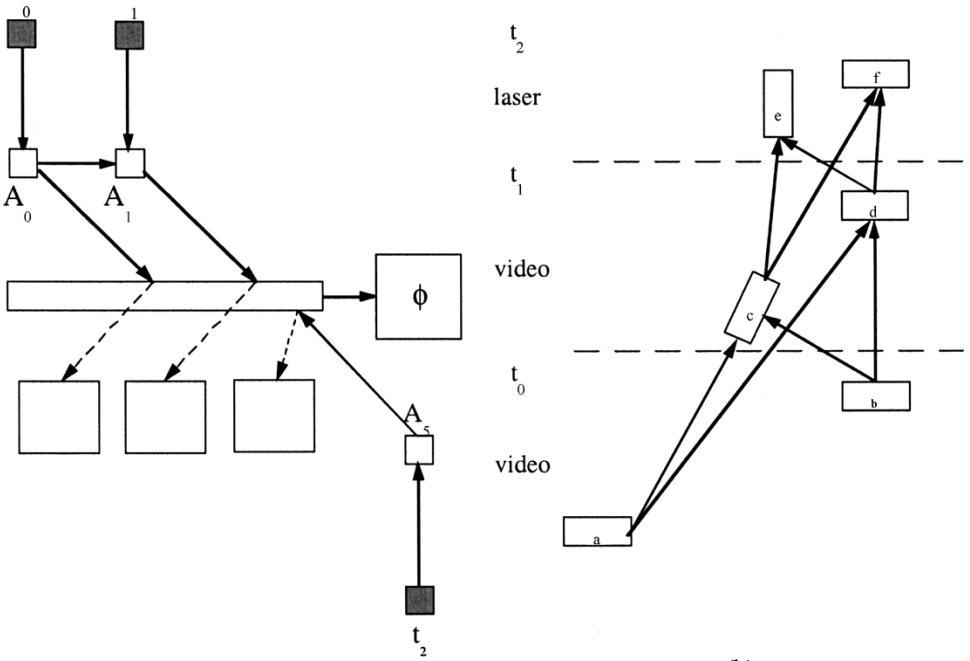
The fusion method is applied to the FCT under consideration of the current proposition, which depends on the query, the sensor types and the data. For all the object combinations and with respect to the qualitative belief values for the various attributes and state variables, the process is carried out with respect to the given proposition. The value set of the qualitative belief values is {h(igh belief), m(edium belief), l(ow belief)}. In practice, this means that a score need to be given to the actual voting alternative of FCT, which may be 3 for a high belief value, and so on. The scores given for a certain attribute alternative may be added up and a total score is available for each alternative. The alternative with the highest score is considered the most probable and the remaining alternatives are ordered with respect to their rank.

A query demonstrating the usage of ΣQL with the sensor sources video and laser-radar can now be provided. Given the input information from the laser-radar [Fig. 3(a)] and the video camera [Fig. 3(b)], the query is as follows: *is there a moving vehicle present in the given area and in the given time interval*? In this query, the laser-radar data can be used as an index to the video. In this way, most of the computational efforts can be



(a) (b)

Fig. 3. (a) Laser radar image of a parking lot with a moving car (encircled). (b) Two video frames showing a moving white vehicle (encircled) while entering a parking lot.



(a) (b)

Fig. 4. (a) Flow of the sensor data fusion process for the video/laser-radar example. (b) Set of objects observed at different times and where the arrows indicate the different situations in a simplification of Fig. 3(a) and Fig. 3(b).

avoided since the vehicles can be identified in almost real time in a laser-radar image. However, in the laser-radar used here, it cannot be determined whether an identified vehicle is moving or not. Consequently, once a vehicle has been identified in a laser–radar image, we need to determine whether it is moving by analyzing a small number of video frames taken in a short time interval. This is possible to accomplish because the location of the vehicle at a certain time is known from the laser-radar information, which is illustrated in Fig. 3(a) and (b). The three images illustrate a situation where a car is first about to enter a parking lot (the two video frames) and at a later time the car has entered the parking lot (the laser-radar image).

The query is logically split into two parts, one looking for the vehicles in the laser-radar image and another looking for the vehicles in the video frames during the same time interval as the laser–radar image. The result of these two subqueries are fused by applying the fusion operator $\phi_{\text{type,position,direction}}$, which includes the fusion procedure with the voting scheme. The fusion is applied with respect to type, position and direction including also their belief values.

The query demonstrating the problem can thus be expressed as

$$\begin{aligned}
&\phi_{\text{pe,position,direction}}\\
&(\sigma_{\text{motion}}(\text{moving})\sigma_{\text{type}}(\text{vehicle})\\
&\sigma_{\text{xy}}(*)\\
&\sigma(\text{T})_{\text{T mod 10=0 and T>t1 and T<t2}}\\
&\sigma_{\text{media_sources}}(\text{video})\text{media_sources}\\
&\sigma_{\text{ype}}(\text{vehicle})\sigma_{\text{xyz}}(*)\\
&\sigma(\text{T})_{\text{T>t1 and T<t2}}\\
&\sigma_{\text{media_sources}}\ (\text{laser_radar})\text{media_sources})\,.
\end{aligned}$$

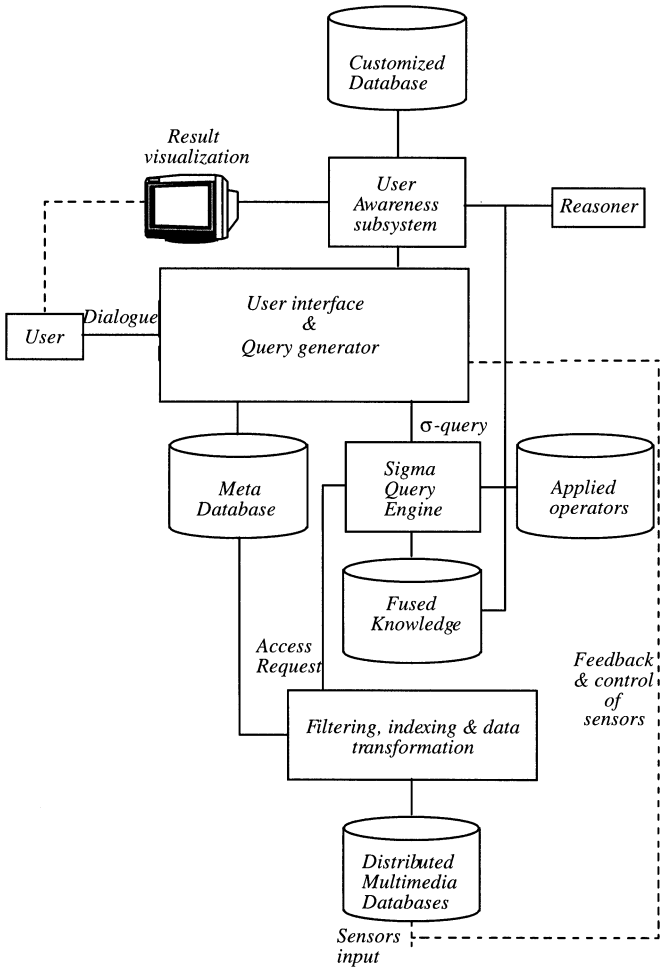


Fig. 5. System architecture for sensor data fusion.

The ϕ-operator performs the fusion process during the execution of the query. Each vehicle identified in any of the images during the query is subject to the determination of these state variables and attributes and their belief values. Fusion is then completed with respect to all the possible combinations of the identified vehicles. The proposition that takes place in the fusion process for this query can thus be formulated as: *can observation A be associated to observation B*?

A simplified description of the situation in Fig. 3(a) and Fig. 3(b) is shown in Fig. 4(b), where just one of the parked cars is left, i.e., the one to the right. The arrows or rather all their combinations taken in sequence correspond to all the possible combinations that will be subject to fusion. Consequently there are eight alternatives to discriminate between.

VI. System Architecture for Sensor Data Fusion

Fig. 5 illustrates a generic system architecture for sensor data fusion. A *user* interacts with a user interface to produce a *σ-query*. This can be done directly or through a domain specific virtual environment customized to the user's level of expertise and described in the *user awareness subsystem*. Once a *σ-query* has been formulated, it can be compiled and its correctness and feasibility checked. For a *σ-query* to be executable all the required operators must have been implemented in the system.

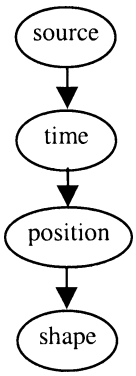


Fig. 6. Hierarchy of dimensions.

The knowledge of what type of queries the system can execute is given in a knowledge base formed by the *Meta Database* and the *Applied operators*. The Meta Database contains a set of tables describing which operators are implemented in the system and how they can be used. The Applied operators are the set of algorithms that implement the operators. Once a query has been successfully compiled the *Sigma Query Engine* executes it against the *Distributed Multimedia Database* or directly against the *sensors input*. During the execution it applies *input filtering, indexing and data transformation* required by the application of the operators in the query. The execution of a query produces *(Fused) Knowledge* that can then be used to modify the virtual environment in which the user operates, providing useful feedback through appropriate *result visualizations*.

One of the important characteristics of this architecture is that the same query language can be used to access any data source. This is possible due to the fact that the Meta Database and the Applied Operators hide the information to be processed, providing the ΣQL processor with a general data model, on which programmers base their queries.

VII. Data Model

The ΣQL query language makes use of the Dimension Hierarchy Data Model (DHDM) to support a common and uniform treatment of heterogeneous data sources. The dimension hierarchy is a generic structure, based upon which different instances of representation schema can be constructed to best suit the sources and the queries. In this data model, a data source type is represented by a set of table schemas called *Representation Schema* (RS for short). Each table schema describes a *dimension* along which to cluster the sources of that type and is characterized by a name and a list of fields: $\mathrm{dim} = [\mathrm{attr}, \mathrm{dim}_1, \mathrm{dim}_2, \ldots, \mathrm{dim}_\mathrm{k}]$ (with $\mathrm{k} >= 0$) where

- dim is the name of the table schema;
- attr is a proper attribute of dim;
- $\mathrm{dim}_1, \ldots, \mathrm{dim}_\mathrm{k}$ are the names of other table schemas.

Given an RS, a *Representation Schema Instance* (RSI) of a source is a set of instances of the table schemas in RS. In the following, an instance of a table schema dim will be referred to as a table of type dim and is composed by a possibly ordered set of rows $(val_{attr}, dval_1, dval_2, \ldots, dval_k)$ where val_{attr} is a value for the attribute $attr$ of dim and $dval_j$ is a reference to a table of type dim_j.

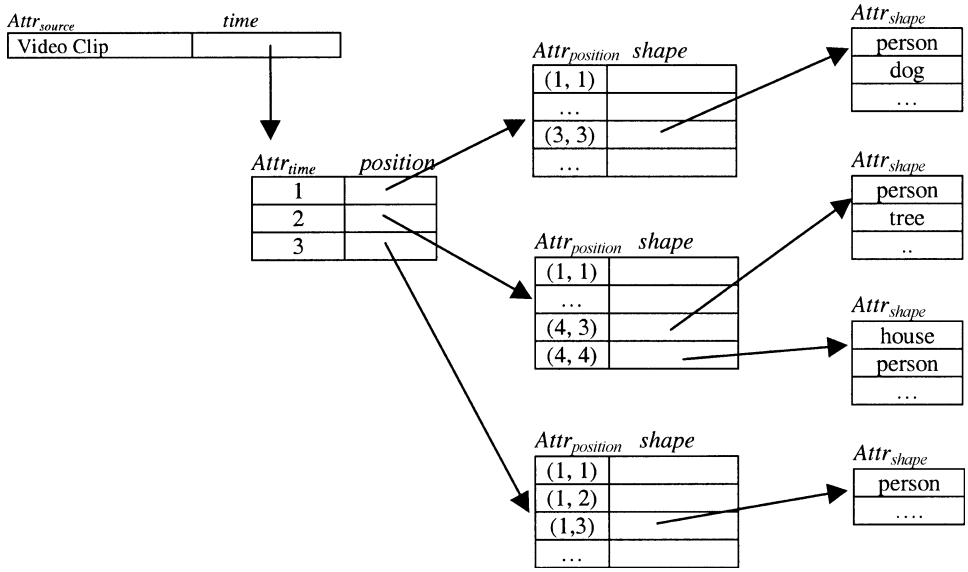


Fig. 7. RSI of a short video clip illustrating the spatial/temporal position of the people shown in the video clip.

Let us give an example showing how to represent a video clip according to DHDM. The first thing to do is to create an RS for video clips. The main idea is to successively project or decompose the video clip along some dimensions. On one hand, the type and number of dimensions strictly depends on the level of detail of the video clip representation we need to reach in order to get to the entities of interest occurring in the video clip. On the other hand, each dimension must have been implemented as an operator that must be present in the *Applied operators* repository. As an example, let us suppose we want to represent a video clip and are interested in representing the spatial/temporal position of the people shown in it. The already implemented dimensions we want to use are then the $time$, the $position$ and the $shape$, in this order. The resulting RS is given by the following table schemas:

$$\begin{aligned} source &= [\mathrm{Attr}_{\mathrm{source}}, \mathrm{time}] \\ time &= [\mathrm{Attr}_{\mathrm{time}}, \mathrm{position}] \\ position &= [\mathrm{Attr}_{\mathrm{position}}, \mathrm{shape}] \\ shape &= [\mathrm{Attr}_{\mathrm{shape}}]. \end{aligned}$$

The initial dimension $source$ defines the sources to be processed $(\mathrm{Attr}_{\mathrm{source}})$ and the dimension $(time)$ along which these are to be decomposed. Similarly the dimension $time$ defines the temporal instances to be considered $(\mathrm{Attr}_{\mathrm{time}})$ and the dimension $(position)$ along which to decompose the resulting entities. The dimension $shape$ does not refer to any other dimension since we are not interested in further details. It can be noted that the chosen RS actually defines a hierarchy of dimensions along which to decompose the source. The underlying hierarchy is shown in Fig. 6.

Let us suppose that the video clip has three frames, each partitioned into 4×4 slots, and shows, among others things, four people in the slot position (3, 3) of the first frame, the slot positions (4, 3) and (4, 4) of the second frame, and the slot position (1, 3) of the third frame, respectively. Fig. 7 shows the corresponding RSI.

It should be noted that the table of type $source$ has only one entity, i.e., the video clip under consideration. This entity refers to a table of type $time$ containing three entities that, in the above example, are the three frames of the video. Each frame refers to a table of type $position$, each containing a set of entities corresponding to the slots partitioning the frame. Finally, the tables of type $shape$ contain entities corresponding to the actually recognized shapes in each slot. For the sake of clarity, not all the tables are shown in Fig. 7.

An alternative way of depicting the RSI shown in Fig. 7 is the graph view given in Fig. 8 where each dimension (table) is depicted with an oval and each entity (row) is depicted by a box. Again for the sake of clarity, not all the graph nodes are shown.

From the graph view of Fig. 8 the hierarchy of dimensions underlying the representation schema becomes evident. As a matter of fact, the hierarchy shown in Fig. 6 is easily built from Fig. 8 by disregarding the entity nodes.

Given a data source, there are many ways of representing it. This means that a source can be represented with different RSIs, depending on which information in the data source needs to be represented and then queried. Therefore, flexibility is the main motivation behind this approach. As an example, an alternative dimension hierarchy for representing a video clip is shown in Fig. 9. Here, again, we look at a video clip as a sequence of frames, but now each frame is seen as a set of separated rows and columns. Moreover, each frame row or column is seen as a set of shapes of a certain type.

VIII. ΣQL Language

To write a ΣQL query on a given data source, a programmer must know which *representation schemas* are available on that source and, for each schema, the corresponding dimension hierarchy. This is similar to SQL, where a programmer must know the table names and the table attributes of the relational database for the query. To better explain the behavior of a ΣQL query, we first need to refine the concept of clustering first introduced in Section III.

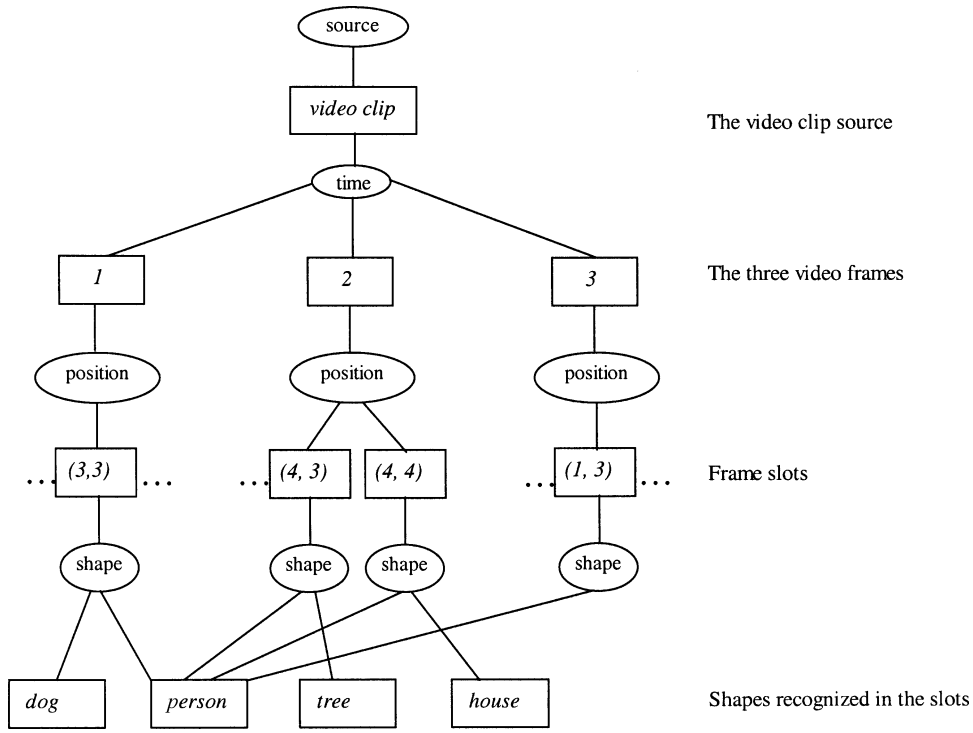


Fig. 8. Graph view of the RSI shown in Fig. 7.

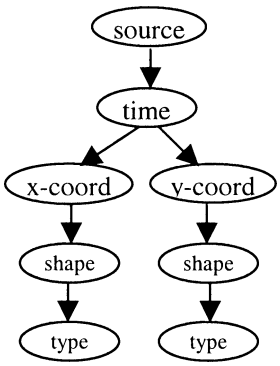


Fig. 9. Alternative hierarchy of dimensions to represent a video clip.

Given a *representation schema instance*, i.e., a set of instantiated tables representing a source, a *cluster* is represented by a set of table rows from one or more tables. Given a subset of tables *st* in a *representation schema instance*, with the term *clustering* we refer to the operation of creating a cluster from *st* by deleting or simply not selecting some of the table rows in *st*.

Examples of clustering on the RSI of Fig. 7, are

$\{time : *\}$ that creates a cluster, on the table of type "time," with all the video frames;

$\{time$: 1, 3$\}$ that creates a cluster, on the table of type "time," with only the video frames with time stamps 1 and 3;

$\{position : (3, *)\}$ that creates a cluster, on the tables of type "position," with all the video frame areas occurring in the third column of each frame.

A clustering operation on a subset *st* of tables can work in one of the following modes: *destructive mode*, where all the table rows or entities that are not selected are deleted from *st*; and *conservative mode*, where all the current table rows or entities that are not selected are kept in *st* but not included in the cluster, i.e., they are virtually deleted. They will be used to show the context of the selected cluster elements.

On the other hand, a cluster can be

open the cluster, created by a clustering operation, is open to successive destructive clustering operations, i.e., the elements in the cluster can be deleted by successive clustering and selection operations;

close the elements in the cluster can only be virtually deleted by successive clustering and selection operations.

As an example, if the clustering operation $\{time$: 1, 3$\}$ on the table of type "time" in Fig. 7 is considered destructive, then the second video frame, i.e., the second row in the table, will be deleted from the RSI.

Basically, a ΣQL query refines a source RSI by successively clustering it on some dimension under some constraints. In other words, by using clustering operations, the ΣQL query selects and/or deletes entities in a data source. The result of a query can be used by a nesting query to further refine the data source or can be output according to some presentation criteria.

The following is a simple ΣQL query template:

```
SELECT dimension_list
CLUSTER clustering_op_list
FROM source [query_re-
sult][{nested_query}]
[WHERE condition]
[PRESENT presentation_method]
```

The FROM clause requires the specification of the input to the query: this can be either the name of the data source, or the result of a previously executed query or a nested ΣQL query. The SELECT clause requires the list of the dimensions along which to cluster the input. The CLUSTER clause requires a list of clustering operations, one for each dimension declared in the SELECT clause. The form of a clustering operation is as those shown above:

$$\{dimension : [\text{filt\_mode}]\ \text{Val}^1_{dimension}, \ldots, \text{Val}^n_{dimension}\}$$

with the addition of the optional keyword "filt_mode" standing for one of the four filtering modes:

1) PUBLIC if the clustering operation is to be *destructive* and the resulting cluster is to be *open*.
2) PRIVATE if the clustering operation is to be *destructive* and the resulting cluster is to be *close*.
3) ALL&PUBLIC if the clustering operation is to be *conservative* and the resulting cluster is to be *open*.
4) ALL&PRIVATE if the clustering operation is to be *conservative* and the resulting cluster is to be *close*.

If no filtering mode is provided then the clustering operation is considered PUBLIC. Each $\text{Val}^i_{dimension}$ refers either to the wild character "$*$" indicating that all the dimension values must be considered, or to a constant value or to a variable whose constraints are to be set in the WHERE clause. The use of the filtering modes allow us to define the granularity of the context of a query result.

If the query is not nested in any other query it may not require any clustering and behave as a standard SQL query. In this case the keyword CLUSTER is followed by the wild character "$*$."

The PRESENTATION clause requires the name of a presentation module that should have been defined "ad hoc" for the presentation of the particular type of the query input. If omitted, the presentation of the query output will be done by using a default presentation module.

Given the dimension hierarchy in Fig. 6 representing a video clip, we want to write a query to retrieve the columns of the second frame in a video clip, containing some people.

```
// Subquery 4
SELECT type
CLUSTER *
FROM
  {
  // Subquery 3
  SELECT shape
  CLUSTER {shape: *}
  FROM
    {
    // Subquery 2
    SELECT x-coord
    CLUSTER {x-coord: *}
    FROM
      {
      // Subquery 1
      SELECT time
      CLUSTER {time: 2}
      FROM Video
      }
    }
  }
WHERE type = 'person'
```

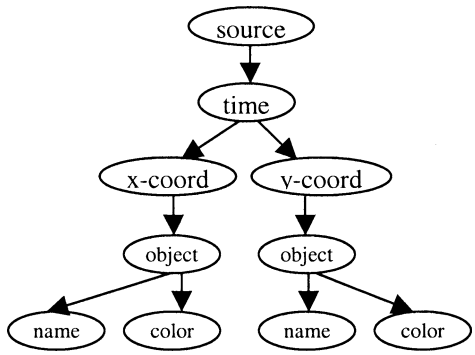


Fig. 10. Another possible hierarchy of dimensions for representing video clips.

The query must be read starting from the inner subquery and proceeding with the closest enclosing ones. Subquery 1 clusters the video source along the $time$ dimension and extracts the frame with time stamp 2. Subquery 2 extracts all the columns from the frame by clustering along the dimension *x-coord*. For each column, subquery 3 extracts all the shapes from all the columns by using the dimension $shape$.

Subquery 4 extracts all the people from all the shapes by asking for shapes of type "person." The final result will then be only the columns of frame 2 containing shapes representing people while all the other shapes will be lost. Note that subquery 4 does not use clustering but, more likely to an SQL query, uses the WHERE clause to further refine the data. This is possible because subquery 4 is not nested in any other subquery.

Given the hierarchy of dimension of Fig. 10, we will now write some queries to retrieve information from a video clip.

In the following, we will consider the simplified input video clip shown in Fig. 11.

Query 1: Extract all the video frame columns containing entities with the name John and entities with the name Bill.

```
SELECT name
CLUSTER PUBLIC *
FROM
  {
  SELECT x-coord
  CLUSTER { x-coord: PUBLIC *}
  FROM
    {
    SELECT time
    CLUSTER { time: PUBLIC *}
    FROM Video
    }
  }
WHERE name CONTAINS 'John' AND name CON-
  TAINS 'Bill'
```

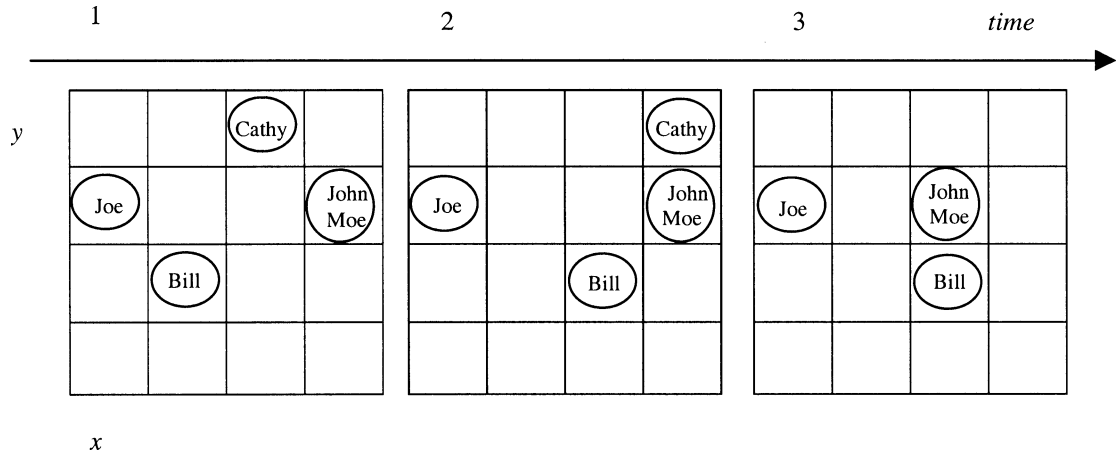


Fig. 11. Simplified video clip.

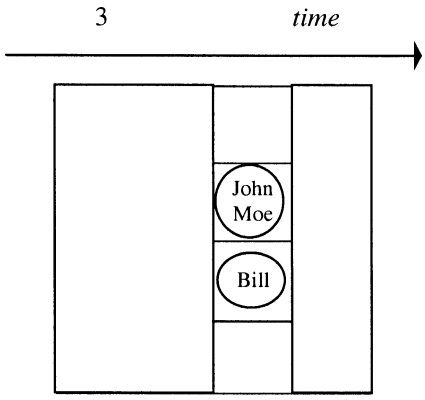


Fig. 12. Extracting all the video frame columns.

The result is given by the following frame column shown in Fig. 12.

Query 2: Extract all the video frame columns containing an entity with name Joe Moe. We want to know who else is in the same frame columns.

```
SELECT name
CLUSTER *
FROM
  {
  SELECT object
  CLUSTER {object: PRIVATE *}
  FROM
    {
    SELECT x-coord
    CLUSTER {x-coord: PUBLIC *}
    FROM
      {
      SELECT time
      CLUSTER {time: PUBLIC *}
      FROM Video
      }
    }
  }
WHERE name CONTAINS 'John' AND name CON-
  TAINS 'Moe'
```

The result is shown in Fig. 13. Note that because of the PUBLIC filtering mode along the dimensions *frame* and *column*, all the frames and columns that are not concerned with the final query have been eliminated. However, because of the filtering mode PRIVATE along the dimension *object*, all the objects have been kept both in the "SELECT object" and "SELECT name" subqueries. The final result shows the required objects in a highlighted form and also (nonhighlighted) others forming their column context.

Let us consider now the hierarchy of dimensions in Fig. 14 for querying the Web.

Let us now consider the following query:

```
SELECT pattern
CLUSTER *
FROM
  {/* subquery 1*/
  SELECT image, url
  CLUSTER {image, url: ALLPRIVATE *}
  FROM
    {/*subquery 2 */
    SELECT page
    CLUSTER { page: PUBLIC *}
    FROM WWW
    }
  }
WHERE pattern CONTAINS 'Bill Clinton'
AND pattern CONTAINS 'dog'
```

Subquery 2 allows the external queries to search one Web page at a time. Because of the PUBLIC filtering mode all the pages that will not match the whole query will be deleted. Subquery 1 will project each page against the dimensions *image* and *url* illustrating all the images and the address of each page. The outer query will then project all the resulting clusters of single images against the dimension *pattern* selecting the images with patterns corresponding to Bill Clinton and a dog. Because of the ALLPRIVATE filtering mode on *image* and *url*, the query will return all the complete web pages containing at least an image with Bill Clinton and a dog, together with their url address.

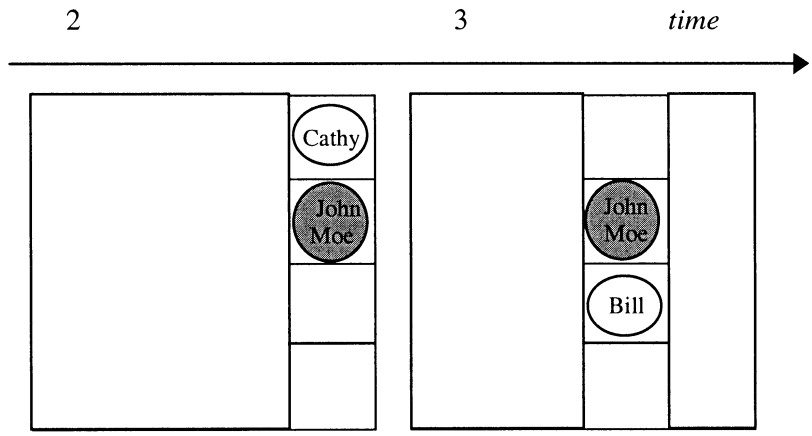


Fig. 13. Extracting all the video frame columns containing an entity with name Joe Moe.

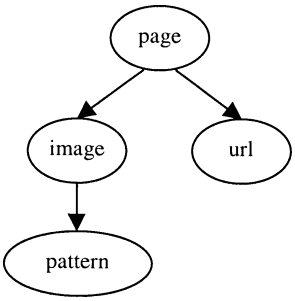


Fig. 14. Possible hierarchy of dimensions for the web.

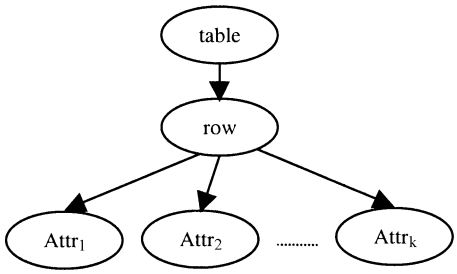


Fig. 15. Dimension hierarchy template for relational databases.

IX. Sensor Data Fusion Using the Merge Operation

In this and the next two sections, several important topics in sensor data fusion, including fusion by the merge operation, the detection of moving objects, and the incorporation of belief values, are discussed in detail. Although we cannot say they are the only topics of interest in sensor data fusion, we found these topics to be very important in order for sensor data fusion to work.

From the database point of view the merge operations are used to fuse information coming from two or more *RSI*s to produce a new *RSI*. There are many possible merge operations and they strictly depend on the particular environment the query has to work in. A simple example of merge operation is the CARTESIAN_MERGE operation defined in the environment of the relational databases. Each operator can be defined on one or more distinct pairs of RSI, but the same operator may not have different implementations for the same pair of RSI. It is important to note that the merge operators may behave differently depending on the clusterings defined in the queries used in the operation. The template of an application of a merge operation is

```
⟨MERGE OPERATOR⟩
⟨FROM⟩
  {select operation [merge operation]}
  ,
  {select operation [merge operation]}
```

Given the hierarchy of dimensions template for relational databases in Fig. 15, and a relational database containing, among other things, a table "Names" with attributes SSN and name containing data about some people. Suppose we want to look for all Web pages containing images about the people whose name is in the database. To do this, we first extract the table "Names" from the database by temporarily storing it in resultTable. This is done by using the following INSERT INTO clause:

```
INSERT INTO resultTable
{
  SELECT Table
  CLUSTER { Table: 'Names'}
  FROM DataBase
}
```

The final result is obtained through the following composed query:

```
MERGE_AND
FROM
  {
  SELECT name
  CLUSTER {name: *}
  FROM resultTable
  },
  {
  SELECT pattern
  CLUSTER {pattern: *}
  FROM
    {
```

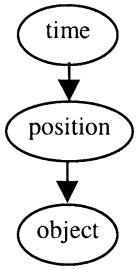


Fig. 16. Dimension hierarchy for VIDEO.

```
SELECT image
CLUSTER {image: *}
FROM
  {
   SELECT url
   CLUSTER {url: PUBLIC *}
   FROM WWW
  }
}
}
```

In this query, the operator MERGE_AND returns a RSI made of the Web documents containing images with the people whose names occur in the name field of the table "Names."

X. Detecting Moving Objects in a Video

Let us consider a data source VIDEO described by an RSI based on the dimension hierarchy in Fig. 16.

This hierarchy does not allows us to write ΣQL queries to detect which objects are moving in a given time interval, though it allows us to select the frames in an interval. To detect movement, we would need to find two frames in the interval containing the same object with high belief value. If the two occurrences are in the same positions then it is likely that the object is still. Otherwise the object is likely to be moving.

In order to identify the moving objects in a VIDEO by using the ΣQL language we need to introduce the concept of *derived attribute*. A derived attribute of an entity stores information produced by the processing of already known information. The derived attributes are calculated by using *transformation methods* that translate an RSI into another. This new RSI describes the same search space but it may be based on a different dimension hierarchy.

As an example, let us consider the following ΣQL query based on the dimension hierarchy of Fig. 16.

```
/* Query 1 */
INSERT INTO video_piece
{
  SELECT time
  CLUSTER {time : PUBLIC * ALIAS vTIME}
  FROM aVIDEO
  WHERE vTIME >= t_in AND vTIME <= t_fin
}
```

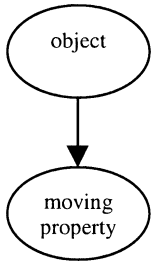


Fig. 17. Hierarchy dimension generated by the show_state algorithm.

Query 1 stores in *video_piece* all the frames of *aVIDEO* in the interval $[t_{in}, t_{fin}]$. In order to detect all the moving objects in such interval we write the following query:

```
/* Query 2 */
SELECT moving_property
CLUSTER *
FROM
  {
  SELECT object
  CLUSTER: show_state: { object:
PUBLIC *}
  FROM video_piece
  }
WHERE moving_property='moving'
```

The inner subquery in Query 2 not only selects all of the objects in the required interval but also builds on them the new dimension *moving_property* by using the transformation method *show_state*. The old RSI is then translated into the new RSI in Fig. 17 that is then used to represent all the objects in *video_piece*. The external subquery in Query 2 is then able to select all the objects with *moving_property*='moving'.

The transformation method *show_state* calculates the *moving_property* attribute value of each object by checking the object coordinates in each frame. In order to give the appropriate values, for each object, it has to calculate the object identity over the frames.

The change of *RSI* in ΣQL is an operation very similar to the creation of a *view* in SQL: indeed, starting with an RSI we build a new RSI by structuring already stored information in a different way. On the other hand the attribute *moving_property*, calculated by the algorithm translating the *RSI*, may be considered as a *derived attribute* in SQL, since it provides information derived by other existing data.

XI. Belief Values

Detecting the object identity over several frames is not an easy task and most of the times it is not possible to state with certainty whether two objects in two different frames are the same. It becomes then necessary to use belief values for derived attributes. In this case we can rewrite Query 2 as follows:

```
/* Query 3 */
SELECT moving_property
CLUSTER *
```

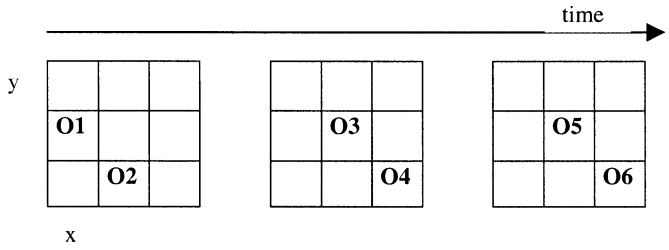


Fig. 18. Example of video_piece.

```
FROM
  {
  SELECT object
  CLUSTER: show_probable_state: { ob-
ject: PUBLIC *}
  FROM video_piece
  }
WHERE moving_property >= 0.9
```

This query selects not only the objects that certainly moved in the required interval in *video_piece* but also the objects that moved with a belief value. As an example, supposing to use the dimension hierarchy of Fig. 18, let us apply Query 3 to the input *video_piece* shown in Fig. 18.

The inner subquery uses the method *show_probable_state* to obtain an *RSI* based on the dimension hierarchy of Fig. 17. The transformation method will use the function *similar* defined as follows:

$$\boldsymbol{similar} : \boldsymbol{object} \times \boldsymbol{object} -> [0,1]$$
$$\text{such as } \boldsymbol{similar}(\boldsymbol{a},\boldsymbol{b}) = \boldsymbol{similar}(\boldsymbol{b},\boldsymbol{a}).$$

This function returns the belief value for the identity of its two object arguments:

$$\boldsymbol{similar}(\boldsymbol{O1},\boldsymbol{O3}) = 0.7$$
$$\boldsymbol{similar}(\boldsymbol{O3},\boldsymbol{O5}) = 1$$
$$\boldsymbol{similar}(\boldsymbol{O1},\boldsymbol{O5}) = 0.7$$
$$\boldsymbol{similar}(\boldsymbol{O2},\boldsymbol{O4}) = 1$$
$$\boldsymbol{similar}(\boldsymbol{O4},\boldsymbol{O6}) = 0.9$$
$$\boldsymbol{similar}(\boldsymbol{O2},\boldsymbol{O6}) = 0.9$$

(for all the other combinations the belief value is zero).

Due to the execution of *show_probable_state*, the objects in *video_piece* are represented as shown in Fig. 19.

The probability that O_1 moved is 0.7 since O_1 is the same as O_3 and O_5, and it has a different x-coordinate. O_2 and O_4 are occurrences of the same object with belief value equal to 1 and have different positions, then we can state that the corresponding object moved. O_3 and O_5 are occurrences of the same object and have the same position in the frames they appear, O_3 (resp. O_5) is the same as O_1 with belief value 0.7 and its x-coordinate is different from that of O_1, then it moved with belief value 0.7. O_6 and O_2 (resp. O_4) are the same object with belief value 0.9 and have different x-coordinates, then O_6 moved with belief value 0.9. The external part of Query 3 deletes all the objects with belief value less than 0.9.

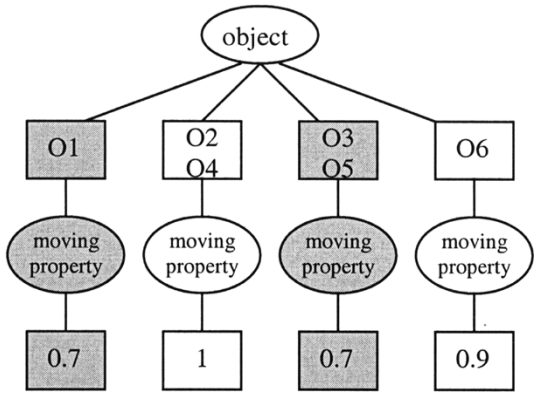


Fig. 19. Output of Query 3 (parts colored in gray are not to be considered).

As an example, let us now provide a ΣQL query translation for the σ-query discussed in Section V. We want to retrieve all the moving vehicles detected simultaneously by a laser radar and a video camera. The query is made of three parts. The fist part inserts into *vehicles* all the objects identified as vehicles by first clustering the data from a laser radar along the time dimension and then clustering the result along the object_3d dimension in the time interval between t_in and t_out.

```
INSERT INTO vehicles
{
SELECT type
CLUSTER *
FROM
  {
  SELECT object_3d
  CLUSTER {object_3d: PUBLIC *}
  FROM
    {
    SELECT time
    CLUSTER {time: PUBLIC * ALIAS T}
    FROM laser_radar
    WHERE T > t_in AND T < t_fin
    }
  }
WHERE type = 'vehicle'
}
```

In the second part, the inner subquery selects a video frame every ten frames in the same interval t_in and t_out. From each of these frames all the objects are retrieved and, successively, with a change of representation, the *moving* status of each object is made explicit, i.e., the attribute *status* is added to each object. Again, an operator such as show_state must exists in the Meta-Database in order to make the recognition of a moving object possible. Finally, all the resulting moving objects are inserted in *moving_objects*.

```
INSERT INTO moving_objects
{
SELECT state
CLUSTER *
FROM
```

```
    {
    SELECT object
    CLUSTER: show_state: {object: PUBLIC
  *}
    FROM
    {
    SELECT time
    CLUSTER {time: PUBLIC * ALIAS T}
    FROM video
    WHERE T mod 10 = 0 AND T > t_in AND T <
  t_fin
    }
  }
WHERE state ='moving'
}
```

The last part of the query simply applies the MERGE_AND construct to the results of the previous parts. Such operation will finally retrieve only the moving vehicles detected by both the laser radar and the video camera in the same time interval.

MERGE_AND

FROM vehicles, moving_object

This application of MERGE_AND is legal since *vehicles* stores a set of objects with attributes type, time, x, y, and z, while x, y and characterize the position of the object in the space. On the other hand, *moving_objects* stores a set of objects with attributes state, time, x and y. The attributes involved that can be *joined* are then time, x and y. It should be noted that the dimensions and directions, occurring in the original σ-query, are not considered here for the sake of clarity.

XII. Discussion

In this paper, we described how to carry out sensor data fusion from multiple sensors. In our approach, the queries are manually created, and then modified, to deal with the lack of information from a certain source or sources, and therefore not only the constraints can be changed, but also the source(s).

An experimental ΣQL query processing system has been implemented by researchers at the University of Pittsburgh, the University of Salerno, and the Swedish Defence Research Agency, to demonstrate the feasibility of applying the proposed techniques to data from three types of sensors, including laser radar, infrared video (similar to video but generated at 60 frames/s) and CCD digital camera. The users have successfully tested a number of queries, ranging from simple queries to complex ones for fusion, and systematic usability study is currently being conducted. Having established the feasibility of the techniques, we now discuss a number of issues for further research.

The sensors in the above experiment are limited to the three prespecified types of image sensors. To handle a large number of different sensors, we propose the following extension [5], [24]: the characteristics, applicable ranges, and processing algorithms of these sensors are stored in a knowledge base, which enables the system to deal with new sensors. The incorporation of domain-specific information into the knowledge base makes this approach extendible to other multimedia applications.

The fusion method is based on a method that is replaceable by other methods. Examples of other fusion methods that can be used are Basian networks [10] and Dempster–Schafer [23]. The proposed information structure is an information flow structure that works in parallel with the queries and allows acquisition and determination of the information necessary to carry out the sensor data fusion process. It is not only necessary to determine the objects, their state variables, and attributes that are requested by the query but also the belief values associated to them. This will put a heavy burden on the user to judge the result of the queries with respect to the belief values returned by the query system based on the uncertainty of the sensor information, because there will always be uncertainties in data registered by any sensor. How to replace the manual query refinement process by a semi-automatic or fully automatic query refinement process is of great importance from a user's point of view and will be further investigated.

Regarding the issue of generality of the ΣQL language, it is at least as powerful as SQL because an SQL query can be regarded as an ΣQL query with the clause "CLUSTER$*$." Since ΣQL can express both spatial and temporal constraints individually using the SELECT/CLUSTER construct and nested subqueries, and sensor data sources are by nature spatial/temporal, there is a good fit. Its limitation is that constraints simultaneously involving space and time cannot be easily expressed, unless embedded in the WHERE clause. Although such constraints may be infrequent in practical applications, further investigation is needed in order to deal with such complex constraints.

Finally, the qualitative methods used by the σ-operators are developed to support indexing and efficient inference making by transforming the information acquired from the heterogeneous data sources into a unified spatial/temporal structure. Such a unified structure is desirable because generic reasoning techniques can be applied independently of the original sensor data structures. Thus, generic σ-operators based on qualitative methods can be designed and implemented to support qualitative structure such as Symbolic Projection, which is discussed further in [4] where a number of alternative qualitative approaches can also be found.

References

[1] J. Baumann *et al.*, "Mole—concepts of a mobile agent system," *World Wide Web*, vol. 1, no. 3, pp. 123–137, 1998.

[2] C. Baumer, "Grasshopper—a universal agent platform based on MASIF and FIPA standards," in *First Int. Workshop on Mobile Agents for Telecommunication Applications (MATA'99)*, Ottawa, ON, Canada, Oct. 1999, pp. 1–18.

[3] K. Chakrabarti, K. Porkaew, and S. Mehrotra, "Efficient query refinement in multimedia databases," in *16th Int. Conf. Data Engineering*, San Diego, CA, Feb. 28–Mar. 3, 2000.

[4] S. K. Chang and E. Jungert, *Symbolic Projection for Image Information Retrieval and Spatial Reasoning*. London, U.K.: Academic, 1996.

[5] S. K. Chang, G. Costagliola, and E. Jungert, "Multi-sensor information fusion by query refinement," in *Proc. 5th Int. Conf. Visual Information Systems (Visual'02)*, Hsinchu, Taiwan, R.O.C., Mar. 2002.

[6] S. K. Chang and E. Jungert, "A spatial/temporal query language for multiple data sources in a heterogeneous information system environment," *Int. J. Cooperative Inform. Syst. (IJCIS)*, vol. 7, no. 2 & 3, pp. 167–186, 1998.

[7] C.-Y. Chong, S. Mori, K.-C. Chang, and W. H. Baker, "Architectures and algorithms for track association and fusion," in *Proc. Fusion'99*, Sunnyvale, CA, July 6–8, 1999, pp. 239–246.

[8] G. Grafe, "Query evaluation techniques for large databases," *ACM Comput. Surv.*, vol. 25, no. 2, June 1993.

[9] F. V. Jensen, *An Introduction to Bayesian Networks*. New York: Springer Verlag, 1996.
[10] E. Jungert, "A data fusion concept for a query language for multiple data sources," in *Proc. 3rd Int. Conf. Information Fusion (FUSION 2000)*, Paris, France, July 10–13, 2000.
[11] ——, "A qualitative approach to reasoning about objects in motion based on symbolic projection," in *Proc. Conf. Multimedia Databases and Image Communication (MDIC'99)*, Salerno, Italy, Oct. 4–5, 1999.
[12] ——, "An information fusion system for object classification and decision support using multiple heterogeneous data sources," in *Proc. 2nd Int. Conf. Information Fusion (Fusion'99)*, Sunnyvale, CA, U.S.A, July 6–8, 1999.
[13] E. Jungert, U. Söderman, S. Ahlberg, P. Hörling, F. Lantz, and G. Neider, "Generation of high resolution terrain elevation models for synthetic environments using laser-radar data," *Proc. SPIE, Model., Simul. Visualiz. Real Virtual Environ.*, vol. 3694, , pp. 12–20, April 7–8, 1999.
[14] L. A. Klein, "A boolean algebra approach to multiple sensor voting fusion," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 29, no. 2, pp. 317–327, Apr. 1993.
[15] H. Kosch, M. Doller, and L. Boszormenyi, "Content-based indexing and retrieval supported by mobile agent technology," in *Multimedia Databases and Image Communication*, M. Tucci, Ed. Berlin, Germany: Springer-Verlag, 2001, pp. 152–166.
[16] D. B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*. Reading, MA: Addison-Wesley, 1999.
[17] S. Y. Lee and F. J. Hsu, "Spatial reasoning and similarity retrieval of images using 2D C-string knowledge representation," *Pattern Recognit.*, vol. 25, no. 3, pp. 305–318, 1992.
[18] J. R. Parker, "Multiple sensors, voting methods and target value analysis," *Proc. SPIE Signal Processing, Sensor Fusion and Target Recognition VI,*, vol. 3720, pp. 330–335, April 1999.
[19] M. Stonebraker, "Implementation of integrity constraints and views by query modification," *SIGMOD*, 1975 PAGES???.
[20] B. Vélez, R. Weiss, M. A. Sheldon, and D. K. Gifford, "Fast and effective query refinement," in *Proc. 20th ACM Conf. Research and Development in Information Retrieval (SIGIR97)*, Philadelphia, PA, July 1997.
[21] E. Waltz and J. Llinas, *Multisensor Data Fusion*. Boston, MA: Artech House, 1990.
[22] F. E. White, "Managing data fusion systems in joint and coalition warfare," in *Proc. EuroFusion98—Int. Conf. Data Fusion*, Great Malvern, U.K., Oct. 1998, pp. 49–52.
[23] F. E. Yager, F. E. Fedrizzi, and F. E. Kacprzyk, Eds., *Advances in Dempster-Shafer Theory of Evidence*. New York: Wiley, 1994.
[24] S. K. Chang, G. Costagliola, and E. Jungert, "Multi-sensor information fusion by query refinement," in *Lecture Notes in Computer Science LNCS*. Heidelberg, Germany: Springer-Verlag, Mar. 2002, pp. 1–11.
[25] T. Horney, E. Jungert, and M. Folkesson, "An ontology controlled data fusion process for a query language," in *Proc. 6th Int. Conf. Information Fusion*. Cairns, Australia, July. 8-11, 2003, pp. 530–537.

Shi-Kuo Chang (F'86) received the B.S. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1965, and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1967 and 1969, respectively.

He was a Research Scientist at the IBM T. J. Watson Research Center, Yorktown Heights, NY, from 1969 to 1975. From 1975 to 1982, he was Associate Professor and then Professor with the Department of Information Engineering, University of Illinois at Chicago. From 1982 to 1986, he was Professor and Chairman of the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago. From 1986 to 1991, he was Professor and Chairman of the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA. He is currently Professor and Director of Center for Parallel, Distributed and Intelligent Systems, University of Pittsburgh. His research interests include distributed systems, image information systems, visual languages, and multimedia communications. He has published over 240 and wrote or edited 14 books. His books *Principles of Pictorial Information Systems Design* (Englewood Cliffs, NJ: Prentice-Hall, 1989), *Principles of Visual Programming Systems* (Englewood Cliffs, NJ: Prentice-Hall, 1990), *Symbolic Projection for Image Information Retrieval and Spatial Reasoning* (New York: Academic, 1966), and *Multimedia Software Engineering* (Norwell, MA: Kluwer, 2000), are pioneering advanced textbooks in these research areas. He is the Editor-in-Chief of the *Journal of Visual Languages and Computing* published by Academic Press, and the Editor-in-Chief of the *International Journal of Software Engineering & Knowledge Engineering* published by World Scientific Press.

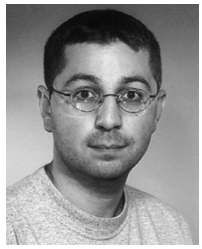
Gennaro Costagliola (M'95) received the Laurea degree in computer science from the University of Salerno, Italy, in 1987, and the M.S. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, in 1991.

He is currently Professor and Director of the Laurea degree courses in computer science at the University of Salerno. His research interests include programming languages, visual languages, parsing technologies, multimedia databases, web technologies.

Dr. Costagliola was guest coeditor of the February 2002 Special Issue of on Querying Multiple Data Sources for the *Journal of Visual Languages and Computing*. He is a member of the ACM and the IEEE Computer Society.

Erland Jungert received the Ph.D. degree in computer science from the University of Linköping, Linköping, Sweden, in 1980.

He has been a Research Staff Member at the Swedish Defence Research Agency (FOI), Linköping, since 1980, except for a short period in 1985–1986, when he was a Visiting Professor at the Illinois Institute of Technology, Chicago. Since 1987, he has been Director of computer science research at FOI and, since 1997, he is also a part-time professor at the Computer Science Department, University of Linköping. His current research interests include query languages, methods for qualitative spatial reasoning and various aspects of GIS. He is a coauthor of one book on spatial reasoning and a co-editor of two other books on visual languages and spatial reasoning. He is also an associate editor of the *Journal of Visual Languages and Computing*.



Francesco Orciuoli was born in Salerno, Italy, on April 6, 1975. He received the Laurea degree in computer science (cum laude) from the University of Salerno, Italy, in 1999.

He is a Research Contractor with the Department of Computer Engineering and Applied Mathematics (DIIMA), University of Salerno), in the area of distributed object infrastructures. His fields of interest are component-based design, compiler building techniques, design patterns, object oriented analysis, design and implementation, relational databases, web services, and UML.

Appendix C

A Visual Query Language for Uncertain Spatial and Temporal data

Visual Information System, Amsterdam, July 5, 2005.

Silvervarg, K., Jungert, E.

A Visual Query Language for Uncertain Spatial and Temporal data

Karin Silvervarg, Erland Jungert
FOI, (Swedish Defence Research Agency)
Box 1165, S-581 11 Linköping, Sweden

{karin, jungert}@foi.se

Abstract. Query languages for sensor data will have similarities with traditional query languages but will also have diverging properties that cause a higher complexity than the traditional ones. Both types require data independence. However, as different sensors create data of heterogeneous type the commonly used methods for data selection cannot be used. Furthermore, sensor data will always be associated with uncertainties and since also sensor data fusion must be possible to carry out this cause further problem in development of the query languages. Here a visual query language for sensor data input is discussed from these perspectives to allow a complete set of spatial temporal queries by means of its visual user interface.

1 Introduction

Query languages intended for multiple sensor data sources and other types of external data sources such as text messages differ in many ways from conventional query languages. A characteristic of this type of systems, which include multiple sensor data sources, is that they generally are more complex than traditional query systems. In particular, they will in most applications, be concerned with queries of spatial/temporal type. It will come from several different sources, but one important type of source is sensor data. Such data sources must be able to handle information that in various ways is uncertain. There are several sources of uncertainties in sensor data; one is due to limitations in the sensors and the navigation systems of the platforms carrying the sensors; another depends on the resolution of the sensor data; and a third reason is that some sensor types are sensitive to weather conditions such as snow or rain. All types of uncertainties will consequently have effects on the way the queries are executed. Another consequence when multiple data sources are involved is that in those cases fusion [16] of data must be possible to handle. Sensor data fusion becomes specifically complicated since the sensor data mostly are of heterogeneous type, i.e. sensor data are of different types.

Other problems that appear concern the selection of the data sources. Sensor technologies are constantly developing and for this reason it becomes almost impossible for a user to have sufficient knowledge about the capabilities of all occurring sensors. Thus the selection of data sources should be carried out by the query system independently of the users, for instance by using ontology [21]. This will result in a system that from a user's perspective is sensor and sensor data independent [22], [8]. Another motivation for this approach is to delimit the workload of the users and let them concentrate on their primary activities. In this way they do not need to have any particular technical knowledge on sensors and sensor data. It should be enough for a user to have a general under-

standing of the problems associated with the sensor data e.g. due to the uncertainties the sensors cannot correctly measure all possible attributes of the sensor data.

The work discussed here, which is an extension of [26], is focusing on a visual user interface of a query language for multiple sensor data sources. Clearly, many of the characteristics mentioned above will have an impact on such a user interface; among them sensor data independence. To achieve sensor data independence the system cannot and should not communicate concepts related to the sensors to the users. Instead, the system should work with concepts relevant and familiar to the users. This is along with tradition of traditional query systems where data independence plays an important role.

A large number of applications for query languages for sensor data fusion can be foreseen. Among these are applications where the query language is integrated to a command and control system for military applications and for crisis/emergency management systems but other less complex applications exist as well.

In this paper we only look at the problems concerned with specifying the query. How to present the result of the query is also an important problem, but that is a different problem, and we will not discuss that further in this paper.

Among the related works that should be pointed out are the work by Abdelmoty and El-Geresy [1], who have designed a system for graphical queries that is a filter based approach. It is primarily designed for spatial queries with similarities to ΣQL. Malan et al [24] works with data that in uncertain in the temporal aspect, in their case searching old African art where the dating is imprecise. Other approaches that focus on temporal queries of varying complexity are [17], [12], [11], [15] and [14]. Chang [6] has made an approach to use c-gestures for making spatial/temporal queries in what he calls a sentient map. Bonhomme [5] have proposed a visual representation for spatial/temporal queries based on a query-by example approach and related variation of this, called query-by-sketch, is presented in [4]. Hirzalla and Karmouch [19] have a more direct query approach to spatial/temporal queries, but they treat the spatial and the temporal parts separately and not together.

This paper is structured as follows. In section 2 the problem definition is presented and discussed. After this follows a presentation of the ΣQL query language in section 3, which forms the bases of the work presented here. In section 4 some of the basic elements that are present in the query language are presented. In sections 5 and 6 are the aspects of spatial and temporal queries discussed. After this follows in section 7 a discussion of the aspects of uncertainty. The section on uncertainty is followed by a section including a set of examples illustrating Visual ΣQL. Furthermore, there is also a discussion of the problem of completeness of the queries in section 9. Finally, conclusions from the work are given in section 10.

2 Problem definition

The main focus of the query language described in this work is concerned with moving ground based objects that correspond to various types of vehicles. The sensor may either be airborne, e.g. UAVs, aircrafts, or satellites. As these objects may be moving from time to time the sensor system must be able to detect and classify these objects. The consequence of this is that the query system primarily must be designed to respond

to spatial/temporal queries where uncertainties in the data must be taken into consideration.

Given the above background the main problem in this work has been to develop a *visual user interface* for a query language where spatial/temporal queries are in focus.

Generally, a set of elements can be associated with the queries. These elements relates to *where?, when?* and *what?*, that is where? corresponds *the area of interest* (AOI), when? to the *time-interval of interest* (IOI) and finally what? to the object type(s) that is asked for. These three elements are basically part of most, trivial as well as complex, queries since it is obvious that if we ask for a particular object (vehicle) then we must also consider a particular area where the object can be found but also a certain time interval during which it can be found in that area.

A further problem is associated with the visualization of the query results. To be considered in connection to this are again the aspects of uncertainty and the context in which the query, as well as the result, are directed. The first one of these aspects cannot be avoided because of the limitations of the sensors and the sensor platforms. The context is concerned with the geographical background of the requested objects. Thus the context can be demonstrated by means of geographical map information, corresponding to traditional map objects, such as roads, forests, lakes etc. Clearly, the map information is not only required for visualization of the query result but also for representation of the area of interest in the queries as will be demonstrated subsequently.

3 The query language, ΣQL

The query language discussed in this work is called ΣQL [1], [10], [7]. Originally, ΣQL was developed as a query language for querying of a single sensor image at a time. However, later it has evolved into a query language for multiple sensor data sources with capabilities for sensor data fusion and sensor data independence. The various sensors may be of different types and generates heterogeneous sensor data images. For this reason a large number of algorithms [20] for sensor data analysis must be available and administered by the system.

It is required that selection of sensors and algorithms must be carried out autonomously and for this reason means for such selection must be available. In ΣQL this is controlled by the ontological knowledge based system [21], [22]. The reason for the autonomous selection is to allow sensor data independence. Sensor data independence is motivated for a large number of reasons. One reason, which already has been mentioned, is to allow the user to concentrate on the work at hand without any knowledge of the sensors and their data types. Another motivation is to make repetitive queries possible without interference from the users. In this way light and weather conditions may, for instance, change during the period when the query is repeated across a specified area of interest resulting in different selections of sensors.

This query language allows classification of objects not only from the sensor data sources; it also allows cuing (detection) of possible candidates as a first step towards classification. Sensors used for cuing can, for example, be a ground senor networks or a synthetic aperture radar (SAR), while sensors for classification can be IR/camera, laser/radar and CCD/camera. The distinction between these two classes of sensors is that

the former covers a much larger area than the latter. Thus the former can be used to quickly search through the AOI for object candidates much faster than the sensors used for classification. They can also determine a much smaller search area for the classification sensors. Consequently, this depends on differences in coverage and resolution, i.e. the classification sensors have a low coverage and a high resolution opposite to the cuing sensors.

The basic functionality of the query language can be described as follows. A query is inserted by the user and then the input is fed in to the dependency tree generator which in a dialogue with the ontological knowledge structure generates a set of nodes in the dependency tree. As long as there are nodes in this tree, new sub-queries can be built (one for each of the selected sensors), refined and executed by the query processor. Once the sub-queries have been executed instances are created in the multilevel view database to support query refinement in a subsequent step. As new sets of dependency tree nodes are generated new sub queries can be executed and their result refined. This goes on until the dependency tree becomes empty, i.e. when there is no more sensor data available to process. In the final step data fusion is, if applicable, carried out using the result of the sub queries as input and then the process terminates. This process is further discussed in [8].

Sensor data fusion [22] is another property of ΣQL. It is quite unique and does not occur in traditional query systems. The motivation for sensor data fusion is to allow information from different sensors to support identification, but also to complement the information since different sensors can register different things; for instance a CCD camera might see that a car is blue, while an IR camera might see that the engine is running.

A serious question is how to interpret the fused result of a query. The approach taken here has been to associate a *belief value* to each result from the various sensors that are used in a query. Belief values are determined by the involved sensor data analysis algorithms and a common value is determined by the fusion process, which also is forwarded to the user as a part of the query result. All belief values are normalized and a high value means that a user may have a higher confidence in the result. It is not only to facilitate the fusion, but also to give the user a sense of how strong belief or confidence he can have in the result. This is an important aspect of the system that has been introduced to give the users a higher degree of trust in the query result as well as in the query system.

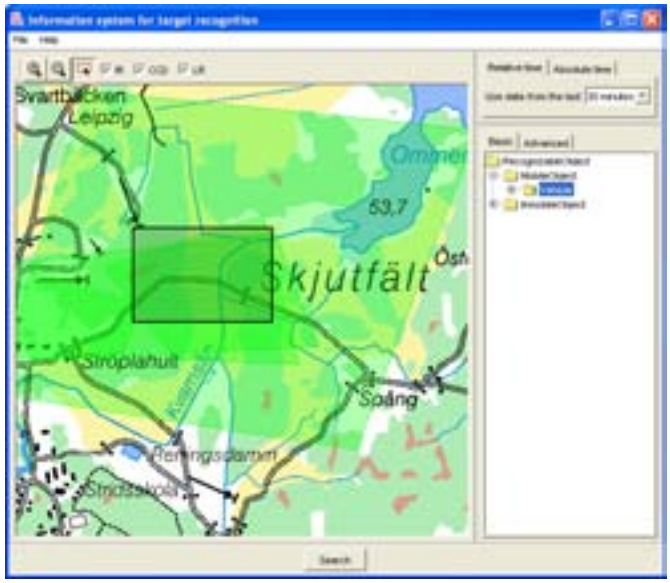


Figure 1 Simple selection of AOI, IOI and object type.

4 Basic Query elements

From a logical representation of ΣQL we have developed a visual language, called Visual ΣQL [25][26]. The basic questions to be considered are *where?*, *when?* and *what?*. We have chosen to let the user mark the AOI in a map and IOI, in its simplest form, is set by the start and end points in time, see figure 1 in the upper right corner. Queries can be repeated over time, i.e. the same query can be applied several times but with different IOI. Object types can in their simplest form be chosen from a list, but the user often has other requirements on the result. He may not be looking for all vehicles, but only for all moving vehicles, or all vehicles moving along a certain road. Consequently, a way to select the object types and put them in relation to each other is required. The solution is to use a structure with both object types and relations. There are many different approaches to this, one is query-by-example [5], and another quite similar is query-by-sketch [4]. We have chosen the approach of dataflow [18]. The object types are simply selected from a list, and eventually the user can apply relations to these objects, thus putting restrictions on the query result.

The user interface is built up around a work area and palettes. The work area is the space where the object types and the relations are placed and set in relation to each other by using a visual approach. The palettes contain the object types and the relations, organized according to attribute types, to be easily navigated.

Objects correspond to all kinds of object types that can be found by the sensors although in our application ground vehicles are of primary interest. The objects have both properties that are static, e.g. type of vehicle and color, but they also have properties that

may vary over time e.g. velocity and orientation. Objects can also be of geographical type e.g. roads, towns. An ontology containing a hierarchy of the objects is available as well making it possible to say that a car is a vehicle etc. [21]. The ontology describes the properties of the different objects. A car may, for instance, have attributes of type orientation, color, size, position, velocity, while a road may have speed restrictions, pavement and position. When the user selects an object type and places it in the workspace it is visualized as a rectangular box with the object type written inside, see e.g. figure 2.

As all objects are assumed to be linked to various relations there has to be ways to specify the details of the query. Once the type of objects is set together with the relations, the relations delimit the answer to include just those objects for which the relations are true. The relations can be either unary or binary. Unary relations are for example “color equals blue” or “velocity is greater than 50 km/h”. The binary relations can be undirected or directed. Directed means that the order of the involved objects matters, for instance the relation “before” gives a different result depending on the order of the involved objects, whereas the result of “equals” does not.

The user can select the relations from a palette. The relations are also visualized as boxes, where the possible relation is illustrated with an icon rather than by text, since it is often simpler for a human to grasp an icon [5]. To distinguish the relations further from the objects they also have a different color at the edge of the box, see figure 2.

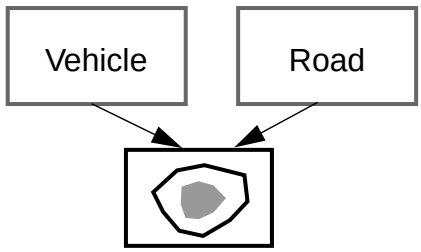


Figure 2 The user has selected the object-types vehicle and road, and the relation *inside*.

Objects and relations are connected to each other with arrows. Everything passed between a pair of such “boxes” is represented as a set of tuples. Output from an object box corresponds to a tuple of size one that simply contains an item from the set described in the object; e.g. a vehicle. In a binary relation two different tuples are related to each other and the resulting output tuples may contain more than one element.

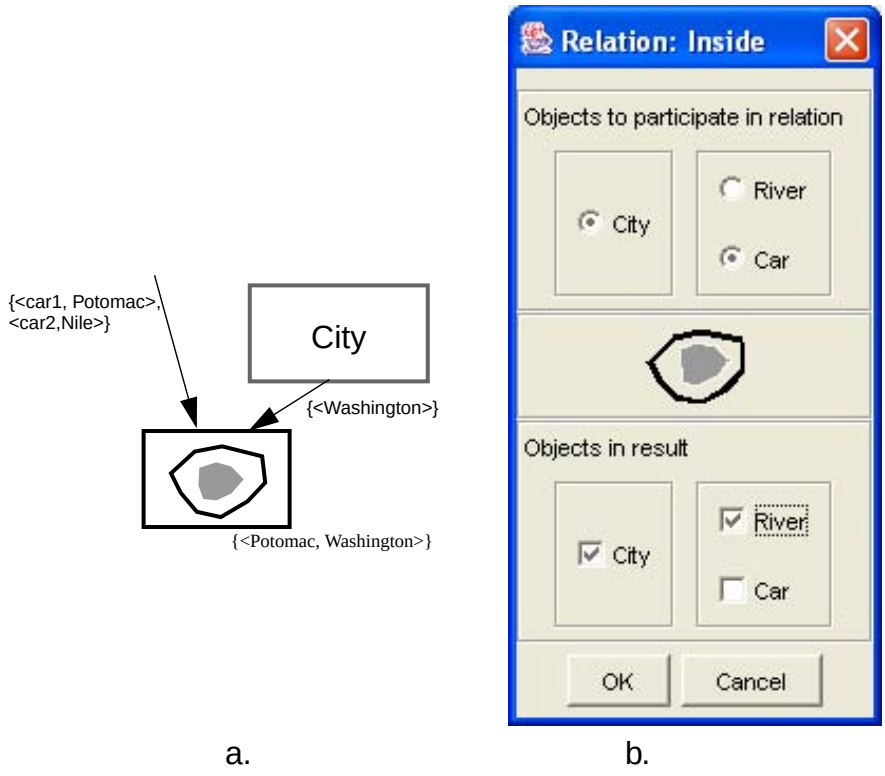


Figure 3 a. A relation where one of the tuples has more than one element and where the result contains only a part of all possible elements.
b. Settings of that relation.

If an input tuple contains more than one element the user has to define which of the elements in the tuple that is part of the relation, see figure 3b. In the resulting tuple the user can choose to include any parts from the input tuples. For example, in figure 3a we have the relation *inside*. One of the participating sets contains cities and the other could be a result of a relation relating the cars to the nearby rivers. In this case we could relate the cars to the cities to find out which of the cars that are in the city. In this case probably only car1. The resulting tuple from the relation can contain car, river, and city, or any subset thereof i.e. car and river; car and city; river and city; car; river; city. In figure 3a the resulting set is shown where the tuple consists of river and city.

The query language also includes some set operations, i.e. union, intersection and set difference. They are treated similarly to relations, but function a bit differently. The union operator simply produces a set containing all tuples from both incoming sets. The only restriction is that all the resulting tuples must have the same number of elements. To make it meaningful the user pairs elements from the incoming tuples. If some ele-

ments that are paired do not contain the same type of objects, the ontology is used to find the object that is closest above both objects in the object hierarchy to get the resulting type. This situation may occur in queries where objects of similar types are asked for. For instance, the incoming tuples may hold elements of the tuples <car, road> and <road, truck, river>; then the result may contain <vehicle, road>, which is more meaningful than other possible combinations.

Intersection is a bit different from union, because in intersection only a single element in each of the input tuples is chosen, just like in our normal relations. These objects are used to compute the intersection just like in normal set operations. The resulting tuple can include elements from all incoming tuples. If, for instance, <car, river> is intersected with <car, road> and car in both tuples is chosen for the intersection then the result could be <car, road, river>.

Set difference is similar to intersection in the aspect that only one element in the participating tuples are selected. Contrary to intersection set difference is a directed relation so all elements that are in the first, but not in the second tuple is kept as the result. Similarly to intersection the resulting tuple may contain elements from both participating tuples.

On all relations the *not* operator can be applied. Usually this means that all results for which the relation would be true is now false and vice versa, but it will work a bit different when dealing with uncertainties, see chapter 7 for further explanations. *Not* is visually denoted by drawing a line diagonally across the icon, see figure 4.

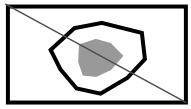


Figure 4 *Not* applied to the inside relation.

All types of relations can not be applied to all types of objects, for instance, it is normally meaningless to apply a color comparison to a city. To satisfy the need to know, which relations that can be applied to which type of object the ontology has been expanded. It also contains information about, the properties of each object type. This can be matched with the requirements the relations have on the attribute values the objects being related must have.

No relation is in any way selected to be the result of the query. Instead all “boxes” can be seen as the results or partial results. This will give the user a better opportunity to analyze the consequences of different relations. Another effect is that the workspace may contain several unrelated queries on the same objects, AOI and IOI.

5 Spatial queries

In [13], Egenhofer identifies eight atomic topological relations *disjunct*, *meet*, *equal*, *inside*, *coveredBy*, *contains*, *covers*, and *overlap*, which have different spatial relational properties. All these have been included in the query language as a basis for spatial que-

ries. We have also found needs for directional spatial relations i.e. *northOf*, *southOf*, *north-eastOf* etc. Similar to this we have also directional relations that are determined with respect to a local object or position, i.e. *inFrontOf*, *behind*, l*eftOf* etc.

All relations require different properties with respect to the objects to which they relate. For instance, *inFrontOf* requires that the position and the orientation of the object is known, while inside requires an extension in space. To make sure that the relations only operate on objects with the required relational properties the ontology also includes information about the type of properties that can be expected from each type of object. Then each relation has a set of requirement on the object properties. The spatial properties defined here are *hasExtension*, *hasPosition* and *hasOrientation*.

All of the mentioned spatial relations are binary. Sometimes there is a need to relate an object to a fixed point or area and for this purpose we have included the *spatial attribute entity* (SAE) that only has a position or area and no other properties. A SAE can be used together with all of the relations mentioned above to determine objects that relate to fixed points or areas. The only difference when relating an object to a SAE compared to when relating a pair of objects to each other is to determine the output tuple. Since a SAE is not an object it is not allowed to be included in the result.

6 Temporal queries

The classical work on temporal relations has been done by Allen [3]. He has defined 13 binary relations that concerns relating two time interval to each other. These relations are: *before*, *after*, *meets*, *metBy*, *overlaps*, *overlappedBy*, f*inishes*, *finishedBy*, *starts*, *startedBy*, *contains*, *during* and *equals*. We have chosen to use the same icons for these relations as those used by Hibino et al [17]. The only difference is that we have decided to use squares instead of circles, see figure 5; the reason for this is described in the section on uncertainties.

Figure 5 Visualization of Allen's 13 temporal relations.

In analogy to spatial queries there is also a need for relating objects to fixed points in time or time intervals. Thus the functionality for creation of a *temporal attribute entity* (TAE) has been included. If only a point in time is needed the start and end of the time

interval become equal. Consequently, Allen's relations [3] can be used to relate objects to fixed times as well.

Determination of tracks is a quite common task in most sensor data systems. To produce tracks by means of a query language from sensor data requires special attention. Thus, when general attribute relations are combined with either spatial or temporal relations then it is quite simple to determine tracks. However, when combining general attribute relations with both spatial and temporal relations then the system must keep track of which observations that fulfill both the time and spatial relations since otherwise simultaneousness may be lost.

The objects that are related by the temporal relations, just like the spatial relations, require certain properties. While the spatial relations have three different properties, the temporal relations have just a single one, i.e. *hasTime*. The reason for this is that time is one-dimensional while space, so far, in our system is 2D.

7 Uncertainties

As have been pointed, out sensor data are always associated with uncertainties that depends not only on the limitations in the sensors but also to limitations in the navigation systems, unless the sensors are at fixed positions. Obviously, these uncertainties will influence the result of the queries in more than one way; especially when dealing with extremely large data volumes and when the data from multiple data sources are part of the sensor data fusion process. It also turns out that from a user's perspective there are two aspects that need to be handled. That is, uncertainties in space and in time.

All spatial relations should be possible to apply in a mode where uncertainties will be considered. This is visually distinguished by representing the areas inside the icons by more fuzzy looking symbols. The effect of this is that a relation may be true with respect to the uncertainties but in reality with completely accurate data the relation may not be true. The advantage of this is that no crisp results will be missed. Take for instance the *inside* relation; here all areas that could be inside will be returned even though they also might be slightly outside, but with respect to uncertainties we can not tell for sure, which is the case.

Uncertainties in time are treated similarly. All relations have an uncertain mode, which is visualized by replacing the squares with circles, see figure Figure 6. There are two reasons for using squares in the accurate case and circles in the uncertain. One reason is because it is easier to associate the sharp corners of the squares with the actual data and the round shape of the circle with the more uncertain data. Another reason is that we believe that the users most often will use the uncertain mode of the relations and thus we will be using the same symbols as Hibino et al [17].

Figure 6 The relations before and start, where consideration will be taken to the uncertainties in the data when evaluating the relation.

The result of applying *not* to a relation that account for uncertainties is a bit different from when it does not. Usually, the complete inverse of the result would be kept, but when the uncertainties are considered the result will be a bit different. For example the *inside* relation for uncertainties will include all results that might just be inside. The *not inside* relation will include all results that might be outside, including some results that actually are inside, although it cannot be proven. Thus some results are part of both in the *inside* and in the *not inside* relation.

So far, no complete implementation of this part of the system exists. However, the actual implementation of the evaluation of relations concerned with uncertainties can be made in several ways. Several theories exist, one is rough sets [2] another is the egg yolk theory [23]. There is also some research being done on how to evaluate information with different levels of uncertainty [24]. The aspects of uncertainty of data must be studied further to be able to logically respond to the queries in a correct way and to give the users a better support. The starting point for such a study must be to regard uncertainty manipulation in the queries as the default case.

8 Completeness

The visual query language should be at least as powerful as the relational algebra. The operators of the relational algebra are *union*, *set difference*, *projection*, *selection* and *cartesian product* [27].

Union, is the set of tuples that are in R or S or both. Union in the relation algebra can only be applied to relations of the same arity. Union is directly implemented in our language, with the additional convenience to allow selection of subparts of the tuples to make them equal in size.

The *set difference* of *R* and S contains those tuples in *R* which are *not also* in S. Set difference just like union is implemented directly in Visual ΣQL.

The subset of R that contains all tuples that satisfies a given logical formula is a *selection* of R; this is the equivalent of selecting some of the rows in a table. In this visual query language this is carried out by the relations.

The idea behind *projection* is to remove some elements/components from a relation. If a relation is seen as a table then projection is the act of selecting a set of columns in that table. In this work this corresponds to the selection of which elements that should be included in the resulting tuple.

The *cartesian product* of R and S is the set of all tuples whose components forms a tuple in R and a tuple in S. Our binary relations are cartesian products with a *selection* and a *projection*. If the relation is set up to always be true then there is no selection, and if all elements are selected for the resulting tuple then there is no *projection*. The result is the pure cartesian product.

9 An example

To illustrate the use of Visual ΣQL some examples will be given. The examples can be seen as part of a simple scenario where the fundamental problem is to find different vehicles along or on a riverside express way passing an electrical power plant. There is also a second express way in the vicinity of the area of interest that is not interconnected to the other one. In this scenario there has been some incident directed towards the power plant. The user of the query language is trying to find, identify and track vehicles on the riverside express way where the vehicles may be engaged in further hostile activities directed towards the power plant. To monitor the area a set of ground based sensors have been deployed along the express way, that e.g. can read the license plates of the vehicles or from a top view can determine a set of other attributes such as speed and direction.

Query 1: Find all vehicles close to the power plant near in time of the incident.

The AOI corresponds to a relatively small area around the power plant and the IOI is the day of the incident. The visual representation of this query can be found in figure 7. The output of this query is saved by the user for further usage. The user also labels this information *suspect vehicles.*

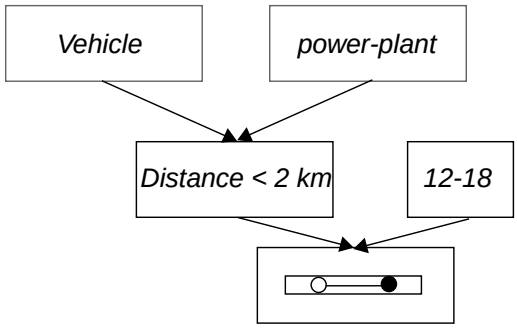


Figure 7 Finding suspect vehicles around the power plant (query #1).

The rational of this query is to try to identify all vehicles that have been in the area at the time of the incident and, which for this reason may be considered suspicious

Query 2: Find similar vehicles on the riverside freeway. (figure 8)

Here the AOI is covering a much larger area and the IOI could be the same as before. This query is visualized in figure 8. Here the term *similar* is determined by means of the object ontology. This and other analogous concepts are also discussed further in [9].

This query is motivated by the needs to find also other vehicles that may be connected to the vehicles that originally were found suspicious.

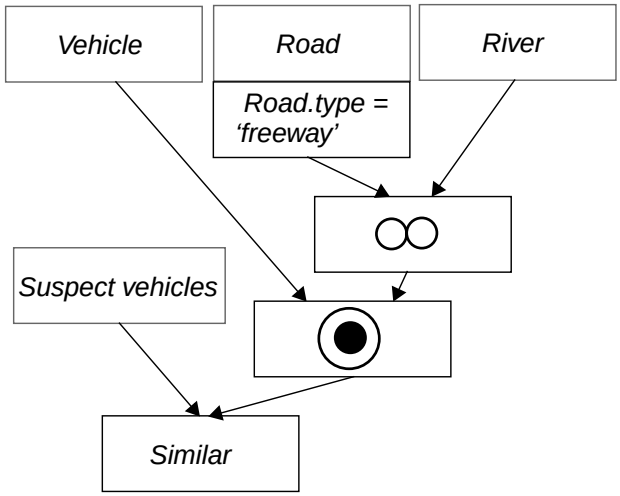


Figure 8 Find similar vehicles on the riverside freeway (query #2).

10 Conclusions

In this work the visual user interface of the query language ΣQL has been discussed together with some of its basic characteristics. The query language allows queries of spatial/temporal type where the input data are generated by sensors of various types. The applications here will be focusing on ground based targets (objects) that basically may be vehicles in a geographical context. The sensors may be both airborne and ground based. Data from these sensors are generally of heterogeneous type. The query system can handle uncertainties due to the sensor system and since multiple sensors may cover the area of interest a method for sensor data fusion has been developed and integrated. The set of possible spatial/temporal queries is complete from a traditional theoretical viewpoint. Particularly, the queries may be concerned with information that in time and space may be both absolute and relative. Relations from such queries are normally determined by means of a set of predefined operators.

A simple demonstrator of Visual ΣQL has been implemented and is gradually extended. Currently, five different sensor types have been integrated to the system among these are laser-radar, IR-camera and a CCD-camera but also an unattended ground based sensor network. In order to be able to handle more complicated and dynamic situations where quite large amounts of data must be handled the query processor will be hooked up to a simulation system.

Future research will focus on user tests of the visual user interface and on the adaptation of the query system to Internet applications. The approach taken for the selection of the sensor data in cases where the sensors are distributed on different platforms attached to the Internet will be based on sets of intelligent agents.

The focus of our current work is on how to specify queries. Future research also needs to find a satisfying solution on how to present the result of those queries. That solution has to solve the problem of displaying results that differ in time and location, and where uncertainties are present in all aspects of the result.

References

1. Abdelmoty, A. and El-Geresy, B., *Qualitative Tools to Support Visual Querying in Large Spatial Databases*, Proceedings of the workshop of Visual Language and Computing, Miami, Florida, September 24-26, 2003, pp 300-305.
2. Ahlqvist, O., Keukelaar, J. and Oukbir, K., *Rough and fuzzy geographical data integration.* International Journal of Geographical Information Science, 14(5):475-496, 2000.
3. Allen, J. F., *Maintaining knowledge about temporal intervals,* Communications of the ACM, vol. 26, no. 11, pp 832-843.
4. Blaser, A. D. and Egenhofer, M. J., *Visual tool for querying geographic databases*, Proceedings of the Workshop on Advanced Visual Interfaces, 2000, p 211-216
5. Bonhomme, C., Aufaure, M.-A. and Trépied, C., *Metaphors for Visual Querying of Spatio-Temporal Databases,* Advances in Visual Information Systems. 4th International Conference, VISUAL 2000. Proceedings (Lecture Notes in Computer Science Vol.1929), 2000, p 140-53
6. Chang, S.-K., *The sentient map*, Journal of Visual Languages and Computing, Vol 11, No. 4, August 2000, pp 455-474.
7. Chang, S.-K. and Jungert, E., *Query Languages for Multimedia Search*, In Principals of Visual Information Retrieval, M.S. Lew (Ed.), Springer Verlag, Berlin, 2001, pp 199-217.
8. Chang, S.-K., Costagliola, G., Jungert, E., *Multi-sensor Information Fusion by query Refinement*, Recent Advances in Visual information Systems, Lecture Notes in Computer Science, 2314, Springer Verlag, 2002, pp 1-11.
9. Chang, S.K., Jungert, E. *Iterative Information Fusion using a Reasoner for Objects with Uninformative Belief Values*, Proceedings of the seventh International Conference on Information Fusion, Stockholm, Sweden, June 30 - July 3, 2004.
10. Chang, S.-K., Costagliola, G., Jungert, E. and Orciuoli, F., *Querying Distributed Multimedia Databases and Data Sources for Sensor Data Fusion*, accepted for publication in the journal of IEEE transaction on Multimedia, 2004.
11. Chittaro, L. and Combi, C., *Visualizing queries on databases of temporal histories: new metaphors and their evaluation*, Data & Knowledge Engineering, v 44, n 2, Feb. 2003, p 239-64
12. Dionisio, J.D.N. and Cardenas, A.F., *MQuery: a visual query language for multimedia, timeline and simulation data*, Journal of Visual Languages and Computing, v 7, n 4, Dec. 1996, p 377-401
13. Egenhofer, M., *Deriving the combination of binary topological relations,* Journal of Visual languages and Computing, Vol 5, pp 133-49.
14. Erwig, M. and Schneider, M., *Spatio-temporal predicates*, IEEE Transactions on Knowledge and Data Engineering, v 14, n 4, July/August, 2002, p 881-901
15. Fernandes, S., Schiel, U. and Catarci, T., *Visual query operators for temporal databases*, Proceedings of the International Workshop on Temporal Representation and Reasoning, 1997, p 46-53

16. *Handbook of Multisensor Data Fusion*, D. L. Hall & J. Llinas (Eds.), CRC Press, New York, 2001.
17. Hibino, S. and Rundsteiner, E. A., *User Interface Evaluation of a Direct Manipulation Temporal Visual Query Language*, Proceedings ACM Multimedia 97, Seattle, WA, USA, 9-13 Nov. 1997, p 99-107.
18. Hils, D., "Visual Languages and Computing Survey: Data Flow Visual Programming Languages", *Journal of Visual Languages and Computing*, vol.3, 1992, pp.69-101.
19. Hirzalla, N. and Karmouch, A., *Multimedia query user interface*, Canadian Conference on Electrical and Computer Engineering, v 1, 1995, p 590-593.
20. Horney, T., Ahlberg, J., Jungert, E., Folkesson, M., Silvervarg, K., Lantz, F., Franssson, J., Grönwall, C., Klasén, L., Ulvklo, M., *An Information Sustem for target recognition*, Proceedings of the SPIE conference on defense and security, Orlando, Florida, April 12-16, 2004.
21. Horney, T., *Design of an ontological knowledge structure for a query language for multiple data sources*, FOI, scientific report, May 2002, FOI-R--0498--SE.
22. Horney, T., Jungert, E., Folkesson, M., *An Ontology Controlled Data Fusion Process for Query Language*, Proceedings of the International Conference on Information Fusion 2003 (Fusion'03), Cairns, Australia, July 8-11.
23. Lehmann, F. and Cohn, A. G., *The eggyolk reliability hierarchy: Semantic data integration using sorts with prototypes.* Proceedings of the third international conference on Information and knowledge management, ACM Press, 1995, pp 272-279.
24. Malan, K., Marsden, G. and Blake, E., *Visual query tools for uncertain spatio-temporal data*, Proceedings of the ACM International Multimedia Conference and Exhibition, n IV, 2001, p 522-524.
25. Silvervarg, K. and Jungert, E. Aspects of a visual user interface for spatial/temporal queries, Proceedings of the nineth International Conference on Distributed Multimedia Systems, Miami, Florida, September 24--26, 2003, pp 287-293.
26. Silvervarg, K. and Jungert, E., *Visual specification of spatial/temporal queries in a sensor data independent information system*, Proceedings of the tenth International Conference on Distributed Multimedia Systems, San Francisco, California, September 8-10, 2004, pp 263-268.
27. Ullman, J., Principles of Database and Knowledge - Base Systems, Volume 1. Computer science press, Rockville, 1988.

Appendix D

Uncertain topological relations for mobile point objects in terrain

Distributed multimedia Systems (DMS'05), Banff, Canada, Sept. 5-7, 2005.

Silvervarg, K., Jungert, E.

Uncertain topological relations for mobile point objects in terrain

Karin Silvervarg, Erland Jungert
Swedish Defence Research Agency
karin@foi.se, jungert@foi.se

ABSTRACT

This paper proposes a simplified solution to how uncertainties in relations can be handled when concerned with relatively small, mobile objects, for instance vehicles, in a spatial temporal database system. Incompleteness, inconsistency, vagueness, imprecision, and error both in the sensor data and from the processing of that data result in uncertainties. These uncertainties can be handled by assigning broad boundaries to the objects. There are models for how to handle broad boundaries of spatial object in the general case, but in this special case a lot of those relations are not applicable. Thus a simplified way to manage topological relations with respect to moving artifacts in the terrain is proposed. A solution for how to treat relative positioning between these objects is also proposed.

1. INTRODUCTION

Sensor data are gradually used more as input to a large variety of systems. Examples of such systems are command and control systems for both military and civilian purposes. In the latter case the applications focus may be concerned with emergency management systems. Other examples are robotics, safety and security systems. In many of these systems query languages are required. When designing query languages for sensor data sources a number of problems occur. These problems relates in part to the types of sensors used but also to those cases when multiple sensors are used to collect data more or less simultaneously. In the latter case methods for sensor data fusion will be required. As a consequence, query languages for multiple sensor data sources must be designed to fuse sensor data. The fusion techniques may be of different types. An example of a query language with an integrated sensor data facility is discussed in [6]. A particular problem that occurs in conjunction with the use of sensor data is the uncertainty in the data. The data uncertainties can be of different types, e.g. incompleteness, inconsistency, vagueness, imprecision. Basically, uncertainties in sensor data are generally due to imperfections in the sensors and in the sensor data analysis. These uncertainties must be handled properly by the query language. However, the way the uncertainties should be handled depends to a large extent on the class or classes of problems the queries should be applied to. The uncertainties have, in particular, effects on the queries applied to the spatial relations that occur between extended geographical object [3] or on spatial relations between man-made objects or man-made objects and extended objects where the latter can be part of the background context. A general approach to handle the uncertainties in these problems is to represent the object with what sometimes is called *a broad boundary.* The object thus has an uncertainty component that depends on extension and position. This representation includes an interior of the objects that definitively belong to the objects. The broad boundaries, on the other hand, include parts that are part of the object and parts that are not. Consequently, the broad boundary mirrors the uncertainties of the object. This way of representing the uncertainties are in many ways related to the theory of rough set [8]**.** Since the degree of uncertainty in sensor data primarily depends on the type of sensor this means that different sensors under different circumstance deliver uncertainties of different character, which has consequences on how the query language should handle the actual uncertainties. The solution to this problem will be further discussed subsequently in this paper for small man-made objects in relation to larger objects that are part of the background. The background is here sometimes called the context.

The general structure of this work can be described as follows. Chapter 2 introduces the objectives. Chapter 3 gives a brief overview of the query-system that is used. Chapters 4, 5 and 6 describe different issues concerning topological relations. Chapter 7 describes other relations that can be applied to mobile point objects. This is followed by the conclusions and directions for future work in chapter 8.

2. OBJECTIVES

The main objective of the work discussed in this here is to, by default; consider existing uncertainties in input data. The input data used in the query language discussed here are basically coming from various kinds of sensors. The sensors are generally of image generating types, such as IR, and laser-radar. The uncertainties in these sensors are for the most part due to imperfections in the sensors. As a consequence, when a query is applied to multiple sensors the uncertainties of each sensor must be taken into account when the query answers are determined and before the answers are delivered to the users. In earlier work the

uncertainties were just concerned with how certain the answer is with respect to the requested object type(s). This is by no means sufficient since the sensor data uncertainties will have consequences on other aspects of the queries as well. Among those aspects can uncertainty in position and speed of various entities be mentioned; where these aspects generally are called status variables as they quite often are subject to changes contrary to ordinary attributes. The focus of this work is to consider how these uncertainties affect a number of relations that will occur between the entities. These relations are generally of spatial type that concern man-made and background objects.

3. ΣQL

The query language that we use is called ΣQL [1]. The system [7] is divided into a visual user interface, a query processor and the sensor nodes to which the sensor data sources are attached. The query processor includes a knowledge system that operates in conjunction with an ontology. The purpose of this knowledge system is to support automatic selection of the sensors and sensor data algorithms for the data analysis that together deliver the information used to respond to a query. In a first set-up of the query processor the actual sensors have been a digital-camera, an infrared camera and a laser radar. However, the system is not limited to these three sensor types; others can, on demand, be attached as well. The sensor nodes include also means for target recognition. For this purpose a database containing a library of target models is attached to the ΣQL-processor. The target models stored in this library are used by the image analysis processes to recognize objects found in the sensor data inquired by the users. A meta-database containing the available information that has been registered by the sensors is also attached to the query processor. The query system includes, contrary to conventional query languages, a sensor data fusion module. The purpose of this module is primarily to fuse information extracted from the sensor data, which correspond to sub query results. These data emanate generally from multiple sensors whose sensor data altogether are of heterogeneous type.

The visual user interface is designed to allow both simple and complex queries of spatio/temporal type. The user indicates the area of interest (AOI) in a map and the actual time interval of the query and finally, through a part of the underlying ontology, determines the requested object type or types. In this approach of the query language basically just vehicles are requested. These vehicles can be selected through the visualization of the ontology.

For the time being, complex queries concern vehicles and different types of spatial and temporal relationships that may occur between the vehicles and between vehicles and background information [9]. Background information generally means geoinformation. The most important spatial relationships are *topological relations*, *directions* and *distances*. Queries that allow the combination of spatial and temporal conditions are possible as well. The queried objects are found in sensor data and include sets of attribute and status values. Associated with the object information is also a belief value, which indicates how much the system believes that a certain object is of a specific type. Thus a belief value is a quantification of the uncertainty associated with the sensors and the sensor data they produce. Consequently, the belief values serve a purpose but are by no means useful when determining which relations that may exist between the objects. Due to the uncertainties in the sensor data, these relations include uncertainties, which must be considered when the system responds to the queries. The expressions used are normally found in where-clauses in traditionally text oriented query languages contrary to the situation here where the interface is purely visual.

4. TOPOLOGICAL RELATIONS

Topological relations concerns the way in which two geometrical objects can relate to each other in two dimensions. In [4], Egenhofer identifies eight atomic topological relations for areas *disjoint contains, inside, equals, meet, covers*, *coveredBy*, and *overlap*, see figure 1. These relations assume that the exact sizes of the surfaces are known.

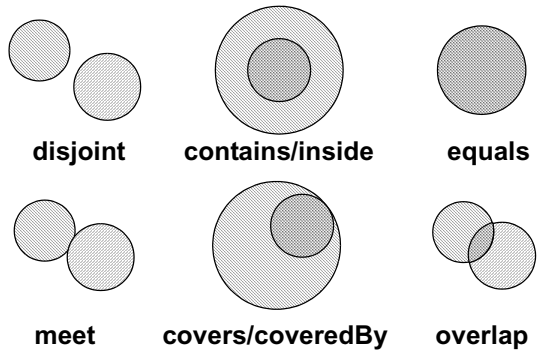


Figure 1. The eight topological relations. Contain/inside and covers/coveredBy are respectively two cases since it depends on which area is surrounding which.

When uncertainties are introduced these relations are not sufficient since the exact positions of the edges of the surfaces are not known. One way to handle this is to add a broad boundary [2]. Then the center of the area signifies the absolutely certain minimum area and the outer boundary signifies for instance the 80% certainty for the area. The resulting topological relations of the areas with broad boundaries are not eight but 44 or 52 [3] if we allow holes in the areas. However, in the work described here this number has been considerably reduced due the object

types that are subject to the study, that is in this case some obvious simplifications have been made, which will be discussed further subsequently.

5. TOPOLOGICAL RELATIONS BETWEEN POINT OBJECTS

In our system the focus is on mobile objects acquired from sensor data. Mobile objects can have uncertainties concerning both the size of the objects and their locations. These uncertainties could be handled with broad boundaries. A consequence of limiting ourselves to mobile objects is that the uncertainties in size are limited, as well. If we are able to identify the type of the objects, for example, car or truck, then the uncertainty in size becomes even more limited. The magnitude of the uncertainty in size is negligible compared to the possible uncertainty in position. Even the size of the objects can be neglected compared to the usual size of the uncertainty in position. Consequently, we can approximate the mobile object with only an area that equals the uncertainty in position. The topological relations between two areas are limited for the eight basic relations described in figure 1, but we should consider that since we are taking about mobile objects they cannot in practice overlap, not even a little. Thus the relations *contains, inside, equals, covers, coveredBy,* and *overlap* have no equivalence in reality for this type of objects and may for this reason be excluded. Mobile objects may only by close to each other. This is called the proximity, which is equivalent to the topological relations *meet, contains, inside, equals, covers, coveredBy,* and *overlap,* or they can be distant which corresponds to the relation *disjoint*. Thus these eight relations can be reduced to just two, see figure 2.

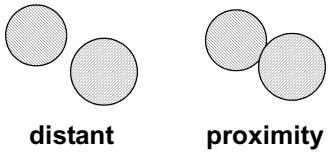


Figure 2. The only two relevant topological relations when concerned with mobile artifacts related to mobile artifacts.

6. TOPOLOGICAL RELATIONS BETWEEN POINT AND BACKGROUND OBJECTS

As was described in chapter 5 mobile objects with uncertain positions can be approximated with an area corresponding to the uncertainty in the position. Background objects, for instance forests, lakes, cities, on the other hand are better approximated with an area including a broad boundary. The kernel of the area corresponds to the part of that with certainty belongs to the object and the broad boundary corresponds to the uncertainty part. As described in section 4, the number of possible topological relations between two simple areas is eight according to Egenhofer [4]. If the areas have broad boundaries the number of possible relations is 44 according to Clementini et al [2].

Another aspect to take into consideration is that since the boundaries of the areas are based on uncertainty it does not seem to be realistic to include a relation that requires the borders to coincide exactly, like the classical relation *meet*, see figure 1.

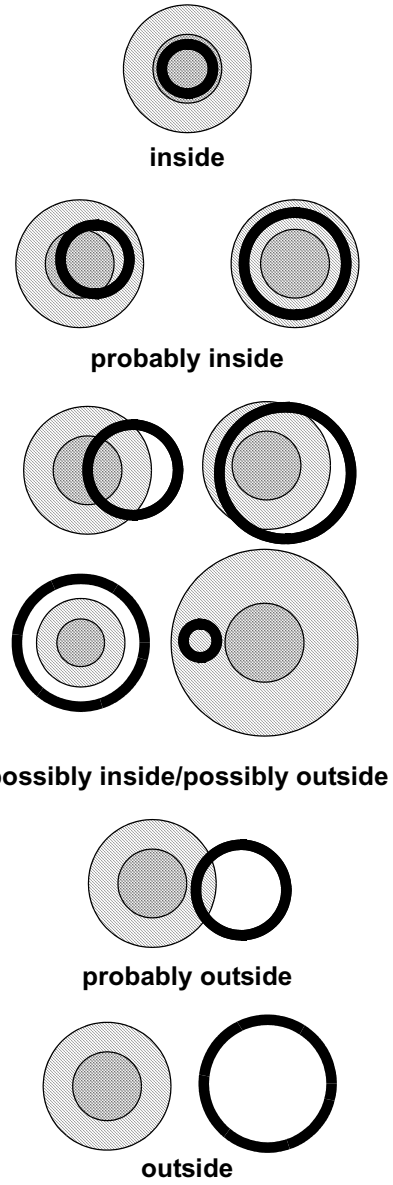


Figure 3. The nine topological relations between a simple area and an area with a broad boundary when the areas do not contain any holes.

The possible topological relations between an object with a simple area and an object with an area with a broad boundary can be expected to be a subset of the 44 relations given some limitations. These limitations can be formulated by two simple rules:

1. The point object is a simple area, while the background object has an area surrounded by a broad boundary.

2. The borders of the objects never coincide exactly.

When reviewing the 44 relations and removing all variants that break at least one of these two rules the results is the nine topological variations. Since the simple area is only an approximation of a point object it can never *cover* or *overlap*, etc. Thus the relations have been grouped into five groups; *inside*, *probably inside*, *possibly inside/possibly outside*, *probably outside* and *outside*, see figure 3. In effect the result is only 4 possible relations.

In [3] Clementini et al introduced relations between two area objects with broad boundaries that may have holes as well. This gives an additional eight relations. When evaluating those relations we add one rule to our set:

3. The point object may not have a hole in its area.

The review of these eight relations result in three remaining relations that are applicable to our case. All three of these relations can be put into the existing group *possibly inside/possibly outside*, see figure 4.

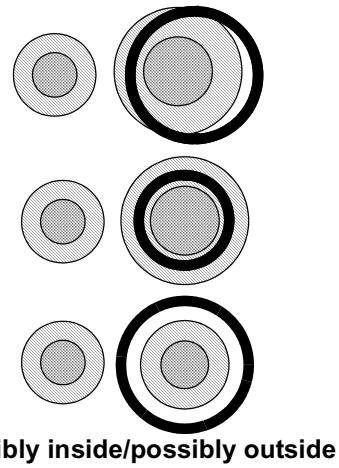


Figure 4. The three additional topological relations between a simple area and an area with a broad boundary containing holes.

7. OTHER POINT OBJECT RELATIONS

Determination of the position of an object based on one or more sensor data sources will always be associated with some types of uncertainty, thus we can never be sure that the given coordinate values are correct. For this reason, a measure of the uncertainty must be available. Usually, for each given position the area of uncertainty is normally represented with an ellipse that sometimes is generalized into a circle. The size and shape of this area depends generally on the sensor type. However, in a query language, which has to respond quickly to the various queries such uncertainty areas are unpractical. For this reason the areas of uncertainty must be replaced with something more efficient. In this work, we have chosen to describe the location uncertainty of a point object by a rhombus. This is motivated not just by its usefulness as a descriptive structure of the position uncertainty but also because it is a convenient way to determine object relations of the types that are discussed in this work. Furthermore, it is also useful, for the determination of directions between a pair of objects, where the secondary rhombus can be found in relationship to the primary one, as the area where the direction cannot be sufficiently correct settled may be vary depending on the uncertain location. The use of such a structure for determination of directions is quite common and variations of it have been used during a long time. Figure 5 shows the orientation of the rhombus and its directional structure relative to a local coordinate system. In 5a the directions are global (north, ...) while in figure 5b the direction are just local (in_front_of, ...) to the object. The area covered by the rhombus is called the *proximity* of the object. Observe also, for instance, that here the direction of a point between *north* and *northeast* is described by the interval [*north, northeast*] or alternatively if the interval is open as]*north, northeast*[. Again this way of describing direction is due to the presence of uncertainties. Thus it cannot be said that the direction of an object is just north, no matter if the given coordinates says so. Instead, again because of the uncertainty of the position, it is]*northwest, northeast*[. Consequently. if the direction of an object relative another one should be determined then the corresponding directional interval should be determined. However, if the object or at least a part of it falls inside the proximity then the direction cannot be determined and for the distance we can just say that the distance between the two objects are very close. This is simply expressed by saying that the objects are in the proximity of each other.

The area of proximity is described by its four corner points by means of its local coordinate system:

$(\varepsilon,0)(0,\varepsilon)(-\varepsilon,0)(0,-\varepsilon)$

It is also easy to see that

$\varepsilon = |\Delta x| + |\Delta y|$

determines the edge of the proximity area.

Here ε is depends on the maximum positional error, i.e.:

$$maxerror = \varepsilon/(\sqrt{2})$$

A consequence of this is that the rhombus is somewhat larger than the actual area of uncertainty but this is negligible since the difference does not contribute much to the positional error. After all, other types of uncertainty areas are approximations as well.

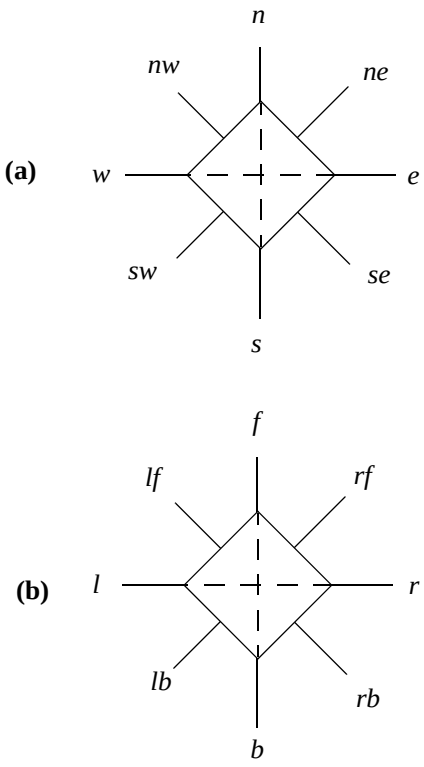


Figure 5. The rhombus proximity representing an object location and its area of uncertainty including also the directions outside that area for both global and local directions.

Determination of the direction of an object relative to another object, both with uncertain positions, is quite trivial and does not require any heavy calculations and can be illustrated by the following cases in terms of rules. An object in the open interval between north and north-east is delimited by the rhombus, the y-axis and the ne-line, i.e.:

If $|\Delta x| + |\Delta y| > \varepsilon$ and $|\Delta x| > 0$ and $|\Delta y| > |\Delta x|$ then] *n, ne* [

If instead the interval is closed we get

If $|\Delta x| + |\Delta y| >= \varepsilon$ and $|\Delta x| > 0$ and $|\Delta y| >= |\Delta x|$ then [*n, ne*]

For an object with the direction *north of* the result is delimited with the edge of the rhombus and the nw and the ne lines, i.e.

If $|\Delta x| + |\Delta y| > \varepsilon$ and $|\Delta y| > |\Delta x|$ and then] *nw, ne* [

A final illustration is an object inside or on the edge of the proximity which is described by

If $|\Delta x| + |\Delta y| <= \varepsilon$ then *proximity*

The methods for determination of the relations between two objects with uncertain positions can now be introduced. This can be illustrated by the two cases in figure 6. The case to the left in the figure shows two objects, A and B, where the areas of uncertainty are overlapping. The conclusion of this is that object B is in the proximity of A. In the right alternative there is no overlap, which indicates that B is outside A and that the direction of B relative A is [*n, e*]. Clearly, the inverse relations are in both case equivalents.

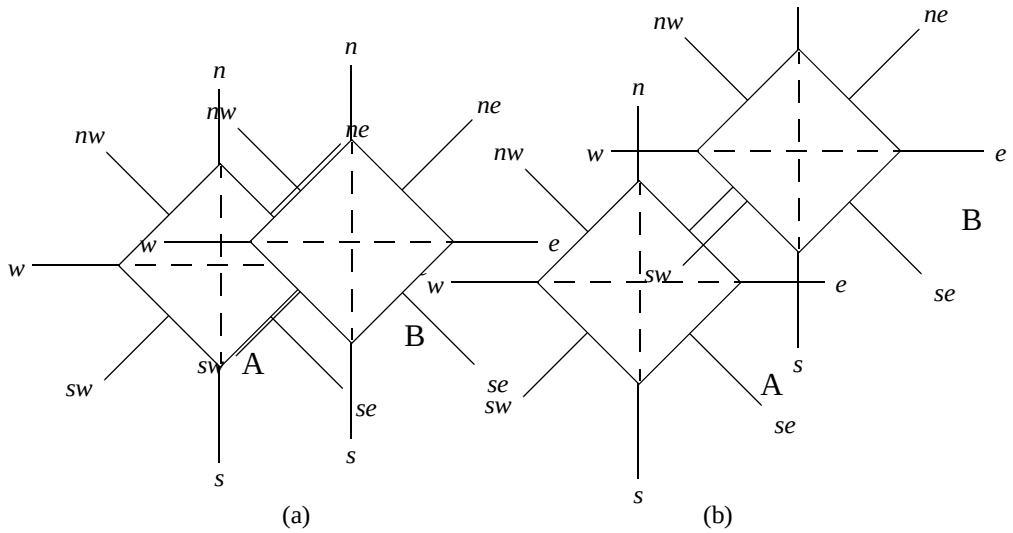


Figure 6. Some possible relations between the objects A and B; (a) B is in the proximity of A; (b) B is outside the proximity of A and B is [*n, e*] of A.

This methodology can be used for determination of qualitative distances as well. The set of qualitative distance measures proposed here is {*proximity, close, distant*}. This set can, of course, be extended but for the time being it is sufficiently adequate. The qualitative distance structure is illustrated in figure 7. The close and distant distances can be determined from the following rules:

If $|\Delta x| + |\Delta y| >= \varepsilon$ and $|\Delta x| + |\Delta y| < K\varepsilon$ then *close*

If $|\Delta x| + |\Delta y| > K\varepsilon$ then *distant*

where K is a constant that is application dependent. *Proximity* is determined from the rule introduced earlier in this section.

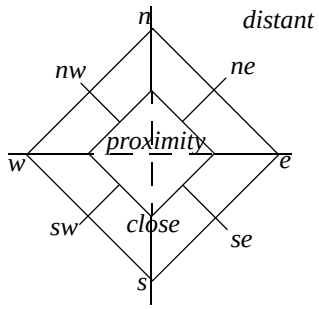


Figure 7. The qualitative distance structure for point objects.

8. CONCLUSIONS

In this paper a solution for how to handle mobile objects detected by sensors has been presented. The focus has concerned with how to handle uncertainties caused by the sensors that affect the reasoning about those objects. Firstly, mobile objects can be approximated with an area equivalent just to the uncertainty in position. A consequence of this is that management of mobile artifacts with uncertain positions differ from how areas with uncertain boundaries should be treated. Finally a way of reasoning qualitatively about relations between such objects has been proposed.

In the paper the uncertainty areas of the point objects are approximated with a circle or a rhombus. In many cases that is a good approximation, but for some sensors that is not a sufficiently good approximation. Future work will include looking into how this work can be generalized to include sensors with more irregular uncertainty areas as well.

Another research topic that has not yet been explored is how to handle uncertainties over time. Apart from affecting relations like before and after it might also influence the relative position if the two objects were not detected at the same time.

9. REFERENCES

1. Chang, S.-K., Costagliola, G., Jungert, E and F. Orciuoli, *Querying Distributed Multimedia Databases Data Sources in Information Fusion Applications*, IEEE Trans. on Multimedia, Vol. 6, No. 5, October 2004, 687-702.
2. Clementini, E. and Di Felice, P., *Approximate Topological Relations*, International Journal of Approximate Reasoning, v 16, n 2, February, 1997, p 173-204.
3. Clementini, E. and Di Felice, P., *A spatial model for complex objects with a broad boundary supporting queries on uncertain data*, Data and Knowledge Engineering, vol. 37, pp. 285-305, 2001.
4. Egenhofer, M., *Deriving the combination of binary topological relations*, Journal of Visual languages and Computing, Vol 5, pp 133-49.
5. Hall, D. L. and Llinas, J. (Eds.), *Handbook of Multisensor Data Fusion*, CRC Press, New York, 2001.
6. Horney,T. , Jungert, E., Folkesson, M., *An Ontology Controlled Data Fusion Process for Query Language*, Proceedings of the International Conference on Information Fusion 2003 (Fusion'03), Cairns, Australia, July 8-11.
7. Horney, T., Ahlberg, J., Jungert, E., Folkesson, M., Silvervarg, K., Lantz, F., Franssson, J., Grönwall, C., Klasén, L., Ulvklo, M., *An Information Sustem for target recognition*, Proceedings of the SPIE conference on defense and security, Orlando, Florida, April 12-16, 2004.
8. Pawlak, Z., *Rough sets*, International Journal of Computer and Information Sciences, 11(5):341-356, 1982.
9. Silvervarg, K. and Jungert, E., *Visual specification of spatial/temporal queries in a sensor data independent information system*, Proceedings of the tenth International Conference on Distributed Multimedia Systems, San Francisco, California, September 8-10, 2004, pp 263-268.
10. Worboys, M. F., *GIS - A Computing Perspective*, Taylor & Francis, london, 1995, pp 99-100.

Appendix E

An Ontology Controlled Data Fusion Process for a Query Language

Information Fusion, Cairnes, Australia, July 8-11, 2003.

Horney,T. , Jungert, E., Folkesson, M.

Proceedings of the Int. Conf. on Information Fusion (Fusion'03), Cairnes, Australia, July 8-10, 2003.

An Ontology Controlled Data Fusion Process for a Query Language

Tobias Horney, Erland Jungert, Martin Folkesson
Swedish Defence Research Agency (FOI)
Box 1165, SE–581 11 Linköping, Sweden
{tobho, jungert, marfol}@foi.se

Abstract - *Query languages designed for acquisition of data from multiple sensor data sources where the data generally are of heterogeneous type requires a number of internal functionality that is not available in traditional query languages. The required functionality can, for instance, be a method for multi-sensor data fusion, methods for query optimization and refinement. A further required technique relates to the problem of selecting sensors that efficiently can respond to the various queries without laying the responsibility of the sensor management on the users. The approach taken in this work has been to introduce an ontological knowledge-based system that can support selection of both sensors and object recognition algorithms as well as the control of the sensor data fusion process.*

Keywords: Ontology, data fusion, query language.

1 Introduction

Information systems attached to various types of heterogeneous data sources, which mainly consist of multiple sensors are required in many different applications that will be integrated into command and control systems. Applications can be both of military and civilian type and example of the latter may be emergency management where functionality for information fusion, i.e. basically for situation and impact assessments, are required. From a user's perspective this kind of systems will become very complex and consequently tools to control the sensor environment and, in the end, to support decision-making will be needed as well. These aspects can, however, be viewed as two sides of the same coin where the decision support tool is facing the end-user while the means for controlling the internal environment must be integrated into the system to control all aspects of the management of the sensors and the sensor data. In the latter case a controversy is apparent in that the question will arise whether the end-users of a multi-sensor system should be able to directly view the sensor data or whether they should just be able to overlook abstract high-level information extracted by the system in the query process. A solution to this problem is a system design characterized by the latter aspect, i.e. a system where the sensor raw-data will be hidden from the user. This is motivated by the complexity of the system, to minimize the heavy workload of the end-users and to allow the users to make their decisions under less stressful conditions. Another aspect of importance is that the users may lack capabilities for analysis of sensor data images. An approach to overcome these issues may be described as *sensor data independence*. To establish sensor data independence two characteristics must be fulfilled, first the system must be able to select a sensor while considering e.g. the present weather and light conditions. Secondly, a proper recognition algorithm must be chosen as well, i.e. an algorithm that supports an efficient recognition of the requested targets under the existing conditions. Finally, means to control the sensor data fusion process and to determine the interconnections between the controlling part and the fusion process must be established. A system designed to support all these characteristics will need a structure that on the basis of the requested targets is able to pick the most appropriate sensor(s) and recognition algorithm(s) and access, analyze and eventually fuse the information gathered from the sensors. In the work described here an ontological knowledge-based system has been developed to enforce sensor data independence and to control the sensor data fusion process. Finally, the ontological system is integrated into a query language called ΣQL [1], [2] particularly designed for acquisition of target information from heterogeneous data sources. The work on the ontological knowledge-based system was originally carried out as a master thesis [3].

Work combining the use of ontologies with information fusion in various ways is still a relatively new research topic and for this reason not much work has been done. Among the work that has been done, so far, the most common focus seems to be concerned with the use of ontologies for determination of the input from sensors and other types of heterogeneous input data sources. For example, in [4] it is demonstrated how an ontology can be used on simulated sensor data input for automatic recognition of various types of symbolic targets while synthesizing the recognition algorithm automatically at run-time. In [5] a generic and extensible prototype platform for intelligence fusion is presented. The fusion process is designed as an *algebra* of fusion operations and the data and knowledge semantics are given explicitly by an input ontology

together with descriptive models of the ontological concepts. Ontology-based programming using the NUT language is presented in [6]. A specification method and problem-solving technique is demonstrated by an example. [7] describes the use of an ontology in a system that is facilitating data, information and knowledge to dynamic end-users. Finally, work combining the use of ontologies in connection with query languages can be found in [8].

The structure of this paper is the following: Section 2 presents the problem. Section 3 gives a brief overview of the host system, i.e. the query language. In section 4, the environment of the ontological knowledge-based system, including the knowledge structure i.e. the ontology itself, is discussed. Section 5 presents the taken multi-sensor data fusion method and in section 6 the complete data fusion process is discussed together with some important issues involved in this process. In section 7, the conclusions and future research is presented.

2 Problem description

Among the requirements in query languages generality is probably the most important and basic one. Establishing generality is consequently a fundamental task in the query language design process. Query languages for multi-sensor data are in this regard no exception. The problem of developing a sufficiently general query language for sensor data in which sensor data fusion plays a vital role involves a number of design issues of which two will be further discussed in this work. The background of these two aspects can be found in the users' work situation, which in particular becomes very complicated in sensor oriented systems. This work situation is for the most part very intense and where the users have designated tasks that may have to be finished in a very short time. For this reason the workload of the users must be delimited by eliminating such tasks that are not involved with the tasks primarily assigned to the users. Among these tasks are selection of sensors, the analysis/interpretation of sensor data and multi-sensor data fusion of particular importance. From a user's perspective the elimination of these tasks eventually will permit the design of a much more general query interface that will allow a more focused and problem oriented usage. Thus, the problem addressed here can be formulated as follows:

> *How can, in a query language for sensor data, tasks concerned with the selection of sensors, analysis of sensor data and control of the multi-sensor data fusion process automatically be carried out without user involvement?*

The approach taken here to solve this problem is mainly based on the use of an ontological knowledge-based system that takes care of the selection of sensors and recognition algorithms for analysis of the sensor data. This concept is subsequently called *sensor data independence* [9] and is related to what is called data independence [10] in traditional database theory. Sensor data independence is an important feature of ΣQL but will not be discussed further in this work. Nevertheless, the ontological approach has a number of other consequences that must be taken into consideration as well. Since many sensors are slow in collecting data from large areas, means to speed up the turn around process must be developed; this is generally called *query refinement* [11]. Furthermore, aspects of uncertainty in sensor data must be considered. These latter aspects are of concern for the data fusion control process and will be discussed further in section 6.

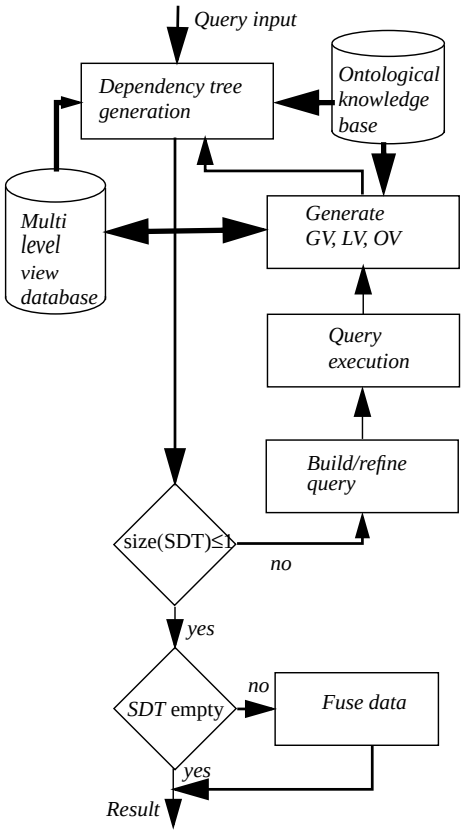


Figure 1: The information flow of ΣQL.

3 ΣQL - an overview

User queries applied to ΣQL will as a consequence of the sensor data independence concept be based on terms that for the most parts are familiar to the users. Clearly, this is due to the fact that they do not have to include any sensor information. The user concepts that need to be identified are (1) the area of interest (AOI), (2) the time interval of interest (IOI) and (3) the object type that will be subject to the search in the query. These three concepts are the most important ones that must be determined in order to get a response to a query. Sometimes object attribute values and object relations must be part of the query as well. The design of a useful graphical user interface is currently going on. Two basic query classes exist, i.e. queries applied to historical data and real time sequentially repeated queries that go on over long periods of time during which the various sensor types may vary as a consequence of changing conditions like weather and light. Currently, a fairly simple prototype of ΣQL can be run.

The basic functionality of the query language can be seen in figure 1. A query is inserted by the user and then the input is fed in to the dependency tree generator which in

dialogue with the ontological knowledge structure generates a set of nodes in the dependency tree and as long as there are nodes in this tree new subqueries can be built (one for each of the selected sensors), refined and executed in the query processor. Once the subqueries have been executed instances are created in the multilevel view database to support query refinement in a subsequent step, meaning a new set of dependency tree nodes are generated which in turn means that new subqueries will be executed and possibly further refined. This goes on until the dependency tree becomes empty, i.e. when there is no more sensor data available to process. In the final step data fusion is carried out, if applicable, and then the process terminates which is further discussed in [11].

A view is a symbolic representation, i.e. a mapping, of some part of the world in some resolution. Currently, there are three views defined; the global, local and object views. Those views describe the object structure (object view), the object in a small object surrounding (local view) and the object position in relation to other objects (global view). The multi-level view database is where all the symbolic information about objects, object surroundings etc. is stored. More details about the multi-level view database can be found in [11].

A serious question is how a user should interpret the fused result of a query. The approach taken here has been to associate a *belief value* to each result from the various sensors that are being used in a query. A final belief value is eventually determined by the fusion process, which also is forwarded to the user as a part of the query result. All belief values are normalized and a high value means that the user may have a high belief or confidence in the result. This is an important aspect of the system that has been introduced to, if possible, give the user a high degree of trust in the query result as well as in the query system as such. A fundamental problem associated with this idea is how to calibrate the belief values. This work is going on and is subject to further studies.

4 The Ontological Knowledge-based system

The knowledge represented in the ontological knowledge base is modelled in a hierarchical manner known as the ontology. All concepts in the universe of discourse, the interesting properties of the concepts and the important relations between the concepts are modelled. The hierarchy has the ultimately general concept called *Thing* at the top. All other concepts inherit directly or indirectly from Thing. This means that *Everything is a thing*. The hierarchy is organized so that more specialized concepts appear further down the inheritance chain. The concepts of this ontology are divided into three main parts. One part models everything that can be sensed by the sensors and everything that can be recognized by the recognition algorithms or cued by the cueing algorithms. The second part models the characteristics of the sensors and the recognition and cueing algorithms. The third and last part models all the conditions that have an impact on the appropriateness of the sensors and recognition/cueing algorithms. When the ontological structure has been created it is populated with instances. For example, in figure 2, the *Tank* concept is shown. Different tanks can be added to the ontology as instances of that concept. When instances are added to an ontology the ontology becomes a knowledge base. In reality, there is a fine line between where the ontology ends and the knowledge base begins [12]. Here, the ontology filled with instances is called the ontological knowledge base, or simply the knowledge base.

4.1 Things to be Sensed and Recognized

The "Things to be Sensed and Recognized" part of the ontology models everything that can be sensed by the sensors and everything that can be recognized by the recognition algorithms or cued by the cueing algorithms. It is represented in the ontology, see figure 2, by the *ThingToBeSensed* concept and subconcepts. Two subbranches exist: *PropertyToBeSensed* and *RecognizableObject*. The *ThingToBeSensed* concept has the relation *HasAppropriateSA* describing which combinations of sensors and algorithms (SA) are appropriate for finding the *ThingToBeSensed*. This is an important relation for the AFFAS algorithm, see section 4.5, which is the algorithm designed to determine which sensors and which cueing and recognition algorithms to be applied in a certain query. Note that relations are inherited, so a relation that exists in *ThingToBeSensed* also exists in all concepts that inherit from *ThingToBeSensed*.

The subbranch *PropertyToBeSensed* models the attributes (e.g. colour) that the recognizable objects might have. Representing object properties separately enables specific SAs to be applied to determine specific attributes, not only to determine recognizable objects. This is represented in the ontology by the *PropertyToBeSensed* concept.

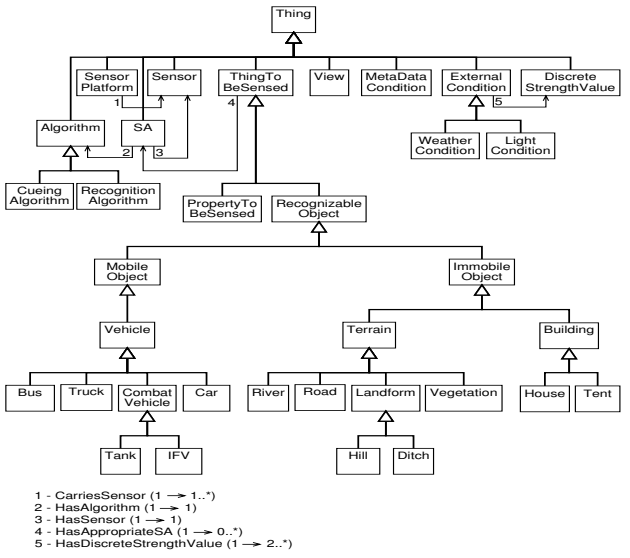


Figure 2: Ontology overview, the knowledge structure.

The second subbranch of *ThingToBeSensed* is *RecognizableObject* which models the objects that can be recognized by the recognition algorithms or cued by the cueing algorithms. Notice that every recognizable object (RO) has the relation *HasAppropriateSA* because every RO inherits from *ThingToBeSensed*. Recognizable objects are further

divided into mobile and immobile objects. The ontology is populated with vehicles that are taxonomically located under *MobileObject*. Immobile objects are objects that cannot move. Important such objects are different kinds of terrain objects and buildings.

The *Vegetation* concept is used for modelling the condition terrain background. Therefore *Vegetation* not only fits into the *ThingsToBeSensed* part of the ontology but also to the *Conditions* part because it models a condition that is considered in the AFFAS algorithm.

4.2 Sensor and Algorithm Characteristics

The "Sensor and Algorithm Characteristics" part of the ontology models the characteristics of the sensors and the recognition and cueing algorithms.
This part includes the concepts

- *Sensor platform*
- *Sensor*
- *Algorithm* (incl. subconcepts)
- *SA*

The *SensorPlatform* concept models the available sensor platforms. This concept is also used for modeling the condition sensor platform, where the type of the platform is taken into consideration. Therefore *SensorPlatform* not only fits into the *Sensor and Algorithm Characteristics* part of the ontology but it also fits into the *Conditions* part because it models a condition that is considered in the AFFAS algorithm. The *SensorPlatform* concept has the relation *CarriesSensor* describing which sensors the platform carries.

The *Sensor* concept is used for modelling sensors. Currently, the following sensors are modelled: CCD (digital camera), IR (infrared camera) and LR (laser radar).
Algorithms are used for extracting information from the sensor data collected by the sensors. The most important types of algorithms are recognition algorithms and cueing algorithms. Other types of algorithms working on sensor data might very well be modelled. That is why the general *Algorithm* concept is included. The *RecognitionAlgorithm* concept models the recognition algorithms used for recognizing different types of things in sensor data and the *CueingAlgorithm* concept models the cueing algorithms used to decrease the size of the region of interest for the recognition algorithms, thereby pointing out potential target objects.

SA is an important concept since it models the combination of sensors and algorithms. A combination is modelled by the two relations *HasSensor* and *HasAlgorithm*. The *HasSensor* relation connects the *SA* to a *Sensor* and the *HasAlgorithm* relation connects it to an *Algorithm*. The *SA* concept is introduced because there is a many-to-many relation between the *Sensor* and *Algorithm* concepts. To model this, an *SA* is created in the ontological knowledge base for every sensor and algorithm combination that works together.

4.3 Conditions

The "Conditions" part of the ontology models the conditions that have an impact on the appropriateness of the sensors and the recognition/cueing algorithms. The conditions are state conditions describing the state of something, for example how rainy it is. Each of the existing conditions will only be briefly discussed here. The concepts that make up this part are

- *View*
- *MetaDataCondition*
- *ExternalCondition* (incl. subconcepts)
- *DiscreteStrengthValue* (not a condition in itself)

A view is a symbolic representation of a part of the world in some resolution. The view concept is modelled in the ontology by *View.*

The metadata conditions are represented in the ontology by the *MetaDataCondition* concept. Metadata means data about data, and in this case we are interested in the quality of the data and also what sensor platform that captured the data. If no data exists from a certain sensor in a certain area at a certain point in time the data quality is defined as zero for that sensor in that area at that point in time.

External conditions are represented in the ontology by the *ExternalCondition* concept. External conditions are weather conditions (*WeatherCondition* concept) and light condition (*LightCondition* concept). This concept has the relation *HasDiscreteStrengthValue* that provides the connection between external conditions and the corresponding discrete strength values.

Discrete strength value is represented in the ontology by the *DiscreteStrengthValue* concept. All conditions have specific values at any given time and place. A discrete range of permitted values has been defined for each condition. For example, Rain might have the following permitted values: {Dry, Gentle, Heavy}. The *DiscreteStrengthValue* concept models all discrete strength values for all external conditions. The *HasDiscreteStrengthValue* relation provides the connection between external conditions and the corresponding discrete strength values.

Terrain background is modelled using the *Vegetation* concept in the *ThingsToBeSensed* part of the ontology. This is because all types of vegetation known by the system are modelled using that concept, and that is exactly what the terrain background condition is designed to take into consideration. Hence, this condition uses the instances of the *Vegetation* concept to make up its range of permitted values. Examples of terrain backgrounds are sand and water.

4.4 Relations

Relations are used to model how the concepts in the ontology are related to each other. It is important to note that relations are inherited, meaning that if concept B inherits concept A and concept A has a relation to C, then concept B automatically has that relation to C as well. An example of relation inheritance is that the *Tank* concept has the *HasAppropriateSA* relation because it is inherited from the

ThingToBeSensed concept where that relation is defined.

In the context of ontologies a relation is always defined to be a connection from a certain concept to another concept, e.g. the *HasSensor* relation is defined to be a connection from the *SA* concept to the *Sensor* concept. Of course there is an inverse relation saying that a *Sensor* is part of an *SA*. However, this is not explicitly defined, because the inverse relation is not needed here. The following relations are defined:

- *HasAlgorithm* from *SA* to *Algorithm*
- *HasSensor* from *SA* to *Sensor*
- *CarriesSensor* from *SensorPlatform* to *Sensor*
- *HasAppropriateSA* from *ThingToBeSensed* to *SA*
- *HasDiscreteStrengthValue* from *ExternalCondition* to *DiscreteStrengthValue*

The relations are shown in the ontology overview in figure 2. More details can be found in [3].

4.5 Appropriate sensors and algorithms

The ontological knowledge base has been designed to help answering such questions as which sensor data to use under certain circumstances. Also of importance is which recognition and cueing algorithm(s) that should be applied. An algorithm that performs these selections using the ontological knowledge base has been developed. That is, it uses the knowledge in the ontological knowledge base in conjunction with the knowledge base rules (see section 4.5.2) to determine which sensors and recognition/cueing algorithms are the most appropriate under the given circumstances, i.e. the actual ΣQL query, the metadata conditions, the external conditions and the terrain background. The algorithm is called *Algorithm For Finding Appropriate SAs (AFFAS)* and is described in detail in [3]. A short overview of AFFAS is presented in the next section.

4.5.1 Overview of AFFAS

A short description of the four steps of the algorithm is provided below.

Step 1
A list of appropriate SAs is created, with respect only to the type of thing to be sensed. Each SA has an appropriateness value (*Ap*) connected to it.

Step 2
The following things are considered:

- Meta data conditions
- Type of sensor platform
- Data quality

This step alters the appropriateness values of the SAs from step 1 with respect to the information in the metadata.

Step 3
The following things are considered:

- Weather conditions
- Light condition
- View
- Terrain background

This step creates the impact factors describing how strong the impacts from the current external conditions, the view and the terrain background are. This is done for each sensor and algorithm in the SAs in the result from step 2.

Step 4
The results from step 2 and step 3 are weighted together. This step alters the *Ap* values of the SAs in the result from step 2 according to the result from step 3. In the list created in this step, the SAs are prioritized according to the weighted appropriateness values.

4.5.2 Knowledge Base Rules

In the process of deciding upon appropriate sensors and algorithms it is necessary to have rules describing under which conditions certain sensors and algorithms are appropriate. The rules that are used to decide how the impact factors impact the sensors and recognition/cueing algorithms in the SAs can be written in the following form:

"If an impact factor **x** has the discrete strength value **y** then the impact on the sensor/algorithm **z** is impact strength value **v**"

Example 1:
"If the impact factor **Rain** has the discrete strength value **Gentle** then the impact on recognition algorithm **BuildingAlgorithm** is impact strength value **Little**"

Example 2:
"If the impact factor **View** has the discrete strength value **Local** then the impact on the sensor **Standard CCD Sensor** is impact strength value **None**"

A complete set of rules is needed for the system to function properly. Definitions of *impact factor*, *discrete strength value* and *impact strength value* are presented in [3]. The definitions are quite straight-forward.

5 The sensor data fusion method

In the system described above fusion will take place in mainly two different steps, separated by a query refinement. Although logically separated in the process, one and the same reasoning should permeate both fusion approaches. In case a query response is made up by only one set of data in either step, fusion is ruled out.

The first step concerns fusion of attribute data. These data come from the set of algorithms chosen by the ontology for the specific situation. The fused attribute data serve as a unified input to the recognition algorithms. Although estimation and recognition are logically distinct, it is possible to think of using the same algorithm for both cases. The time stamps on sensor data can be considered an attribute among others, so that data association is inherent in this process. The output of this step, together with the chosen object types, can be seen as one or more hypotheses that have to be tried by the recognition algorithms.

No attribute fusion method has yet been implemented. Some kind of clustering method seems to be needed. Such a method should be used to identify distinct hypotheses in a space made up by the given attributes including time.

The clusters could then be represented either by one of its data points, or some mean of its data points. Choosing the former avoids the problem of combining attribute values from different algorithms. Such a combination needs information on which combinations of attribute values that are allowed. Should two attribute values not be consistent for any object in the model library (see section 6.3.3) they must be assigned to two different hypotheses. Alternatively, the situation could be interpreted as the detection of a model that is not stored in the model library. Whether the latter should be allowed or not must be subject to further research.

Still, combining attribute values from different algorithms might be exactly what needs to be done in order to exploit the advantages of fusion. This is most likely the case when the algorithms are incapable of estimating all attributes.

The second data fusion step concerns the output from the recognition algorithms, which consists of an object type with corresponding attribute data and a belief value.

From a system view the second data fusion step should not be critical, i.e. two slightly different approaches should not result in two entirely different fusion results. The role of fusion is here instead regarded as the one of combining partial results in a relatively fast way. This favours a simple approach. Furthermore, we concentrate on recognition results that have *high* belief values. Low belief values can origin from quite a few reasons, such as noise, bad input (hypothesis) etc. High belief values are simply, and for natural reasons, rare, and therefore believed to correspond to a good/correct hypothesis.

Picking the overall best recognition result would make the system sensitive to bad belief value calibration between the different recognition algorithms. Therefore, summing belief values over object types is first performed. Since one or more hypotheses have been tried more times than other, mean beliefs are calculated. In the current implementation rms values are used, which favours good recognition. Hopefully, at this point there is a significant spread in belief for different object types. The winning type is chosen, then the hypotheses corresponding to the best recognition for that object type is taken as final result of the fusion. The approach can be seen as a combination of *weighted, exact plurality voting* and a *selection rule* [13]. Note that the set of attribute values in each hypothesis are not manipulated here, but considered as completely enclosed in the hypothesis.

The division of fusion into two steps relies on the assumption that attribute estimation truly can be separated from recognition. The recognition algorithms must have at least some degree of freedom, in order to obtain results - belief values and refined attribute data - that can be considered independent. However, using mean values in the second fusion step should make the final result robust to unwanted, systematic correlation between results from certain recognition algorithms. Also, a clustering method in the first fusion step would be robust to unwanted correlations in query responses from different attribute estimation algorithms.

6 The data fusion control process

The actual execution of a query, that is, everything performed between the reception of a query entered by the user and the presentation of the query result, is performed in a process controlled by the ontology; this includes the control of the data fusion process as well.

6.1 Overview

There are two basic levels in the data fusion control process. The first level is the cueing level where the area of interest (AOI) can be large and the time interval of interest (IOI) can be long. Cueing in this sense means finding potential target objects (the ones searched for in the query) and pointing out the positions of these potential targets so that the recognition algorithms, possibly using other types of sensor data than was used in the cueing, can classify and/or identify targets at those positions.

The second level is the recognition level where the recognition process takes place. This process is performed in two major steps where the first step involves estimating the attributes of potential targets and the second step involves matching the potential targets against certain models selected from a library of models. Since the system allows for multiple algorithms to perform attribute estimation as well as model matching it is necessary to perform data fusion to get a common understanding about the target object attribute values as well as the target object classification/identification. Therefore data fusion takes place both after attribute estimation and after model matching, see figure 3.

When recognition has been carried out it is time to create an answer to the query. This is done by evaluating the logical expressions enforced in the query by the user.

The concept of query refinement was introduced in [11]. Query refinement means executing the query in an iterative manner where the query itself is refined in each step. Using cueing to point out potential targets so that recognition algorithms only need to work on selected pieces of sensor data is one way of refining a query. Another type of query refinement is when attribute estimation is performed to reduce the search space for the model matching algorithms. An example of this can be that an attribute estimation algorithm estimates the size and the orientation of a potential target, for example by using laser radar data. The orientation information is then used by the model matching algorithms, independently from the type of sensor data they work on, to make an initial assumption about how to rotate the model from the library when trying to match it against the sensor data. This can reduce the search space and thereby the execution time. A third type of query refinement is performed using the object size estimations when selecting models from the model library. All models with a size too different from the size estimations are excluded from the matching process, thereby reducing the search space since fewer models are required.

There are certain dependencies in the system. Those

dependencies are dealt with using the sensor dependency tree. An example of a dependency is that all cueing and recognition algorithms require sensor data of proper type at the given place and time. Another example is that all recognition algorithms (i.e. attribute estimation and model matching algorithms) require cueing.

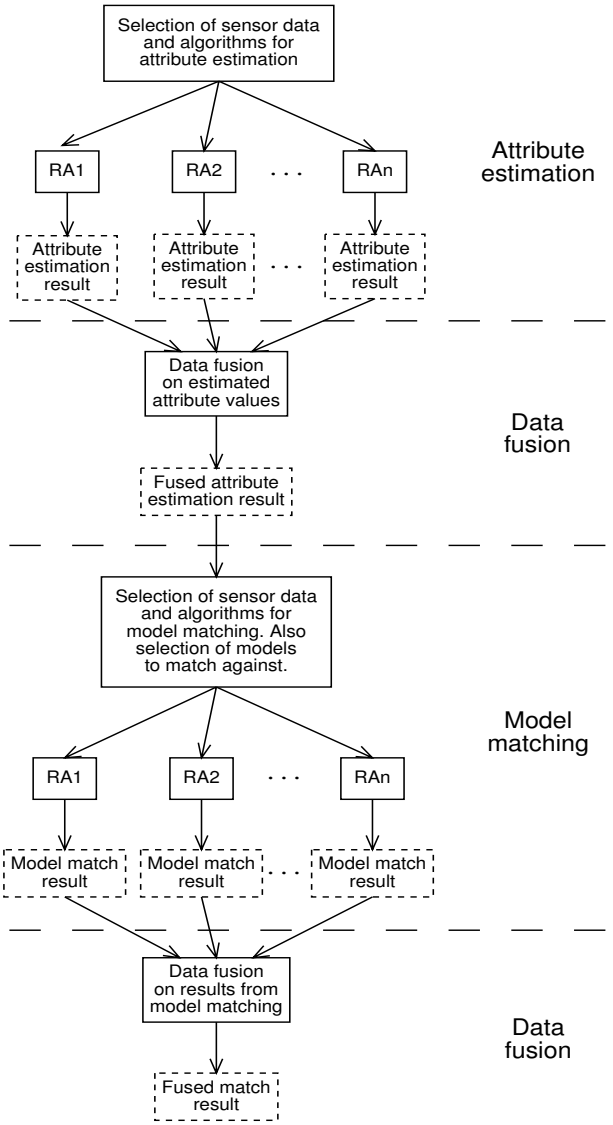


Figure 3: Data flow on the recognition level.

6.2 The cueing level

The cueing level of the target recognition process involves finding potential target objects and pointing out the positions (in spatial and temporal terms) of those objects so that the recognition algorithms working at the recognition level need only to work on sensor data from those spatial/ temporal positions.

The first step is to select which sensor data and which cueing algorithm(s) to use, which is done by the AFFAS algorithm. The next step is to perform the actual cueing by executing the selected algorithms using the selected sensor data. The cueing algorithms work on data gathered inside AOI during IOI. AOI can be large and IOI can be a long period of time. The cueing task is to produce one small area of interest and one small IOI (possibly a point in time) for each potential target. After cueing, the system will perform recognition in each combination of AOI and IOI produced by the cueing algorithms. That means, if multiple cueing algorithms are executed the final cueing result will be the union of the results from the algorithms executed.

6.3 The recognition level

The recognition level of the target recognition process involves classification and possibly identification of the potential targets pointed out by the cueing algorithms. This is done as described in figure 3.

Sensor data and recognition algorithms are first selected that is done by the AFFAS algorithm. In the second step attribute estimation is performed by the selected recognition algorithms that can estimate attributes. If multiple attribute estimation algorithms have been executed data fusion is carried out. In the next step a selection of models from the model library is accessed and the selected recognition algorithms that can perform matching are executed. If multiple matching algorithms were executed data fusion is performed on the match results. In the last step the logical expressions enforced in the query by the user are evaluated and a final result is constructed.

6.3.1 Attribute estimation

In the attribute estimation step the selected attribute estimation algorithms tries to estimate certain attributes of a potential target found by the cueing process. Estimations of attribute values are calculated by extracting information from the sensor data. Multiple attribute estimation algorithms can be executed in parallel if required.

In the current prototype system only one attribute estimation algorithm is used so far, even though further estimation algorithms are under development. The one used in the current prototype works on laser radar data and estimates the length, width and height of the possible targets.

6.3.2 Fusion of attribute estimations

If multiple attribute estimation algorithms are executed or if multiple sensor data are used to determine the attribute values data fusion will be performed to create a final attribute estimation of each attribute. The attributes estimated by the different algorithms, possibly from different kinds of sensor data, can be partially disjunct, that is, some attributes may be estimated by several algorithms whereas others may be estimated by only one algorithm.

6.3.3 Selection of models for matching

When attribute estimation has been carried out the attribute estimations are used to reduce the number of target object models that need to be accessed from the model library and against which the matching algorithms will match. Each model in the library is a CAD-model with textures and/or other kinds of a priori knowledge that can be used by the

matching algorithms when matching against extracted features in the sensor data. Appropriate attributes to use as discriminating factors can for example be the length, the width and the height of the object, i.e. if estimation of such attributes can be performed.

If no attribute estimation has been performed, for example because there are no sensor data available, then it is impossible to make a selection this way and thus other means must be developed to take care of the model selection.

6.3.4 Model matching

By using the attribute estimations as start values in the iterations of the matching process the model matching algorithms can efficiently perform the actual matching and produce a belief value for each model. Multiple matching algorithms can be executed in parallel using the same or different estimated attributes, working on the same or different sensor data types.

In the current prototype system four different model matching algorithms are used; one for laser radar data, two for IR data and one for IR and/or CCD data. Algorithms for further sensor data types are under way.

6.3.5 Fusion of model match results

If multiple model matching algorithms were executed data fusion is performed.

6.4 Query evaluation

When the matching process has been finished it is time to produce an answer to the query. This is done by evaluating the logical expressions enforced in the query by the user using the fused match result including the fused attribute estimations as input data. That is, the earlier steps in the process have created high level information about the objects asked for in the query inside AOI during IOI. Any query requirements on certain attribute values, number of target objects, spatial relations between objects etc. are now handled by the query processor. In the current prototype a very simple query processor is implemented which is only able to process simple types of queries. However, in the near future a new query processor, which is able to process more advanced queries, will be implemented.

7 Conclusions and future research

In this work, an ontology together with its knowledge-based system intended as a controller of the fusion process in the query language ΣQL has been described and discussed. The main purpose of the ontological knowledge base has been to permit sensor data independence that generally can be seen as a method that permits the users to focus on their own work without being bothered with technical aspects that generally require knowledge about various types of sensors and sensor data. The fusion control mechanism determines, under the control of the ontological system, what to fuse and when to do it in conjunction with an approach to query refinement developed to improve the quality of the query output, but also to reduce the turn-around time required to respond to a query.

Future research on where the ontological knowledge base can be used relates to applications which will require reasoning to determine more complex information and knowledge than can be done by a query language. Examples of such applications are data mining, generation of common operational pictures but also situation analysis. In the latter case, the idea is to demonstrate how an ontology can be used in the higher levels of information fusion.

References

[1] Chang, S.-K., Costagliola, G., Jungert, E. and Orciuoli, F., *Querying Distributed Multimedia Databases and Data Sources for Sensor Data Fusion*, accepted for publication in the journal of IEEE transaction on Multimedia, 2003.

[2] Chang, S.-K. and Jungert, E., *Query Languages for Multimedia Search*, In Principals of Visual Information Retrieval, M.S. Lew (Ed.), Springer Verlag, Berlin, 2001, pp 199-217.

[3] Horney, T., *Design of an ontological knowledge structure for a query language for multiple data sources*, LiTH-IDA-Ex-02/22, Department of Computer and Information Science, Linköping University, Sweden, March 2002.

[4] Kokar, M. M. and Wang, J., *Using Ontologies for Recognition: An Example*, Proceedings of the 5th International Conference on Information Fusion, Annapolis, Maryland, July 2002, pp 1324-1330.

[5] Royer, V. and Challine, J-F., *A Platform for Interoperable Fusion Models*, Proceedings of the 3rd International Conference on Information Fusion, France, July 2000.

[6] Kotkas, V., Penjam, J. and Tyugu, E., *Ontology-based design of surveillance systems with NUT*, Proceedings of the 3rd International Conference on Information Fusion, France, July 2000.

[7] Capraro, G. T., Berdan, G. B., Berra, P. B., Spina, J. and Liuzzi, R. A., *An architecture for providing Information Anytime, Anywhere and on Any Device - An Ontological Approach*, Proceedings of the 5th International Conference on Information Fusion, Annapolis, Maryland, July 2002, pp 1331-1338.

[8] Mena, E. and Illarramendi, A., *Ontology-based query processing for global information systems*, Kluwer Academic Press, Boston, 2001.

[9] Jungert, E., Silvervarg, K. and Horney, T., *Ontology driven sensor independence in a query supported C_2-system*, Proceedings of the NATO workshop on Massive Military Data Fusion and Visualization: Users Talk with Developers, Halden, Norway, September 2002.

[10] Ullman, J. D., *Principles of Database and Knowledge-base Systems*, Computer Science Press, Rockville, Maryland, 1988.

[11] Chang, S.-K., Costagliola, G. and Jungert, E. (2002), *Multi-Sensor Information Fusion by Query Refinement*, Proceedings of the 5th International Conference on Visual Information Systems, Taiwan, March 2002.

[12] Noy, N. F. and McGuinness, D. L., *Ontology Development 101: A Guide to Creating Your First Ontology*, KSL 01-05, Stanford University, 2001.

[13] Parhami, B., *Voting algorithms*, IEEE Transactions on Reliability, Vol. 43, No. 4, December 1994.

Appendix F

Iterative Information Fusion using a Reasoner for Objects with Uninformative Belief Values

Information Fusion, Stockhom Sweden, June 28- July 1, 2004.

Chang, S.-K., Jungert, E.

Iterative Information Fusion using a Reasoner for Objects with Uninformative Belief Values

S. K. Chang[1] and Erland Jungert[2]
[1]Dept. of Computer Science, University of Pittsburgh, USA (chang@cs.pitt.edu)
[2]Swedish Defence Research Agency (FOI), Sweden (jungert@foi.se)

Abstract - *Abstract: We describe an approach for iterative information fusion using a context-dependent Reasoner called Pequliar. The system basically consists of a query processor with fusion capability and a Reasoner with learning capability. The query processor first performs a query to produce some initial results. If the initial results are uninformative, then the Reasoner guided by the user creates a more elaborate query by means of some rule and returns the query to the query processor that executes it and returns a more informative answer. Rules may be initially specified by the user and subsequently learned by the Reasoner. Examples of iterative queries are drawn from multi-sensor information fusion applications.*

1 Introduction

Reasoning about data of uncertain type is necessary in a growing number of complex applications and especially for such applications where the basic data sources correspond to sensors of multiple type and where the generated data are of heterogeneous type. Data from such sensor sources are always associated with some level of uncertainty. To gain acceptable result from the analysis of data from multiple sensors multiple sensor data fusion is required [9]. For this reason, information, that for the most part is of spatial and temporal type need to be acquired from the sensors. It has been shown, e.g. in [4], that this can be made by means of a specifies query language. Subsequently it will be demonstrated how the output from the query language can be used as input to a spatial/temporal reasoning tool that can resolve the uncertainty problems. Basically, the Reasoner creates a more elaborate query by means of some rule and returns the query to the query processor that executes it and returns a more expanded answer. The Reasoner described in this paper is called Pequliar and can be described as a post-query-language-reasoner. The name "Pequliar" is an acronym for ***P***(ost)+***e*** +***QU***(ery) ***L***(anguage)+***iar***+***R***(easoner).

Pequliar can be seen as a Reasoner for objects determined through queries in a query language. In particular, the query language used is ΣQL [2], [4]. This query language has the ability to respond to queries where the input data sources are sensors that generate images and where the data is of uncertain type. More specifically, in ΣQL these uncertainties are associated with belief values that must be subject to further interpretation either by the user or as here by the Reasoner. Since we are here talking about multiple sensor data sources as input to the query language, it must be possible to fuse the information from the various sensors prior to the usage in the Reasoner; a capability that is available in ΣQL. The eventual output from Pequliar, that is rule driven, should help the user to get a better understanding of the external environment subject to the analysis carried out by the complete system. The query language and the Reasoner is developed to support a number of applications, such as data mining of primarily unstructured data [11], and for higher levels of information fusion, i.e. situation and impact analysis [9], [5].

As the object information is associated with various kinds of uncertainties and as the object information from a single sensor may or may not correspond to the same objects as acquired from other sensors this will have an impact on how sensor data fusion is carried out. To make it possible for the user to interpret the uncertainty level of an identified object the query language must somehow produce a measure of the actual uncertainty level that can be associated with the object information. In ΣQL this measure is called a belief value. However, this does not always create a result that is simple to interpret. One way to overcome this problem is, as already has been mentioned, to apply the query output to a Reasoner and to combine it with various types of background information that for the most part is context dependent, i.e. if the objects to be reasoned about correspond to vehicles then most likely the background information is geographical. Furthermore, this will also include information about the past behavior of the vehicles as well as the behavior of other objects. Besides being rule driven the reasoning process also requires information from

an ontology. Results from the Reasoner can, however, be handled in various ways, that is, if the result is feasible it is returned to the user, if not there must be a request sent to the sensors for more information followed by a rerun of the query. Besides this, background information may be accessed from other sources like geo-databases. Basically, Pequliar will take care of the query results that may be difficult to interpret without considering the background information. Consequently the Reasoner must also include some element of fusion since data from new data sources are integrated into the process. In Pequliar metadata is a necessary means for determination of the background information that will be needed in the reasoning process.

We describe an approach for iterative information fusion using a context-dependent Reasoner called Pequliar. Section 2 discusses the system design. The system basically consists of a query processor with fusion capability and a Reasoner with learning capability. As described in Section 3, the query processor first performs a query to produce some initial results. If the initial results are uninformative, then the Reasoner guided by the user creates a more elaborate query by means of some rule and returns the query to the query processor. The query processor executes it and returns a more informative answer. Rules may be initially specified by the user, and successful rules are learned by the Reasoner. Section 4 discusses reasoning issues. Section 5 and Section 6 present iterative information fusion based upon examples drawn from multi-sensor information fusion applications. Further research issues are discussed in Section 7.

2 System Design

2.1 System Overview

The system basically consists of a query processor with fusion capability and a Reasoner with learning capability. The query processor first performs a query to produce some initial results. If the initial results are uninformative then the Reasoner guided by the user creates a more elaborate query by means of some rule and returns the query to the query processor. The query processor executes it and returns a more informative answer. Rules may be initially specified by the user and subsequently learned by the Reasoner.

The name of the Reasoner - Pequliar - is obviously related to the word "*peculiar*". This word "*peculiar*" has several different meanings and in the context of this paper means "one of a kind", "singular" or "to be an object with uncertain and/or uninformative belief values". The system is therefore intended to handle objects that are one-of-a-kind, singular, or with uncertain and/or uninformative belief values. To accomplish this, any information peculiar to an object in the application domain must be stored in the ontology, and any information peculiar to spatial/ temporal reasoning must be stored in the rule base, see Figure 1.

2.2 Iterative Query Construction

Each user's query is considered a *compound query* that can be constructed in several iterations. In each iteration, the user constructs an *elementary query*, which is essentially a quintuple: <Object, Source, Time, Space, Direction>. Using query operators the elementary queries can be composed into a nested query in ΣQL. The composition rule is implicit, and the user does not need to know the underlying query language ΣQL for the nested query. What is required is an understanding of the meaning of different query operators. The main purpose of the user interface is therefore to provide a visual interface for the user to compose compound queries from elementary queries using query operators. A preliminary study of the user interface can be found in [10].

2.3 The Query Processor

The result of a ΣQL-query is generally the object types requested by the user including also the attribute and status values of these objects, if requested. Example of attributes are color, size etc. while the status of an object generally corresponds to such characteristics as position, orientation or speed etc. The difference between an attribute and a status value is basically that an attribute is not subject to change in the short range of time, that is, the color of a vehicle may change but not within the time frame of concern to the user. Status values may change within a very short time frame that may be less than seconds; consider for instance position and speed.

Common to all the information returned by ΣQL is that the type attribute is associated with a belief value. Other attributes and status values may have belief values associated to them as well but this is less common. For the most part, such belief values are given to indicate to what extent the result of a query can be believed. In the most general case the belief values are just given for the object types and from each type of sensor data and eventually there is also a belief value given as a result of the fusion process that takes place for the majority of the queries; this is due to the use of multiple sensor data sources. Cases when fusion is excluded may occur just for simple and trivial queries.

Other information from the query processor that might be of concern for the Reasoner are for instance the quality of the data in the area of interest given by the user. To determine the source data quality for a certain area of interest the corresponding meta-data will be required. This must, however, be subject to further research.

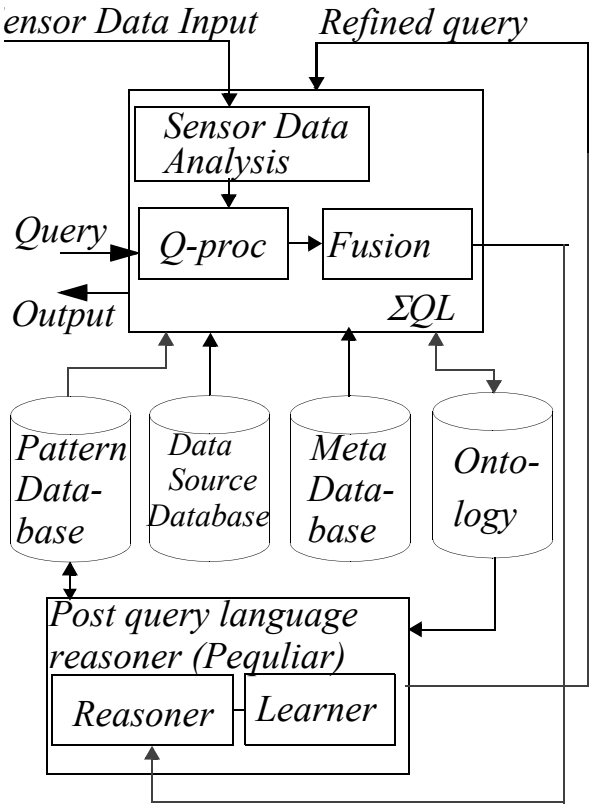


Figure 1. The diagram.

2.4 Iterative Information Fusion by Belief Value Adjustment

New, and hopefully more informative, belief values will be achieved through a reasoning step in Pequliar that includes generation of new and more elaborate query that will be executed subsequently. Input to this reasoning step is mainly the output, that may include the dependency tree information, from the query processor, the context information and meta-data. Secondary to this is the applicable ontological information. The meta-data is used to select the portion of the context information that corresponds to the area of interest (AOI). Once the Reasoner has come to a conclusion in its process a new and elaborate query is created and executed.

3 The Pequliar Reasoner

The Reasoner accepts the output from the Query Processor, and either selects a reasoning rule by itself or by input from the user. The output of the query processor is a collection of entities that are the results of query processing, such as "trucks" recognized by the Query Processor. The Reasoner selects an applicable rule from the following space *S*, which is the Cartesean product of the sub-spaces including sources, objects to be recognized, attributes of objects, time, location and spatial/temporal/semantic relations. In other words,

S = *Source* **x** *Object* **x** *Attributes* **x** *Time* **x** *Location* **x** *Relations*

For example, the Reasoner may need to pick a rule that is applicable to a different source, to recognize a certain type of objects with attributes in certain range, in a certain time interval, for objects in a certain spatial location, and satisfying certain relations. The Reasoner searchers the rule base to select an applicable rule. The rule could be a query template, which is then substantiated and sent back to the Query Processor. If the Reasoner cannot select a rule by itself, either because the rule base is not yet populated or because the space *S* is not properly defined, the Reasoner can accept input from the user. The next time, such rules constructed by the user is remembered, forming part of the scenario.

4 Some Reasoning Issues

The Pequliar Reasoner should be able to carry out a number of different tasks related to a number of different applications that generally are of spatial and/or of temporal character. Applications where the Reasoner may be invoked to support to the users may, for example, include:

- tracking of objects,
- solution of the association problem,
- aggregation of objects,
- prediction of future object behavior in space and over time,
- determination of complex object relationships,
- high level information fusion, e.g. situation analysis.

Determination of the result of these operations is carried out by the Reasoner by means of the learned rules in combination with the meta-data and the available context information. In this way new and more comprehensive queries, of which some may be recursive, can be created from templates in the rule-base. These queries are then executed in the ΣQL query processor. This may lead to a situation that requires a second invocation of the Reasoner that takes place after the comprehensive query have been processed. In this way the Reasoner will be able to learn from the generation of the elaborate queries. However, in the above more comprehensive applications it may not be sufficient to just run a comprehensive query but also to take a step further and perform higher level information fusion, e.g. situation analysis but this is outside the scoop of this work and must be subject to further research efforts.

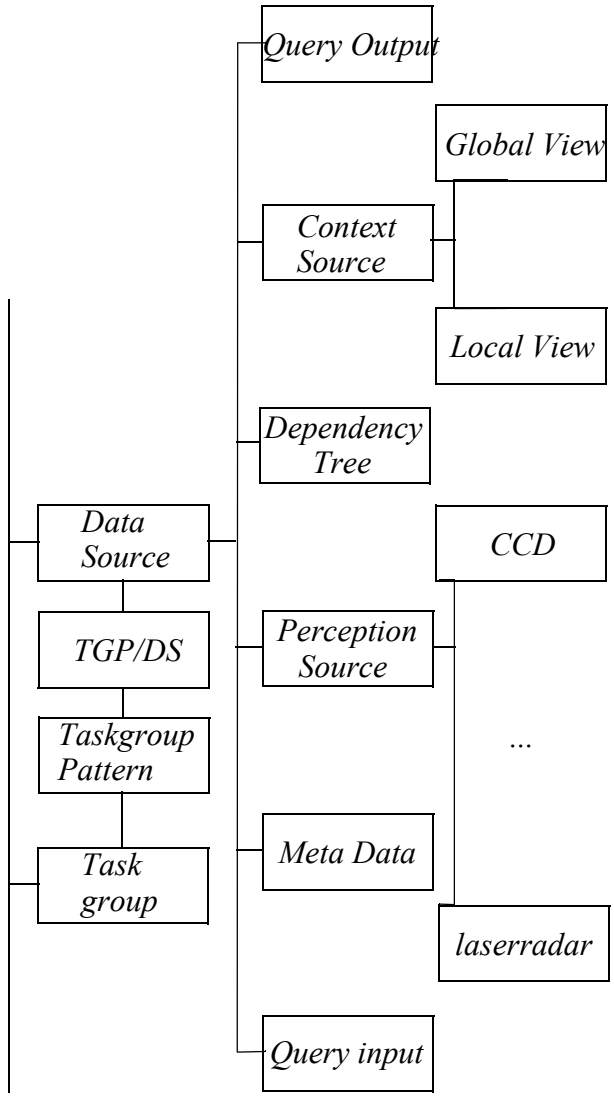


Figure 2. Data sources, task groups and their relations as part of the ontology, to be used for context sensitive object assessment in the Pequliar system.

4.1 Context Sensitive Object Assessment

The background or proper context in which an object may occur and that is subject to a query in ΣQL can be seen as a data source used to enhance the outcome of the query. This was similarly discussed in [3] where the local and the global views were introduced as means for query refinement. Here the two views are jointly called Context Source. An illustration of the use of Context Source to enhance a query result is given in the type I task query example, see section 5.1, where basically the Global View is used to determine whether there are any objects in the proximity of an already retrieved object. As the number of types of queries extends other sources are required as well, which is illustrated in Figure 2.

Query input includes information delivered by the users including such information as area-of-interest, (time) interval of interest and requested object types. In this source, AOI may be used to determine the extension of the context source. Second to this the Meta Data source is used to determine whether there are any data available from any source in Perception Source that correspond to AOI and where Perception Source corresponds to, e.g. data from a sensor. The latter type of sources can be seen as primary sources that are always used as query input. The Dependency Tree that is generated internally to each query can also be used as a data source in cases where queries about query result are of concern. This is illustrated by the task III query type, section 5.3, that asks for pre-fusion sub-results that may be of extreme type. Finally, the result of any given query can be used in an iterative query; this is illustrated by the task I query type. Altogether, the concept of Data Source becomes much more complex when applied to iterative queries generated either in dialogue with or automatically by Pequliar. Although this may cause difficulties in composing iterative queries it also extends the number of possible queries, which in turn, makes ΣQL much more adaptive to complex problems related to, for instance, the higher orders of information fusion, i.e. situation and impact analysis and for this reason it becomes possible to talk about context sensitive object assessment where context refers not just to the proper context but to all the data sources given in Figure 2. The structure in Figure 2 is an extension to the ontology described in [6], which is related to work discussed in [7], [8] that in both cases discuss work where the aim is to develop ontologies for situation awareness. Context Source, refers to the actual geographical background while Perception Source refers to the sensor data sources. Dependency tree is the partial query result from a particular sensor data source. The remaining data sources are self-explanatory.

4.2 Deriving Query Patterns from Query Paths

As mentioned above, the user formulate a query by composing compound queries from elementary queries using query operators. But the problem is that for a compound query, this may make the query building time consuming and boring. An approach to speed up this process is proposed based upon two concepts: *query path* and *query pattern*. A query path $\boldsymbol{p}$ is a sequence of queries (Q_0, Q_1, Q_2, ……, Q_n) in the iterative query formulation process. Basically a query path shows the track of an interactive query construction from Q_0 to Q_n. In each step, a query operator and an elementary query are applied on the previous one. While a query pattern is generalized from query paths, some constant values in the queries may be marked as variables to indicate that these values are exchangeable while doing a pattern matching. Query patterns are important because they could cover most of the frequently used queries and make them simple. In our approach, a set of query patterns $S_{pattern}$ is stored and maintained during the query processing. At the initial state, there are no query patterns in $S_{pattern}$. When one query is finished,

the query path will be shown and user is asked to choose the variables and input a linguistic tag for it. A new query pattern is generated and added to $S_{pattern}$. In query formulation, the query patterns are retrieved dynamically and then selected by the user. Therefore the Pequliar Reasoner can learn new patterns, and use the patterns to provide short cuts for the user to formulate queries.

5 Tasks for Information Fusion

In what follows we describe the typical tasks for information fusion, which can be grouped into three types. The reasoning process as carried out here depends on whether the belief values that are output from any user query has got a value that is uninformative. Three different types of tasks queries have been identified. The first type is concerned with how to improve the result of the original query by considering other aspects of the sensor data sources such as for instance similarity among requested objects but also proximity between the objects. The second type is merely concerned with how to associate different object instances with each other while the third group concerns queries that need to inspect the dependency tree from the last user query. All these three task types are related in that they will result in new elaborate queries that are part of the iterative information fusion process that also includes learning of rules in that process. Here we also demonstrate the three task types with some relevant examples of elaborate ΣQL queries.

5.1 Type I Tasks

These tasks require the generation of new and elaborate ΣQL queries that basically depends on certain conditions among the spatial objects normal found in sensor data.

1) Are there any other objects in the proximity of the retrieved object that are of similar or of equal type as the retrieved object?
   Proximity refers to the AOI and *similarity* to the ontology; the Reasoner creates new elaborate queries from the templates.
2) Have there been any other objects in the proximity of the retrieved object that are of similar or of equal type as the retrieved object?
   The Reasoner creates new elaborate queries from the templates.
3) Is the present background (context) information of proper type relative to the type of the retrieved object?
   Context refers to the geographical background in the AOI and the Reasoner creates new elaborate queries from the templates.
4) Has this object type been previously observed in this background context?
   This query is similar to 3 but require involvement of earlier output instances from ΣQL, i.e. the IOI must be involved as well.
5) Is it possible that the quality of the background information may have influenced the query result?
   For instance is there any missing data in AOI. This could be determined from a particular ΣQL query. For a lot of missing data the risks for not finding a particular object increases.
6) Is the retrieved object partly hidden by some other object that is part of the context?
   The context refers to the local background that must be in high resolution. This allows the Reasoner to create a new elaborate query.
7) Are there any other object types in the proximity of the retrieved object that may have any *impact* of some kind on the found object?
   This query is similar to task type 6 but here the object may have a location that is too close to some other object which will have some more or less serious consequences on the primary object

5.2 Type II Tasks

These task type requires generally the invocation of a particular function that basically is concerned with the resolution of the association problem that may occur in single cases, that is between a pair of registered objects, or as a part of a tracking task, including a time sequence of the registered objects.

8) Can the retrieved object be associated to an earlier single observation?
   This query type requires the solution of the association problem.
9) Can the retrieved object be part of an existing track.
   This task type is a recursive variation of query type 8 that includes the application of the association problem but will not be further elaborated here.

5.3 Type III Tasks

This task type requires only investigation of the result of the various sub-queries of user defined queries, that corresponds, in most cases, to an inspection of the content of the dependency tree

10) Did any sensor (data sources) contribute to the result in any extreme way?
    This refers to the single belief values from the various sensor related sub-queries; only a check of the dependency tree is required.
11) Did the result of some of the sensors (data sources) contradict each other?
    Does not require any new ΣQL query; only an inspection of the dependency tree.
12) Which sensors (data sources) where used to answer the query?
    Does not require any new ΣQL query; only an inspection of the dependency tree.
13) Are there any attributes or status values of the retrieved object that *in particular* could have diverted the outcome of the primary query.
    This means that the attribute did not in any case contribute to the query result; contrary it could

have contributed to another outcome of the query.

6 Examples of Iterative Queries

In this section a set of iterative queries generated by the Reasoner will be shown. These queries are examples of the three task types introduced in Section 5. Most of these queries require two iterations although it is possible to refine query formulation into three or even four iterations, depending upon how the query formulation is done by the user. In the query examples *relation* refer to topological relations between a pair of spatial objects of which AOI is also considered being a spatial object.

Example 1: The following ΣQL query corresponds to Query 1 of Task Type 1. Initially, the user tries to find "trucks" in a certain area of interest. This corresponds to the light grey colored ΣQL query. If the results are uninformative, the user may guide the Reasoner to apply a more elaborate query as shown below. After the elaborate query is processed, the user is satisfied with the results. He can then tell the Reasoner to remember the rule under a certain user-assigned task description, such as "objects similar to trucks".

Query: *Are there any other objects in the proximity of the retrieved object that are of similar or of equal type as the retrieved object?*

```
Select object_k.type, object_k.position, object_j.type
cluster * alias object_k
from PerceptionSource
where relation(AOI, object_k) = 'inside'
   and object_k.type = 'truck'
   and object_j.t = object_k.t
   and distance(object_k, object_j) < δ
   and similar(object_k, object_j)
   and object_j in
       Select object_i.type, object_i.position
       cluster * alias object_i
       from PerceptionSource
       where relation(AOI, object_i) = 'inside'
          and object_i.t = t_given
          and object_i.type = 'truck'
```

Example 2: The following ΣQL query corresponds to Query 2 of Task Type I. As in the previous example, initially the user tries to find "trucks" in a certain area of interest. This corresponds to the light grey colored ΣQL query. The user may then decide to guide the Reasoner to apply a more elaborate query as shown below to find similar objects in previous time periods. After the elaborate query is processed, the user is satisfied with the results. He can then tell the Reasoner to remember the rule under a certain user-assigned task description, such as "objects similar to trucks in previous time periods".

Query: *Have there been any other objects in the proximity of the retrieved object that are of similar or of equal type as the retrieved object?*

```
Select object_k.type, object_k.position, object_k.t,
       object_j.type, object_j.position
cluster * alias object_k
from PerceptionSource
where relation(AOI, object_k) = 'inside'
  and distance(object_k.position,object_j.position)
      < δ
  and similar(object_k.type, object_j.type)
  and t_start < object_i.t < object_k.t
  and object_j in
      Select object_i.type, object_i.position
      cluster * alias object_i
      from PerceptionSource
      where relation(AOI, object_i) = 'inside'
        and object_i.t = t_given
        and object_i.type = 'truck'
```

Example 3: task type I query 3

Query: *Is the present background (context) information of proper type relative to the type of the retrieved object?*

```
Select object_j. type,object_j.position,object_p.type
cluster * alias object_p
from ContextSource
where relation(object_p, object_j) = 'inside'
  and properbackground(object_p.type,
                      object_j.type)
  and object_j in
      Select object_i.type, object_i.position
      cluster * alias object_i
      from PerceptionSource
      where relation(AOI, object_i) = 'inside'
        and object_i.t = t_given
        and object_i.type = 'bus'
```

Example 4: task type I query 4

Query: *Has this object type been previously observed in this background context?*

```
Select  object_p.type, object_i.type,
        object_i.position
cluster * alias object_p
from ContextSource
where object_j.type = object_p.type
  and object_i.type = object_k.type
  and t_start < object_i.t < object_k.t
  and object_i in
        Select object_m.type, object_m.t,
               object_m.position
        cluster * alias object_m
        from PerceptionSource
        where relation(AOI, object_m) =
              'inside'
          and t_start < object_m.t < t_given
          and object_m.type = 'bus'
  and object_j in
        Select object_l.type
```

```
    cluster * alias object_l
    from ContextSource
    where relation(AOL, object_l) =
          'partlyoverlap'
       and object_k in
          Select objedct_n.type,
                 object_n.position
          cluster * alias object_n
          from PerceptionSource
          where
             and relation(AOI, object_n) =
                 'overlap'
             and object_n.t = t_given
             and object_n.type = 'bus'
```

Example 5: task type query 5

Query: *Is it possible that the quality of the background information may have influenced the query result?*

```
Select object_p.sensor
cluster * alias object_p
from PerceptionSource
where relation(object_p.background, object_j) =
      'inside'
and missingdata(object_p.background) > 50
    /* more than 50% of the background data
     may be missing */
and object_p.t = t_given
and object_j in
   Select object_i.type, object_i.position
   cluster * alias object_i
   from PerceptionSource
   where relation(AOI, object_i) = 'inside'
      and object_i.t = t_given
      and object_i.type = 'bus'
```

Example 6: task type I query 6

Query: *Is the retrieved object partly hidden by some other object that is part of the context?*

```
Select object_p.type, object_p.position,
       object_j.type, object_j.position
cluster * alias object_p
from ContextSource
where relation(AOI, object_p) = 'partlyoverlap'
   and ((object_p.type = 'terrain-feature')
     or (object_p.type = 'building')
     or (object_p.type = 'natural-object))
   and ((relation(object_p, object_j) =
         'partlyoverlap')
     or (relation(object_p, object_j) = 'beside'))
   and object_p.t = object_j.t
   and object_j in
      Select type
      cluster * alias object_i
      from PerceptionSource
      where relation(AOI, object_i) = 'inside'
         and object_i.t = t_given
         and object_i.type = 'bus'
```

Example 7: task type I query 7

Query: *Are there any other object types in the proximity of the retrieved object that may have any impact of some kind on the found object?*

```
Select object_k.type, object_k.position,
       object_j.type, object_j.position
cluster * alias object_k
from PerceptionSource
where relation(AOI, object_k) = 'inside'
   and distance(object_k.position,
                object_j.position) < δ
   and object_k.type = 'dog'
   and non-coexistant(object_k, object_j)
   and object_k.t = object_j.t
   and object_j in
      Select object_i.type
      cluster * alias object_i
      from PerceptionSource
      where relation(AOI, object_i) = 'inside'
         and object_i.type = 'cat'
         and object_i.t = t_given
```

Example 8: task type II query 8

Query: *Can the retrieved object be associated to an earlier single observation?*

```
Select object_k.type, object_k.t, object_k.position,
       object_j.type, object_j.t, object_j.position
cluster * alias object_k
from PerceptionSource
where relation(AOI, object_k) = 'inside'
   and distance(object_k.position,
                object_j.position) < δ
   and t_start < object_k.t < object_j.t
   and associate(object_k, object_j)
   and object_j in
      Select object_i.type, object_i.position
      cluster * alias object_i
      from PerceptionSource
      where relation(AOI, object_i) = 'inside'
         and object_i.t = t_given
         and object_i.type = 'truck'
```

Example 10: task type III query 10

Query: *Did any sensor (data source) contribute to the result in any extreme way?*

```
Select object_k.type, object_k.sensor,
       object_k.belief-value, object_k.position
cluster * alias object_k
from DependencyTree
where qualitative-difference
         (object_k.belief-value,
         object_j.belief-value) = 'large'
   and object_k.position = object_j.position
   and object_k.type = object_j.type
   and object_k.t = object_j.t
   and object_j in
      Select object_i.type, object_i.position
```

```
        cluster * alias object_i
        from PerceptionSource
        where relation(AOI, object_i) = 'inside'
           and object_i.t = t_given
           and object_i.type = 'truck'
```

Example 11: task type III query 11

Query: *Did the final result contradict the result from any of the sensors (data sources)?*

```
Select object_k.type, object_k.sensor,
       object_j.type, object_j.position
cluster * alias object_k
from DependencyTree
where object_k.position = object_j.position
   and object_k.type ≠ object_j.type
   and object_k.t = object_j.t
   and object_j in
      Select object_i.type, object_i.position
      cluster * alias object_i
      from PerceptionSource
      where relation(AOI, object_i) = 'inside'
         and object_i.t = t_given
         and object_i.type = 'truck'
```

Example 12: task type III query 12

Query: *Which sensors (Perception Sources) where used to answer the query?*

```
Select object_k.sensor
cluster * alias object_k
from DependencyTree
 where object_k.position = object_j.position
    and object_k.type = object_j.type
    and object_k.t = object_j.t
    and object_j in
       Select type
       cluster * alias object_i
       from PerceptionSource
       where relation(AOI, object_i) = 'inside'
          and object_i.t = t_given
          and object_i.type = 'truck'
```

7 Discussion

In this work a reasoning system for objects with unimformative belief values has been introduced. The Reasoner can be viewed as a post query language processor, since the input to the Reasoner is the output from ΣQL a query language, primarily, developed for data from multiple sensors. An important characteristic of ΣQL is that it allows sensor data fusion and includes an ontology that allows application of queries in a sensor data independence way, that is a user does not have to know which sensors that are involved when a query is answered. In particular, to allow the extension of the queries the query processor has been extended with respect to the data sources. This is mirrored by the ontology that demonstrates how the context information as well as the dependency tree information can be used as input to the refined queries. This is done in the same manner as the perception sources, which were introduced in the original approach to ΣQL, are used. The extended query technique is, finally, demonstrated with a number of elaborate queries that demonstrates the technique with three different task type queries that correspond to pattern that can be determined by the Reasoner or alternative by the user. This illustrates how the Reasoner eventually can learn how to automatically apply the more elaborate queries.

In connection to this work there are also other means that need to be studied further, such as generation of even more elaborate queries to support applications related to, e.g. spatial data mining.

References:

[1] Chang, S.-K. and Jungert, E.: Human and system directed fusion of multimedia and multimodal information using the σ-tree data model. Proceedings of the 2nd International conference on Visual information systems, San Diego, CA, December 15-17 (1997) 21-28.

[2] S.-K. Chang, G. Costagliola, E. Jungert, Spatial/ Temporal Query Processing for Information Fusion Applications, Advances in Visual Information Systems, R. Laurini (Ed.), Lecture Notes in Computer Science 1929, Springer Verlag, Berlin, 2000, pp 127-139.

[3] S.-K. Chang, G. Costagliola, E. Jungert, Multisensor Information Fusion by Query Refinement, Recent advances in Visual information systems, S.-K. Chang, Z. Chen, S.-Y. Lee (Eds.), Lecture Notes in Computer Science 2314, Springer Verlag, Berlin, 2002, pp 1-11.

[4] S. K. Chang, E. Jungert and G. Costagliola, "Queryng Distributed Multimedia Databases and Data Sources for Sensor Data Fusion", to appear in IEEE Transactions on Multimedia, 2004.

[5] [Handbook of Multisensor Data Fusion, D. L. Hall, J. Llinas (Eds.), CRC Press, London, 2001.

[6]] Horney, T., Jungert, E., Folkesson, M., *An Ontology Controlled Data Fusion Process for Query Language,* Proceedings of the International Conference on Information Fusion 2003 (Fusion'03), Cairns, Australia, July 8-11.

[7] Matheus, C. J., Kokar, M., Baclawski, K., *A Core Ontology for Situation Awareness*,Proceedings of the International Conference on Information Fusion 2003 (Fusion'03), Cairns, Australia, July 8-11.

[8] McGuinness, D. L., *Ontologies for Information Fusion,* Proceedings of the International Conference on Information Fusion 2003 (Fusion'03), Cairns, Australia, July 8-11, pp 650-657.

[9] E. Waltz and J. Llinas, Multi-sensor data fusion, Artect House, Boston, 1990.

[10] K. Silvervarg, E. Jungert, *Aspects of a visual user interface for spatial/temporal queries*, proceedings of the 9th international conference on Distributed Multimedia Systems, Miami, September 24-26, 2003,pp 287-293.

[11] S. Chawla, S. Shekhar, W.-L. Wu, U. Ozesmi, *Modeling spatial Dependencies for mining geospatial data: An introduction*, H. Miller, J. Han (Eds.), Geographic Data Mining and Knowledge Discovery (GKD), Taylor and Francis, 1999.

Appendix G

A Fusion Framework for coarse-to-fine Target Recognition

Förelagd Defense and Security Symposium, March 29-31, 2005, Orlando, Florida, USA

Folkesson, M., Grönwall, C., Jungert, E.

A Fusion Framework for coarse-to-fine Target Recognition

Martin Folkesson
Swedish Defence Research Agency (FOI)
Box 1165
581 11 Linköping
martin.folkesson@foi.se

Christina Grönwall
Swedish Defence Research Agency (FOI)
Box 1165
581 11 Linköping
christina.gronwall@foi.se

Erland Jungert
Swedish Defence Research Agency
Box 1165
581 11 Linköping
erland.r.a.jungert@foi

Abstract – *A fusion framework in a query based information system is presented. The system is designed for querying multimedia data bases, and here applied to target recognition using heterogeneous data sources. The recognition process is coarse-to-fine, with an initial attribute estimation step and a following matching step. Several sensor types and algorithms are involved in each of these two steps. Fusion is performed after both steps. An independence of the matching results, on the origin of the estimation results, is observed. This is the basic idea of the fusion framework. Its aim is to increase the overall performance in target recognition. An implementation of the system is described.*

Keywords: Fusion, information system, target recognition, query language, attribute, knowledge system

1 Introduction

The system described in this paper is based on querying multimedia data bases. Results on such queries may then consist of several partial results, originating from different sets of data. To obtain a total result for the query, the partial results need to be *fused*. If the separate results come from processes that are divided into two steps, data can be fused also at an intermediate level. This is done in this application of our system, and illustrated in fig. 1. The intermediate fusion step allows information to be shared between the different processes. This potentially increases the overall performance of the system. The intermediate fusion unfortunately also introduces a risk of *data incest*. Data incest is present when dependent data sources are assumed to be independent. An example of data incest between two sensors is given in [1]. In the application of this paper, the risk of data incest is avoided.

1.1 Related work

Traditionally, the role of fusion in different automatic target classification/recognition systems has been to combine results that has emerged from independent processes. For example, Tseng et. al. [2] examplifies such an approach. However, lower-level fusion of heterogenous data in ATR systems has been approached recently, see e.g.[3]. In [4], a comparison between fusion on different levels is performed. Strategies for dealing with incomplete data has been adressed in connection with clustering problems, see [5], [6].

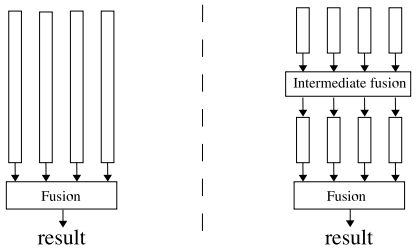


Fig. 1: Left: Fusion is applied to separate, partial outputs. Right: If the separate processes are divided into two steps, an intermediate data fusion step, potentatially increasing the overall performance, can be added.

2 Preliminaries

The system is here designed for recognition of ground targets, mainly military vehicles. Various types of electro-optical sensors are used for the recognition part. The recognition is divided hierarchically into two steps. The first step estimates attributes of a detected target. The second step is the actual recognition, which is performed by matching the data against target models. The model matching is very time-consuming compared to the attribute estimation. Feeding a matching algorithm with estimates of attributes reduces its work load significantly. Hence the introduction of attribute estimation.

The intermediate fusion step is introduced to fuse data from different attribute estimation processes. It has the potential of reducing the total number of matching processes that needs to be performed in the matching step. Also, if the matching processes fail, or are considered too time-consuming in a stressed situation, the fused attribute estimations could be taken as output from the system.

Our recognition process is now a sequence of four steps; attribute estimation, attibute fusion, model matching and model match fusion, see fig. 2.

To complete the information system presented here contains not only query processing, a visual user interface (VUI), and fusion, but also algorithms for attribute estimation and model matching, respectively. Also, an intelligent mechanism for choosing appropriate algorithms for the two steps, in a given situation, is needed. This mechanism must also be able to select sensors for *collecting* data, if this needs to be done before recognition. In our system this

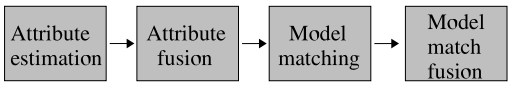


Fig. 2: The recognition is divided into attribute estimation and model matching, each followed by a fusion step.

mechanism is a *knowledge-based system* (KBS), central in the application. It has access to *meta data* about stored sensor data. An overview of the system is found in fig. 3. The different parts will be further described in section 3.

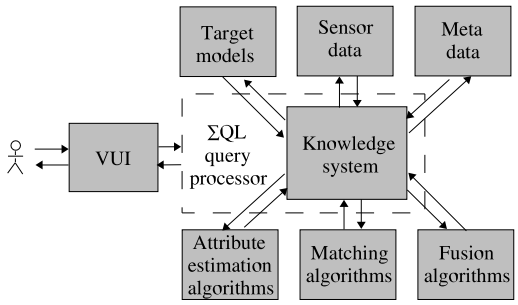


Fig. 3: An overview of the information system when applied to target recognition.

2.1 Query execution

As the user queries the system, he/she is supposed to specify 1) which types of targets he/she is interested in, 2) in which area, and 3) during which interval in time. Then, a detection algorithm is applied to the relevant data. Each of the detected, possible targets then induces a *refined* query, where the regions of interest are small areas around each detected target. The set of refined queries are then further refined, by the attribute estimation. After the estimation, the queries include information on target dimensions, orientations, etc.

The next step is to select target models compatible with the attribute estimates of each query. Each query is thus again refined and possibly degenerated, into several queries, one for each compatible target model. Each of the resulting queries then continues to the matching step, to be evaluated against the data. A schematic of the refinement steps of a query is seen in fig. 4. In the steps where queries possibly degenerate, a degeneration into two queries has here been chosen. Note that the fusion steps have been omitted in fig. 4.

2.2 Problem description

The fusion challenge is to fuse output from the algorithms performing attribute estimation and model matching, respectively. The aim is to increase the overall chance of recognising the target, and reduce the work for the matching algorithms.

For each of the two processes, several algorithms are generally working in parallell. A certain algorithm could

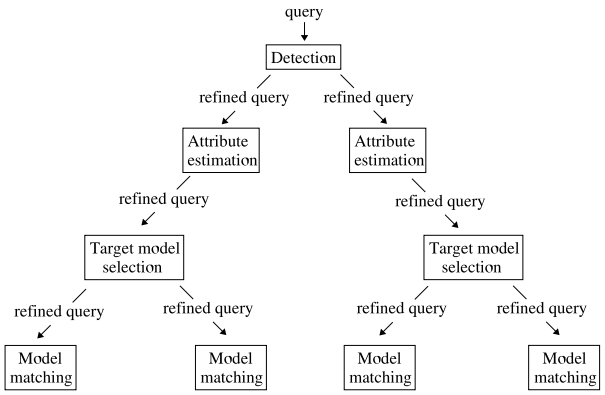


Fig. 4: The three main refinement levels, as a query is processed. In the first and third step, the query can degenerate into two (as in the figure) or more queries. In the model matching, the queries are finally evaluated. Note that the fusion steps have been omitted in the figure.

be applied to different sets of data. Thus, the fact that correlations of different types might be present has to be kept in mind.

Another complication is that different combinations of algorithms and data estimate different sets of attributes. Each produced set of attribute estimates is generally *incomplete*. Typically, some sensors are not capable of estimating certain attributes. For example, heat radiation from a vehicle with a running engine does not show in visible light images. Thus, whether the vehicle's engine is running or not, cannot be detected in visible light (unless exhaust fumes show).

3 The implemented Target Recognition System

This section presents the most relevant parts of the implemented system, see fig. 3. A more thorough description of the system as a whole is given in [7] and [8]. Some parts of the system have been ignored here, others combined under the same heading. References to papers devoted to the separate parts are given in corresponding subsections below.

3.1 User interaction — the query language ΣQL

The user interaction framework consists of a *query processor* and a VUI — *Visual User Interface*. The query language used to process the query is however also the framework of the entire system. In this section, the query language is therefore given some special attention. For a description of the VUI, see [9].

The query language discussed in this work is called ΣQL [10],[11]. Originally, ΣQL was developed as a query language for querying of one single sensor image at a time. However, later it has evolved into a query language for multiple sensor data sources, and capable of sensor data fusion and *sensor data independence* (see below). The various sensors may be of different types and generate heterogeneous sensor data images. For this reason a large number of algorithms for sensor data analysis must be available and administered by the system.

It is required that selection of sensors and algorithms must be carried out automatically, and for this reason means for such selection must be available. In ΣQL this is controlled by the KBS, which is briefly described in section 3.2. The reason for the automatic selection is to allow sensor data independence. In a system, sensor data independence denotes independence between the low-level sensor data processing and the high-level user interaction. Sensor data independence is motivated for several reasons. One is to allow the user to concentrate on the work at hand, without any knowledge of the sensors and their data types. Another motivation is to make *repetitive queries* possible without interference from the users. Repetitive queries are queries the system repeateadly poses to itself. Triggering of such repetition is called for when i.e. light and weather conditions change during the period of time defined by the user (see 2.1). Each time the query is automatically repeated, the selection of data sources is performed all over again. The system thus has a way of adapting to changing conditions.

The basic functionality of ΣQL can be described as follows: The query inserted by the user is forwarded to the *dependency tree generator*, which in a dialogue with the KBS generates a set of nodes in the *dependency tree*. As long as there are nodes in this tree new sub queries can be built (one for each of the selected sensors), refined and executed by the query processor. Once the sub queries have been executed, instances are created in the multilevel view database [12], to support query refinement in a subsequent step. As new sets of dependency tree nodes are generated new sub queries can be executed and their results refined. This goes on until the dependency tree becomes empty, i.e. when there is no more sensor data available to process.

3.2 KBS and meta data

The KBS selects data sources and algorithms, both for the attribute estimation and for the matching. Meta data, describing which kinds of sensor data are available from which areas during which periods in time, are used to make sure that only relevant data sources are selected. In the same manner, only algorithms which can be applied to available data are selected. Several factors and conditions are hereby considered, both internal to the system, and external, such as weather conditions. The KBS also selects models based on the attribute estimates, i.e. it translates sets of attribute estimates into sets of models compatible with the estimates. The KBS can be seen as the center of the system in this application. and is described further in [13].

3.3 Sensor data

We consider three fundamentally different types of sensor data; 2D images from visual and IR camera, 3D point scatter from laser radar and sequences of range images from a gated viewing sensor (a type of laser radar). Three sensors, one longwave infrared (LWIR) sensor, one scanning laser radar operating in the near infrared (NIR), and one visual CCD-camera, were mounted on an airborne platform. The laser radar system's scanning constitutes a zigzag pattern on the ground. The resulting data is in point scatter format, containing 3D position (x, y, z) and reflected intensity (r) in each sample, i.e., the data is an unordered set of samples (x, y, z, r). The sensor systems and the data collection is further described in [8].

These sensor data are heterogeneous concerning dimensions of the target (2D or 3D registration), format (matrix format or irregular sampling), the resolution of the target (i.e, samples/m^2), the texture of the target (three different wavelengths registrating different phenomena), the possibility to track moving targets (the laser radars are scanning and thus not suitable) and views of the target (down-looking or horizontally).

3.4 Algorithms

Recall that the target recognition process is performed in four steps; attribute estimation, attribute fusion, model matching and model match fusion, see fig 2. There are two attribute estimation algorithms working on four types of sensor data. Typical attributes are position, orientation, length, width, height, speed, and temperature. Each estimated attribute is given with an estimated standard deviation, corresponding to the noise level in data and the performance of the algorithm.

The system contains five model matching algorithms working on five types of sensor data. In the model matching step the common target model library is used, where each model is described by its 3D structure (facet/wire frame models) and appearance (visual or thermal textures). Based on the operator's query and the estimated attributes, a set of target models are selected for the matching process. The output is a quantitative degree of match between the model and the sensor data pointed out by the query. This quantitative measure is called *confidence value* (cv). To be able to compare cv:s, the cv is normalised to a value between 0 and 1, where 1 means perfect match. The normalisation is based on matching results of important target data sets. Note however that no exact definition of the cv can be given. The way of obtaining a cv can be different for different combinations of data and algorithm.

4 Fusion approaches

This section presents and motivates the fusion framework for the two fusion steps, see fig 2. The methods themselves are rudimentary and should be considered parts of a framework rather than algorithmic contributions.

4.1 Fusion of attribute estimates

For the first process step, the coarse estimation of attributes, the fusion approach assumes that *the sets of estimates are naturally grouped in attribute space*. This corresponds to a discretisation of possible initial interpretations of the data. To motivate this assumption, consider a set of estimates of a target's orientation on the ground. Values are likely to be grouped around two directions that differ by π rads. Furthermore, consider a situation where the target has a large cubic-shaped rock just behind it. One or more estimates of the target's length might then include the rock. Again, two distinct groups of values would be formed. Combinations

of such estimates for orientation and length are then also distinct. Only one combination corresponds to the correct interpretation of the image data.

Since some attributes of a target normally changes over time, like its orientation on the ground, it is required that data from different sources are aligned in time. If not, an association problem, which is not adressed in this paper, has to be solved beforehand.

The fusion approach chosen is simply to *identify distinct interpretations/groups of sets of estimates*. We loop over the sets and identify groups of compatible sets. Two sets are considered compatible iff, for each attribute, their respective estimates, including estimated deviation, are non-disjunct. For a set of attributes $\{a_i\}$, estimated by algorithms j_1 and j_2, this is written

$$\begin{aligned} \forall\, i;\, [\hat{a}_{i,j_1} - \hat{\sigma}_{a_{i,j_1}}, \hat{a}_{i,j_1} + \hat{\sigma}_{a_{i,j_1}}] \\ \cap\, [\hat{a}_{i,j_2} - \hat{\sigma}_{a_{i,j_2}}, \hat{a}_{i,j_2} + \hat{\sigma}_{a_{i,j_2}}] \neq \emptyset, \\ \Rightarrow\ (\hat{a}_{j_1}, \hat{a}_{j_2})\, compatible \end{aligned} \tag{1}$$

Here, $\hat{a}_{j_1}$ and $\hat{a}_{j_2}$ denote the arrays of attribute estimates provided by the two algorithms, respectively, and $\hat{\sigma}_{j_1}$ and $\hat{\sigma}_{j_2}$ are the corresponding arrays of estimated standard deviations. Compatible sets of estimates are considered qualitatively equivalent interpretations of the data.

If an array of estimates is incomplete, the lacking estimates are set to 0, with infinite deviation, obtaining the interval $[-\infty, \infty]$. This interval is compatible with every true estimate for that attribute. The criterion of compatibility is thus weak for arrays of estimates with a high degree of incompleteness.

When different groups of estimates are identified, we seek to represent each group with a single set of estimates, in clustering often called a *prototype*. Here, *we calculate the group mean estimate for each attribute*. These mean values are then combined into a prototype for the group. Only true estimates are considered. An example for a group with two sets of estimates, containg four attributes, is given below.

Sets of estimates:

$$\begin{pmatrix} \hat{a}_{1,j_1} \pm \hat{\sigma}_{a_{1,j_1}} \\ \hat{a}_{2,j_1} \pm \hat{\sigma}_{a_{2,j_1}} \\ -\infty, \infty \\ \hat{a}_{4,j_1} \pm \hat{\sigma}_{a_{4,j_1}} \end{pmatrix} \begin{pmatrix} -\infty, \infty \\ \hat{a}_{2,j_2} \pm \hat{\sigma}_{a_{2,j_2}} \\ \hat{a}_{3,j_2} \pm \hat{\sigma}_{a_{3,j_2}} \\ \hat{a}_{4,j_2} \pm \hat{\sigma}_{a_{4,j_2}} \end{pmatrix}$$

Prototype:

$$\begin{pmatrix} \hat{a}_{1,j_1} \pm \hat{\sigma}_{a_{1,j_1}} \\ \frac{\hat{a}_{2,j_1}+\hat{a}_{2,j_2}}{2} \pm \sqrt{\frac{\hat{\sigma}^2_{a_{2,j_1}}+\hat{\sigma}^2_{a_{2,j_2}}}{2}} \\ \hat{a}_{3,j_2} \pm \hat{\sigma}_{a_{3,j_2}} \\ \frac{\hat{a}_{4,j_1}+\hat{a}_{4,j_2}}{2} \pm \sqrt{\frac{\hat{\sigma}^2_{a_{4,j_1}}+\hat{\sigma}^2_{a_{4,j_2}}}{2}} \end{pmatrix}$$

4.2 Fusion of matching results

Each prototype consist of estimates of attributes of two categories. Those attributes concerning the target's *properties* are constants, like 'size'. On the other hand, those attributes concerning its *state* are variables, like 'orientation'. The KBS checks which modelled targets are compatible with the estimated properties. Each such model is then combined with the estimated state. The model replaces the estimates of properties. For each matching, the matching algorithm is given a specific target model and estimates of state attributes, instead of estimates describing both target properties and target state.

We call each combination of target model and state attributes a *hypothesis*. A hypothesis is an ultimately refined query, see section 2.1. Thus, each prototype governs one or more hypotheses, via the target model selection (see fig. 4).

The KBS sends each hypothesis to its chosen set of algorithms (and data), for evaluation/refinement by matching. The important observation to be made here is that *each matching process is independent of the origin of the hypothesis*. In other words: Given a certain hypothesis, with certain estimated attribute values, it does not matter which data sources were used to obtain the values. The quantified match result — the cv — will not depend on the origin of the estimates in the hypothesis. Thus, there should be no risk of data incest in this framework.

Each match result is an evaluation of a certain model, in a certain state, made by a certain combination of data and algorithm. The algorithm locally optimises the cv by refining the state attribute estimates. It is assumed that this refining is small, so that different refined states from the same hypothesis are compatible, hence considered qualitatively equivalent (see section 4.1). Thus, when considering the obtained set of cv:s, we can make inference over each of these three "dimensions"; match model, state and data-algorithm-combination. Compare the situation to sorting of objects in a data base, based on values of their different attributes.

The method currently used is to simply *select the best match result for each model, regardless of state*, and present to the user. This is motivated by the assumption that the user is primarily interested in the type of the target, rather than its state. The choice to select the *best* match results for each model, instead of calculating e.g. means, is motivated two-foldly:

- As stated above, correlations between data and algorithms are unknown, thus mean values could be misleading.

- High cv:s are in a sense, more trusteable than low. Low could be explained not only by a bad (wrong) hypothesis, but alternatively by noise. High cv:s should be more rare, hence more significant.

The selected match results are sorted by descending cv before presentation. Also, the user could choose to threshold low cv results out, upon querying the system.

Supplying the user with a *list* of match results, instead of a single one, can also be two-foldly motivated:

- Similar situations has shown that it can be cruicially important to be informed about uncertainties in

classification/recognition results, see [14].

- The user can make a *relative* interpretation of the cv:s, which should be easier than interpreting a single value (see section 6.2).

4.2.1 Correlations

This section considers possible correlations between cv:s, and further motivates the choice of the simple match fusion method presented above.

As a certain algorithm could handle different data sets, one can not assume that cv:s obtained in different *data* are independent. In the same way, a certain set of data could be accessed by more than one algorithm. Hence one can not assume that cv:s obtained by different *algorithms* are independent. Furthermore, algorithms themselves could be systematically correlated. This might also be the case for sensors and their correponding data sets.

All possible correlations of the above can not be investigated in advance, and built into a match fusion method. This is due to the fact that the system is intended to flexibly add new data sources and algorithms, and combine them into data-algorithm-combinations in runtime. This is done without investigations of correlations and without update of the match fusion method. Instead, it is believed that fusion methods should be as robust aginst possible correlations as possible.

5 Example

In this section, we will describe in detail how a query is processed until a final answer is given to the user. The IR and 3D data from the airborne system, described in section 3.3, are used. The 3D data is treated both in its raw format (3D point scatter and NIR reflectance data) and transformed to a digital elevation model (DEM). Thus, we have four different types of sensor data. Below we will analyse the information retrieval concerning three vehicles placed in open terrain.

Assume that the user enters the query "Report all main battle tanks (MBTs) present in the one kilometer wide area around the position (500, 500)", where the specified area covers targets A, B, and C. Since all tanks in the target model library are within the dimensions $7.4\pm0.52 \times 3.6\pm0.2$ meters, these values are inserted into the query. The query is then sent to the detection algorithm, which returns three queries, one for each detected target in the area, see table 1. The three queries are each sent to the four available sensor nodes (3D, DEM, NIR, and LWIR) for attribute estimation. Twelve queries are returned from the attribute estimation, see table 2.

The results from the attribute estimation are then subject to attribute fusion. The set of attributes, see Table 2, is analysed using equation (1). The fusion of the attribute estimates to prototypes is processed as below:

1. All attribute estimates regarding target A are consistent and fused to one prototype of a target being approximately 6.78×3.78 meters in size.
2. All attribute estimates regarding target B are consistent and fused to one prototype of a target being approximately 7.50×2.79 meters in size.
3. Two attribute estimates regarding target C (LWIR and DEM) are consistent and fused to one prototype.
4. The 3D attribute estimate regarding target C cannot be grouped with any other attribute estimate and thus forms a separate prototype.

Thus, the result from attribute fusion is four prototypes, see Table 3. Note this reduction in complexity. These prototypes are returned to the KBS. Only one of the prototypes, describing target A, is consistent with the MBT class. All target models consistent with this prototype are extracted from the target model library, and for each extracted model, a query is sent to the model matching step. In this particular case, two models (representing two different MBTs) are found. Since both LWIR data and 3D data are available, two model matching algorithms are invoked.

From the model matching four evaluated hypotheses are returned. Some results are shown in Table 4. With the model match fusion currently used the hypotheses are grouped per target type. The hypothesis are sorted for each model on confidence value and the query with the highest confidence value for each model is stored. In this example, one query for the T72 model and one for the Leclerc model is stored, see Table 5. These are returned to the user as the final result. The two suggestions, with corresponding cv:s, are hence the decision support given to the user.

Table 1: The input query for detection and the output queries. Empty fields are omitted (sensor, type, confidence value, height, speed and temperature).

Field	Input	Target A	Target B	Target C
Class	MBT	MBT	MBT	MBT
Position	(500, 500)±500	(560, 400)±20	(620, 460)±20	(590, 440)±20
Orientation (deg)	0–360	0–360	0–360	0–360
Length (m)	6.88–7.92	6.88–7.92	6.88–7.92	6.88–7.92
Width (m)	3.40–3.80	3.40–3.80	3.40–3.80	3.40–3.80

Table 2: Result of attribute estimation.

Target	Sensor data	Estimated	Estimated	Estimated
True class		orientation	length	width
True Dimensions		(deg.)	(m)	(m)
A	3D	316.1±2.3	6.34±0.82	3.49±1.02
MBT (T72)	DEM	317.1±4.1	6.74±0.58	3.70±0.58
7.13×3.52 m	NIR	317.2±4.9	7.00±0.70	4.00±0.70
	LWIR	314.9±4.1	6.99±0.56	3.92±0.56
B	3D	335.2±2.0	7.30±0.69	2.64±0.83
Anti-tank gun	DEM	335.4±3.1	7.45±0.44	2.72±0.44
7.42×2.78 m	NIR	336.0±4.3	7.65±0.61	2.88±0.61
	LWIR	333.4±3.5	7.60±0.48	2.92±0.48
C	3D	306.3±2.0	7.64±0.80	2.38±0.96
Truck	DEM	305.6±6.5	7.97±0.92	2.58±0.92
8.40×2.50 m	NIR	Fail	Fail	Fail
	LWIR	307.0±3.9	7.95±0.54	2.57±0.54

Table 3: The queries representing the four prototype after attribute fusion. The fields type, confidence value, height, speed and temperature are omitted.

Field	Target A	Target B	Target C	Target C
Sensor	All	All	3D	DEM/LWIR
Class	MBT	MBT	MBT	MBT
Position	(562.6, 399.3)±0.3	(619.8, 458.8)±0.3	(591.1, 437.1)±∞	(591.4, 437.3)±0.5
Orientation (deg.)	314.8–318.4	333.2–336.9	304.3–308.3	301.1–310.9
Length (m)	6.43–7.16	7.12–7.89	6.84–8.44	7.41–8.49
Width (m)	3.36–4.28	2.44–3.16	1.42–3.34	2.03–3.11

Table 4: The four hypothesis, input values, and resulting cv are shown. The fields class, position, length, width, height, speed and temperature are omitted.

Sensor	3D	LWIR	3D	LWIR
Type	T72	T72	Leclerc	Leclerc
Orientation (deg.)	314.8–318.4	314.8–318.4	314.8–318.4	314.8–318.4
CV	0.92	0.96	0.59	0.65

Table 5: The queries representing the two hypotheses after model match fusion. The fields height, speed and temperature are omitted.

Field	Hypothesis 1	Hypothesis 2
Sensor	LWIR	LWIR
Class	MBT	MBT
Type	T72	Leclerc
CV	0.96	0.65
Position	(562.6, 399.3)±0.3	(562.6, 399.3)±0.3
Orientation (deg.)	314.8–318.4	314.8–318.4
Length (m)	6.43–7.16	6.43–7.16
Width (m)	3.36–4.28	3.36–4.28

6 Discussion

This paper has described a fusion framework in a information system that is applied to coarse-to-fine target recognition. An implementation of the system was described. The fusion framework utilises the independence of each matching result on the origin of the data sent to matching. Also, it avoids possible unwanted correlations in results from different matchings, by not calculating e.g. mean values. Nevertheless, the fusion methods are rudimentary. Several issues remain more ore less unadressed, and call for future work. Such issues are discussed in subsequent sections.

6.1 Incompleteness

When a model matching algorithm produces a cv, it locally optimises and refines the attributes estimating the target's state. If the hypothesis is incomplete, so that one or more of these attributes have not been estimated, the matching algorithm has to estimate it from scratch. For example, if the matching algorithm receives no estimate on the target's orientation, it needs to estimate it. This means that two matching algorithms that receive the same incomplete hypothesis risk to estimate the lacking attribute in different ways. For the orientation on the ground, for instance, one would guess that mainly two different orientations would be represented (see section 4.1).

Thus, incompleteness of a hypothesis could mean that two refined sets of attribute values, originating from that hypothesis, are incompatible. Treating the states as equivalent in the match fusion would then be wrong. Filtering of match results could unvoluntarily exclude wanted results.

In the first fusion step, the fusing of coarse attribute estimates, the method combines values from different sets into the prototype. This makes the risk of an incomplete prototype smaller. On the other hand, there is a risk that estimates from different sets turn out to be incompatible, when combined into the prototype. Note that the term 'incompatible' is here used for *different* attributes. It means that if no target can comply with a certain combination of attribute values, that combination of attribute values is incompatible. To avoid the risk of an incompatible prototype, the prototype could alternatively be constructed by choosing one of the sets in the group (according to some strategy). The high risk of incompleteness is however then brought back.

If a high degree of incompleteness in hypotheses is detected before matching, the matching could possibly be performed as a preliminary matching. The result from this matching could then be used both as a preliminary result sent to the user, and as a basis for better hypotheses. The matching would then be rerun, eventually governing the final result.

6.2 The cv:s

The generatings of cv's are invisible to the fusion methods. It is assumed that the values are calibrated so that two equivalently good matches between target data and model, from two different combinations of data and algorithms, yield the same cv. In the implemented system this has however shown to be a non-trivial issue. Should a certain algorithm systematically produce higher cv's than all other algorithms, it might be overrepresented in the total result. This should be avoided by a fusion method that is robust against bad calibration. Treating the combinations of data and algorithms as individual classifiers, each with a "vote", seems close at hand. This is however complicated by the unknown correlations between different combinations of data and algorithms, see section 4.2.1.

A definition of how cv should be interpreted is also hard to give. If the output is a *list* of match results, the distribution of cv's is assumed to supply the user with information enough. If however only one match result is in the output, the user is forced to interpret a single value. There are two obvious possible interpretations, evidential and probabilistic, respectively. Only knowing that the cv is a 'degree of fit', it seems as if an evidential interpretation is the most suitable for *high* cv's. This would read out something like 'heavy support'. A probalistic interpretation would read out 'likely'. This would be a bit misleading, since more than one model might obtain a high cv, especially if target models are similar. For *low* cv's, however, a probabilistic interpretation — 'unlikely' — seems natural. The evidential interpretation would now be the misleading one. It would read out something like 'some support', since evidence is always supportive. However, a categorisation of obtained cv's into 'high' and 'low' is generally not straight-forward, which further complicates the task.

7 Conclusions

The two-step nature of the target recognition process opens the possibility to an attribute fusion step imbedded in the process. Thanks to the quantitative matching, the attribute fusion step can here be added without risk of data incest. The purpose of this attribute data fusion step is to capture different, distinct initial interpretations of the data, and make it possible to inference over these.

Further work needs to be done on the problem of incompleteness. Rules for handling situations when prototypes are incomplete needs to be tested with respect to computational feasibility etc.

The second fusion step is a sorting which should be made according to user preferences. Ideally, the user ends up with a list of cv's that reveals the truth about the target. However, more work is needed to ensure that the result is a valuable desicion support also in the extreme situations - when the resulting list of cv's is very large, or contains only one item.

Evaluation of the system has been held back by lack of data. In the future, simulated data is expected. Evaluation should then preferrably be made with true end users — to investigate whether a list of match results is an appropriate level of decision support or not.

References

[1] Egils Sviestins. A synergetic partnership. *Swedish Journal of Military Technology (Militärteknisk Tidskrift) (in Swedish)*, (1):14–19, 2003.

[2] B-L. Tseng, C-Y. Lin, D. Zhang, and J.R. Smith. Improved text overlay detection in videos using a fusion-based classifier. In *Proc. of the 2003 Int. Conf. on Multimedia and Expo*, pages 473–476, Baltimore, MD, USA, 6–9 July 2003.

[3] G. Powell, D. Marshall, R. Milliken, and K. Markham. Data fusion of flir and ladar in autonomous weapons systems. In *Proc. of the 6th Intl. Conference on Information Fusion; 'Fusion -03'*, pages 350–357, Cairns, Australia, 8–11 July 2003.

[4] A.M. D'Costa and A.M. Sayed. Data versus decision fusion for classification in sensor networks. In *Proc. of the 6th Intl. Conference on Information Fusion; 'Fusion -03'*, pages 350–357, Cairns, Australia, 8–11 July 2003.

[5] Richard J. Hathaway and James C. Bezdek. Fuzzy c-means clustering of incomplete data. *IEEE. Trans. on Systems, Man and Cybernetics — Part B:Cybernetics*, 31:735–744, 2001.

[6] Mats Bengtsson and Johan Schubert. Fusion of incomplete and fragmented data. Technical Report FOI-R–0047–SE, Command and Control Systems Dept, Swedish Defence Research Agency (FOI), Linköping, Sweden, 2001.

[7] Erland Jungert and Christina Grönwall (eds.). From sensors to decision — towards improved situation awareness in a network centric defence. Technical Report FOI-R–1041–SE, Command and Control Systems Dept, Swedish Defence Research Agency (FOI), Linköping, Sweden, 2003.

[8] Jörgen Ahlberg and Tobias Horney. An information system for target recognition. In *Proc. of SPIE Defense and Security Symposium 2004; Multisensor, Multisource Information Fusion: Architectures, Algorithms and Applications*, pages 163–175, Orlando, FL, USA, 12–16 April 2004.

[9] Karin Silvervarg and Erland Jungert. Visual specification of spatial/temporal queries in a sensor data independent information system. In *Proc. of the 10th Int. Conf. on Distributed Multimedia Systems*, pages 263–268, San Francisco, CA, USA, 8–10 September 2004.

[10] Shi-Kuo Chang, Gennaro Costagliola, Erland Jungert, and Francesco Orciuoli. Querying distributed multimedia databases and data sources for sensor data fusion. *IEEE Trans. on Multimedia*, 6(5):687–702, 2004.

[11] Shi-Kuo Chang and Erland Jungert. Query languages for multimedia search. In Michael S. Lew, editor, *Principles of Visual Information Retrieval*, pages 199–217, Springer Verlag, Berlin, Germany, 2001.

[12] Shi-Kuo Chang, Gennaro Costagliola, and Erland Jungert. Multi-sensor information fusion by query refinement. In S-K. Chang, Zen Chen, and Suh-Yen Lee, editors, *Recent Advances in Visual information Systems*, volume LNCS 2314, pages 687–702, Hsin Chu, Taiwan, March 2002. Springer Verlag, Berlin, Germany, 2002.

[13] Tobias Horney, Erland Jungert, and Martin Folkesson. An ontology controlled data fusion process for a query language. In *Proc. of the 6th Intl. Conference on Information Fusion; 'Fusion -03'*, pages 530–537, Cairns, Australia, 8–11 July 2003.

[14] S. Banbury, S. Selcon, M. Endsley, T. Gorton, and K. Tatlock. Being certain about uncertainty: how the representation of system reliability affects pilot decision making. In *Proc. of the Human Factors and Ergonomics Society 42nd annual Meeting*, pages 36–39, Chicago, IL, USA, 5–9 October 1998.

Appendix H

Agent architecture for a query language in NVD-Environment

(in Swedish), FOI-memo 1025, September, 2004.

Horney, T., Jungert, E.

Agentarkitektur för frågespråk i NBF-miljö

Tobias Horney och Erland Jungert
Inst. för data- och informationsfusion
Avd. Ledningssystem
FOI, Linköping

Introduktion

I en nätverksbaserad värld med en stor mängd sammankopplade plattformar och sensorer behövs ett ledningssystem som på ett bra sätt kan ta tillvara all information i nätet för att kunna presentera en konsistent och väl anpassad (”filtrerad”) lägesbild till olika användare, allt från soldaten i fält till högsta militära ledningen. För att stötta användaren i hanteringen av all denna information har ett frågebaserat informationssystem utvecklats. Frågespråket heter SQL och finns beskrivet i [1, 2].

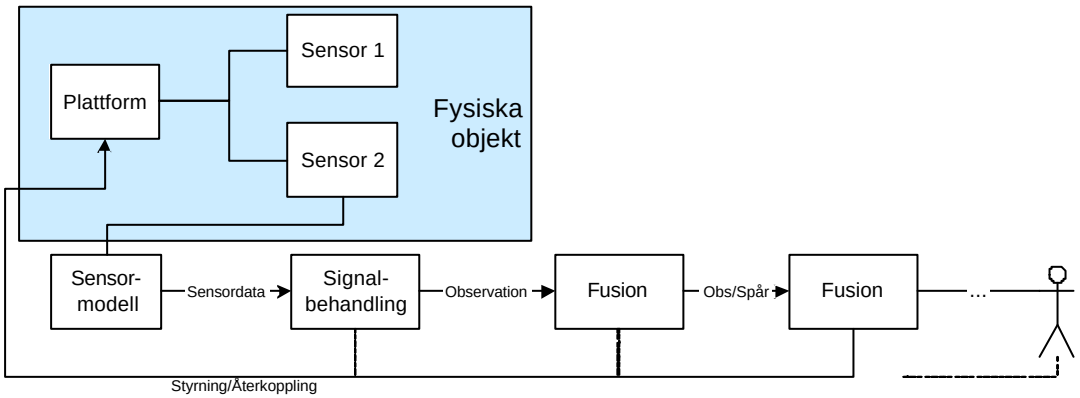
Problemet som behandlas i detta memo är hur frågespråket SQL automatiskt kan hitta lämpliga datakällor på nätet och hur systemet kan få åtkomst till data från dessa källor.

Bakgrund

Uppkopplade på nätet finns diverse plattformar som kan vara allt från en soldat till en UAV, ett markgående spaningsfordon, ett fartyg på havet eller en fast monterad stolpe på vilken man kan montera en eller flera sensorer. Sensorer kan i sin tur vara allt från ögonen eller öronen på en soldat till en radar eller en elektrooptisk sensor i en UAV.

Hela konceptet mappas enkelt från den militära världen till exempelvis civila katastrofsituationer där en konsistent lägesbild av ett område önskas.

I figur 1 ges en schematisk beskrivning av hur informationsflödet från sensor till användare kan se ut.



Figur 1. *Schematisk beskrivning av informationsflöde från sensor till användare.*

Sensorer sitter på plattformar och levererar sensordata. Med sensordata menas i detta sammanhang data från en sensor innan någon måldetektion har genomförts. Exempel på

sensordata är således en IR-bild, en radarplott, en SAR-bild och en punktskur från en laserradar.

Sensordata i sig är inte mycket värda. För att få ut något värde ur sensordata måste man extrahera information ur datamängden, exempelvis genom måldetektion. Med måldetektion menas här att man i sensordata försöker detektera ett eller flera objekt som man är intresserad av att presentera i lägesbilden. Objekten man försöker detektera kan vara exempelvis trupper, fordon eller geografiska formationer. Måldetektion kan göras automatiserat med datorbaserad signalbehandling eller manuellt (den mänskliga hjärnan är duktig på exempelvis detektion av fordon i bilddata). Att kombinera datorns och människans förmågor är naturligtvis också möjligt. Måldetektioner kallas här observationer. En observation är alltså något som observerats i sensordata. En observation har ett eller flera attribut, exempelvis position, hastighet och objekttyp på det som observerats. Det är viktigt att förstå och på ett robust och tillförlitligt sätt hantera alla de osäkerheter som alltid finns i en observation.

Ofta är sensorn som levererar data och måldetektionen hårt kopplade, exempelvis i fallet där sensorn är det mänskliga ögat och måldetektionen görs i den mänskliga hjärnan, eller i fallet där radarplottar bara finns internt i radarsystemet och målspår är det som radarsystemet levererar.

Nu är inte problemet löst ens när man nått så långt att man har observationer på allt man vill visa i sin lägesbild. För att bygga en bra lägesbild utifrån alla observationer krävs en lång rad steg där information förfinas, vägs ihop och utvärderas, dvs fusioneras. Den fusionerade informationen måste sedan presenteras på ett lämpligt sätt för att användaren av det ledningssystem där lägesbilden presenteras ska kunna tillgodogöra sig informationen på ett så bra sätt som möjligt. Om ingen filtrering görs, utan all information som finns tillgänglig trycks ut till användaren, är risken stor att han kommer drabbas av alldeles för mycket information. Det sätt vi använder för att ge användaren tillgång till den lägesbild han behöver i olika situationer är att ge honom möjlighet att kunna ställa frågor till systemet, exempelvis ”Visa alla terränggående fordon som passerat genom område x under tidsintervallet y”.

Problem

Användaren av ett informationssystem ska inte behöva bry sig om vilka plattformar, sensorer och databaser som finns tillgängliga för att besvara hans fråga. Systemet ska istället automatiskt hitta lämpliga datakällor (sensorer, databaser, etc) med den information som behövs för att besvara frågan.

Statusläge

Arbete med hur ett system kan välja lämpliga sensordata och lämpliga algoritmer för att bearbeta dessa sensordata för att besvara en specifik fråga från användaren har gjorts tidigare [3, 4].

I detta memo beskrivs en ansats för en annan del av problemet, nämligen hur systemet automatiskt kan hitta datakällor på nätet och hur systemet kan få åtkomst till data från dessa källor (sensorer etc).

Ansats

För att lösa problemet föreslås en arkitektur baserad på intelligenta agenter, med två typer av agenter. Den första typen är en resursallokeringsagent och den andra en dataåtkomstagent.

Intelligent agent, vad är det?

Agenter är autonoma enheter som kan ses som att de uppfattar sin omgivning genom sensorer och agerar via effektorer. Agenter är i praktiken datorprogram som exekverar i någon beräkningsenhet. Att säga att de är autonoma betyder i någon utsträckning att de har kontroll över sitt beteende och kan agera utan inblandning från människor eller andra system. Agenter uppnår mål eller utför uppgifter för att uppfylla de mål de designats för. Generellt sett kan dessa mål och uppgifter vara kompletterande eller i konflikt med varandra. Att de är intelligenta betyder att de uppfyller sina mål och utför sina uppgifter så att några givna prestandamått optimeras. [5]

Varför använda intelligenta agenter?

Nedan presenteras några argument för att använda intelligenta agenter [5].

Effektivitet

Agenter kan arbeta asynkront och parallellt och detta kan resultera i förbättrad total prestanda (såvida overheaden som uppkommer pga koordination mellan agenterna inte kostar mer än man tjänar).

Robusthet

Om en agent misslyckas betyder inte det nödvändigtvis att hela systemet blir värdelöst, eftersom andra agenter i systemet kan överta dess uppgifter.

Skalbarhet

Systemet kan anpassas till ett större problem genom att addera fler agenter utan att detta nödvändigtvis påverkar de andra agenterna negativt.

Kostnad

Det kan vara mycket mer kostnadseffektivt än ett centraliserat system, eftersom det sätts ihop av små delsystem med låg kostnad per enhet.

Systemutveckling och återanvändbarhet

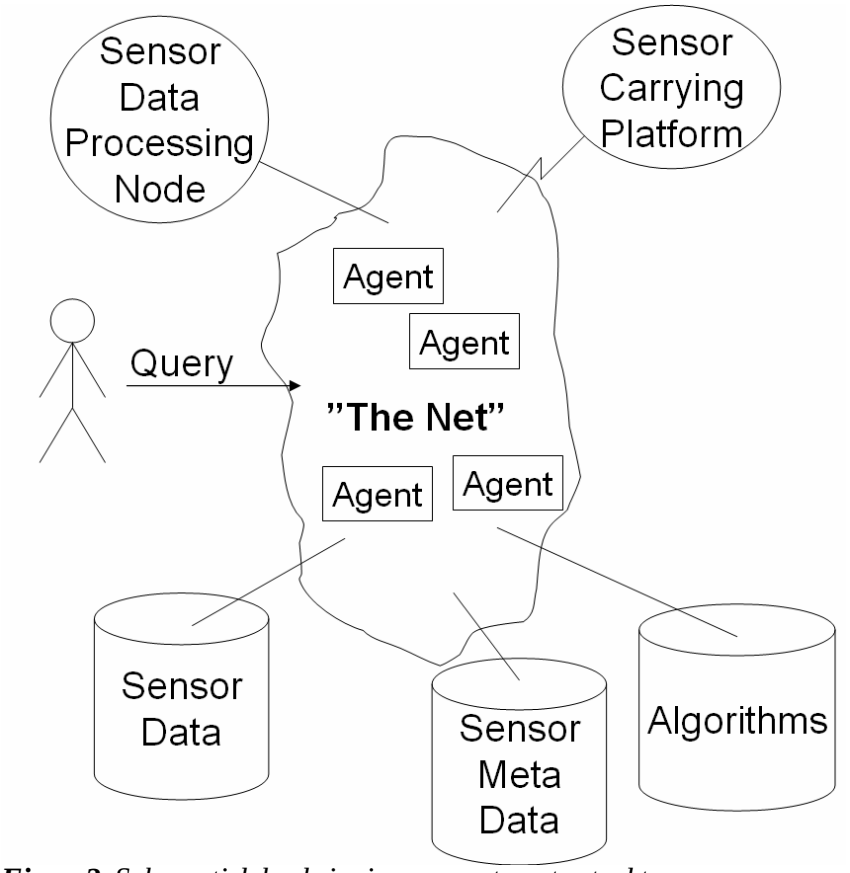
En agentbaserad arkitektur medger att enskilda agenter kan utvecklas separat av olika specialister och att hela systemet kan testas och underhållas lättare. Dessutom kan det vara möjligt att konfigurera om och återanvända agenter i olika applikationer och scenarier.

Agenter

I detta avsnitt beskrivs de agenter som föreslås. Först ges dock en övergripande beskrivning av systemet de är tänkta att verka i.

Systemets funktionalitet och struktur

I figur 2 ges en schematisk beskrivning av strukturen för systemet agenterna är tänkta att verka i.



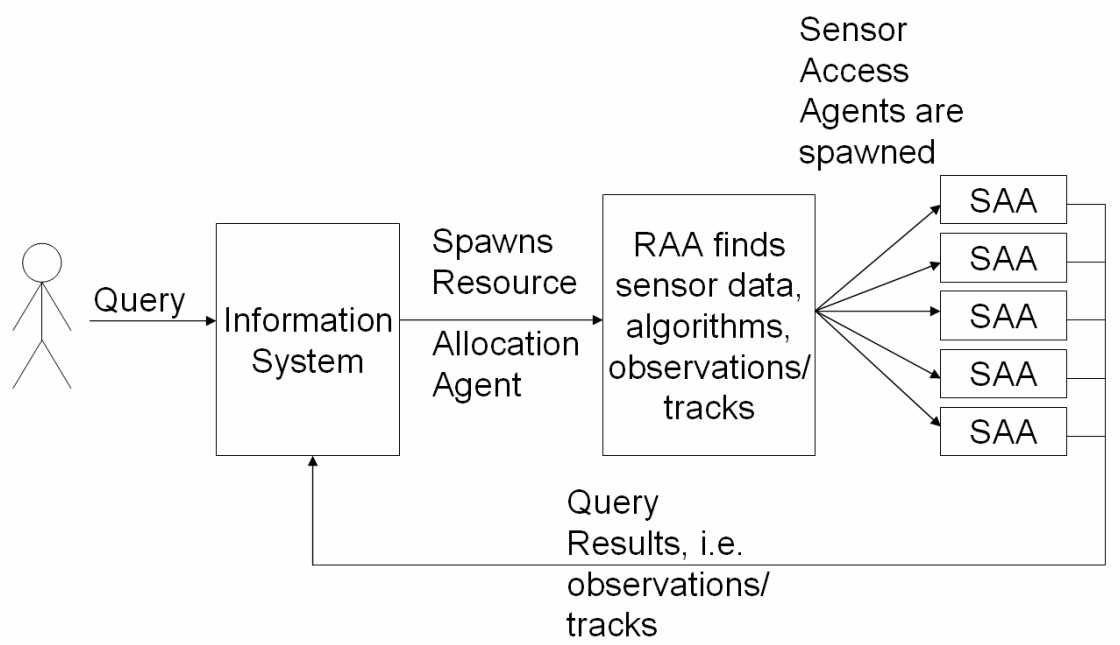
Figur 2. *Schematisk beskrivning av systemets struktur.*

För att få hög robusthet krävs att systemet är distribuerat och att viktig information finns lagrad på flera ställen i nätet (redundans). Eftersom mängden sensordata i ett sådant här system är mycket stor krävs att data behandlas så nära källan som möjligt och att förädlad information kommuniceras mellan de olika noderna i nätet. Naturligtvis kan man tänka sig situationer där man exempelvis vill tanka ner rå sensordata från en flygande plattform, men detta måste vara undantagsfall. Alla delar i systemet bör uppträda så autonomt som möjligt så att kommunikationen kan hållas nere, eftersom bandbredden på olika kommunikationslänkar kommer att vara begränsad, men kanske ännu viktigare i många fall är att utgående kommunikation röjer en plattforms närvaro.

Det gäller alltså att på ett intelligent sätt kommunicera så lite information som möjligt vid så få och lämpliga tidpunkter som möjligt. Intelligenta agenter lämpar sig väl för denna

typ av autonomt uppträdande. Exempelvis kan de förhandla med andra agenter (som har andra mål än agenten själv) om vad som är lämpligt att göra i olika situationer.

En schematisk beskrivning av hur en fråga från en användare leder till att agenter skapas finns i figur 3.



Figur 3. *Schematisk beskrivning av hur en fråga leder till att agenter skapas.*

Resursallokeringsagent

När en användare ställer en fråga skapas en resursallokeringsagent. Denna agent är sedan ansvarig för att leta upp och hålla reda på de resurser som behövs för att besvara frågan över tiden. Resurser som agenten letar upp och håller reda på är sensordata, algoritmer för behandling av sensordata (måldetektionsalgoritmer etc), observationer och målspår som kan behövas för att besvara frågan. Dessa resurser finns tillgängliga antingen i databaser eller i sensorplattformar, men är i båda fallen åtkomliga via nätet.

Det är resursallokeringsagentens uppgift att avgöra vilka resurser som är lämpliga för att besvara den givna frågan. En mycket viktig del i detta är att resursallokeringsagenten måste ha tillgång till information om resurserna (s.k. meta-data), bla. information om vilket område sensordata täcker och vid vilken tidpunkt den samlats in. Antingen måste alla sensorer leverera meta-data (täckningsinformation m.m.) hela tiden till en databas med meta-data, eller får agenten fråga alla sensorresurser på nätet vem som har data för det område och det tidsintervall som frågan gäller och sedan från sensorresurserna som har lämpliga data svara. Den förra tekniken kan kallas ”meta-data push” och den senare ”meta-data pull”.

Det är inte bara sensordata resursallokeringsagenten letar efter utan även förädlad information såsom observationer och målspår av efterfrågade objekt. Dessa kan finnas lagrade i databaser eller hämtas online från plattformar.

Resursallokeringsagenten ska dessutom leta upp lämpliga måldetektionsalgoritmer som kan användas på de sensordata som hittats. Vilka måldetektionsalgoritmer som är lämpliga och därför ska letas upp väljs med hjälp av det ontologiska kunskapssystem som tidigare utvecklats [3, 4]. Genom att distribuera ut lämpliga delar av det ontologiska kunskapssystemet till olika noder i nätet kan en fråga exekveras på ett distribuerat sätt efterhand som agenten hittar lämpliga resurser. Algoritmerna kan lagras lokalt i informationssystemet eller någon annanstans på nätet, exempelvis i sensordatabearbetningsnoder, där algoritmerna kan exekveras. Genom att ha många sådana noder kan algoritmerna exekveras på ett distribuerat sätt, gärna så nära datakällan som möjligt. Man kan med fördel ha en sensordatabearbetningsnod i en sensorbärande plattform.

Eftersom resursallokeringsagenten är ansvarig för att hålla koll på lämpliga resurser för att besvara frågan över tiden måste den fortsätta leva och leta resurser så länge frågan är aktiv. En fråga kan mycket väl behandla tidsintervall som sträcker sig in i framtiden, exempelvis ”Visa alla stridsvagnar i område x från en timme bakåt i tidenoch två timmar framåt i tiden”. I en sådan fråga kommer agenten fortsätta leta efter resurser för att besvara frågan i två timmar.

Det är även möjligt att låta agenten försöka få tag i lämpliga resurser genom att exempelvis styra sensorer av lämplig typ till det intressanta området, dvs man kan låta agenten sköta sensorstyrning. Låter man många agenter ”tävla” om de tillgängliga resurserna på ett intelligent sätt kan man använda detta för att hantera resurserna på ett lämpligt sett (s.k. sensorstyrning) för att optimera den globala lägesbilden. Denna möjlighet med resursallokeringsagenten är ännu ej utredd, men stor potential torde finnas. Det arbete som hittills gjort är fokuserat på att låta resursallokeringsagenten hitta lämplig information och hålla redan på den.

Dataåtkomstsagent

När resursallokeringsagenten har allokerat resurser för att besvara frågan skapas en dataåtkomstagent för varje sensordata-, observations- och målspårsresurs.

Om resursen är sensordata kommer dataåtkomstagenten ta med sig information om vilka algoritmer som ska appliceras för att få lämpliga observationer. En lämplig sensordatabearbetningsnod väljs (en bearbetningsnod med lämpliga prestanda och ledig beräkningskraft så nära datakällan som möjligt) av agenten och algoritmerna appliceras på aktuella data. Agenten vet hur den ska komma åt sensordata och hur den ska applicera algoritmerna i bearbetningsnoden. Agenten tar sedan resultatet (observationerna) och levererar dessa till informationssystemet där vidare bearbetning (fusion, resultatpresentation etc) kan ske. Man kan även tänka sig algoritmer som skapar målspår utifrån sekvenser av sensordata (istället för observationer). Om detta sker levererar agenten målspåren till informationsystemet, se mer i stycket om målspår nedan.

Om resursen är en observation levererar agenten denna observation direkt till informationssystemet för vidare bearbetning.

Om resursen är ett målspår levererar agenten målspårsuppdateringar till informationssystemet under den tid som frågan är aktiv. Dataåtkomstagenterna lever alltså kvar så länge frågan och målspåret lever, därefter upphör de att existera.

Slutsats

Intelligenta agenter lämpar sig tillsynes väl för att på ett robust och intelligent sätt kunna hitta och få åtkomst till lämpliga resurser på nätet i det framtida nätverksbaserade försvaret. Med hjälp av agenterna kan frågor från användaren av ett informationssystem exekveras på ett distribuerat sätt. Detta är bra ur både beräknings- och robusthetssynpunkt.

Tillgång till data om sensordata (meta-data), bla. täckningsområden för insamlad data är en nyckelfråga för att resurserna i nätet ska kunna utnyttjas på ett bra sätt.

En agentbaserad arkitektur medger att enskilda agenter kan utvecklas separat av olika specialister och att hela systemet kan testas och underhållas lättare. Dessutom kan det vara möjligt att konfigurera om och återanvända agenter i olika applikationer och scenarier.

Koppling till tjänstekonceptet

Användningen av intelligenta agenter mallar in mycket väl i tjänstekonceptet. Olika typer av tjänster på olika nivåer finns tillgängliga i nätet. Tjänsterna har väldefinierade tjänstegränsytor som används av agenterna för att komma åt tjänsterna.

Referenser

[1] E. Jungert, C. Grönwall et al., “From sensors to Decision – Towards improved situation awareness in a network centric defence”, *FOI-R--1041--SE*, FOI Linköping, december 2003.

[2] S.-K. Chang, G. Costagliola, E. Jungert, F. Orciuoli, “Querying Distributed Multimedia Databases and Data Sources for Sensor Data Fusion”, *accepted for publication in the journal of IEEE transaction on Multimedia*, 2004.

[3] T. Horney, “Design of an ontological knowledge structure for a query language for multiple data sources”, *FOI-R—0498—SE*, FOI Linköping, maj 2002.

[4] T. Horney, E. Jungert, and M. Folkesson, “An Ontology Controlled Data Fusion Process for Query Language,” *Proceedings of the International Conference on Information Fusion*, Cairns, Australien, juli 2003.

[5] G. Weiss (ed), ”Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence”, *The MIT Press*, Cambridge, Massachusetts, London, England, 2000.

Appendix I

Determination of Terrain Features in a Terrain Model from Laser Radar Data

3-D reconstruction from airborne laserscanner and InSAR data, Dresden, Germany, October 8-10, 2003.

Lantz, F., Jungert, E., Sjövall, M

DETERMINATION OF TERRAIN FEATURES IN A TERRAIN MODEL FROM LASER RADAR DATA

Fredrik Lantz, Erland Jungert, Mats Sjövall
Swedish Defence Research Agency (FOI)
Box 1165, S-581 11 Linköping, Sweden
tel +46 13 37800, fax +46 13 378050
{jungert, flantz}@foi.se

COMMISSION III, WG 3

KEY WORDS: Digital terrain model, laser radar data, terrain features, symbolic structures, matching

ABSTRACT:

Determination of terrain data features in a high resolution terrain data model are required in a large variety of applications where visualization is an important aspect. This is a complex problem because a terrain model is generally a mapping of the reality into a continuous 3D surface model of arbitrary shape, requiring an extremely large amount of data especially when a high resolution is required. Data from such a terrain data model need to be efficiently stored, maintained and visualized. Furthermore reduction of the stored data without loss of relevant information is required. In this work a terrain model that allows the determination of various terrain features through symbolic filters is proposed. The filters are compositions of symbolic surface elements. It will be demonstrated how these filters can be used to obtain objects and object features for visualization and other purposes.

1 INTRODUCTION

Development of digital terrain models has for a long time been subject to extensive research efforts, see e.g. (Maguire 1991). The driving force behind these efforts is a large number of applications, e.g. visualization of the terrain in three dimensions, determination of the line of sight or the areas covered by specific sensors. Two main data structures for representation of digital terrain models can be found. The first one is the gridded model, which is favorable because it is regular but may require the storage of too many dense data point if a high resolution is required. At the other extreme we have the TIN model (triangulated irregular network) for which a fairly large number of approaches has been proposed. The advantage of this structure is that it requires a smaller number of data points than the grid structure given the same resolution. The irregular structure requires more complex algorithms both in generation and analysis and it may occasionally result in inefficient storage structures as well. When the purpose of the terrain model is not only to allow visualization or data analysis but also to query data represented at a very high resolution neither of these two structures is useful. The grid structure, because it will require too many data points and the TIN model because of the complexity of the irregular structure. Besides, neither of these two structures are well suited to applications where matching is an important and necessary ingredient. Consequently, another approach must be taken when such operations are required.

The long term goal in this work is to design a query language based on a type of query-by-example technique that should be able to work on three dimensional data ranging from a very high resolution (≤ 0.5 m) and up to a resolution of, say, 30 m. The upper limit is here determined from the observation that objects with a size of a couple of hundred meters in extension correspond to the maximum size of the requested objects in our applications. The data used here for creation of the terrain model is registered by a scanning airborne laser-radar called TopEye (property of TopEye AB, Sweden). The terrain model can be described as a hybrid between a grid structure and a triangle structure and can simply be described as a regular grid structure augmented with a number of significant irregular data points and lines. The atomic parts of this structure are called a tile. Each such elementary structure can be triangulated in isolation. Another advantage is that these tiles can be stored in a database in accordance to a relatively simple structure (Lantz 2000;Elmqvist 2001b) from which data can be efficiently accessed. A further advantage is that the data structure will allow the generation of a high-resolution terrain model with less data points than an ordinary grid structure for the same resolution; this has been demonstrated in (Lantz 2000). The main advantage of this structure is that it can be transformed into a symbolic structure, which efficiently will allow various types of matching processes. Operations that can be supported by such a structure are e.g. detection of changes over time (change detection) in the terrain and detection of specific terrain objects/features defined by the users as well as a combination of the two. This work is primarily concerned with the development of a *symbolic representation* of the terrain model which will be demonstrated for an application where the purpose is to determine the position/existence of user defined objects; this will be illustrated with a number of examples. Apart from being an appropriate structure for querying, the structure can also be used as an index for a height model and should, as such, be seen as a complement to a height model and not as a replacement. The symbolic structure can also be used for determination of drivable terrain which is outside the scope of this work.

Some related work concerned with techniques for querying terrain data for various features and objects can be found in the literature. Although at this time no symbolic query structure has been found. Despite this, some of the given approaches are quite interesting. In (Bradly 1999) Bradley presents a high-resolution terrain database for a gridded representation covering parts of Mars and where data should be stored in a relational database system. Queries can be written in SQL. The main purpose is to explore the planet and for that reason a fairly large number of terrain features have been determined. Clematis et al. (Clematis 1998) describes a technique for parallel fuzzy queries applied to a spatial data model. A 3D-query space for GIS is described by Fritsch and Schmidt (Fritsch 1994). Their system includes digital terrain data described in terms of a conceptual model called a NIAM-diagram and by also using the entity-relationship model these diagrams can be used to describe occurring object relations in a powerful way. Queries can be formulated in standard SQL. The database is not limited to just terrain data, other types of geo-information is integrated as well. The terrain database includes gridded data and a type of hybrid TIN-structure.

2 THE PROBLEM DOMAIN

The main purpose of this work has been to use laser radar data in developing a technique for symbolic description of the terrain and to create a structure with the following characteristics:

- *The symbolic terrain elements used for description of the terrain should correspond to a classification of a tile.*

This problem corresponds to identifying the members of a class of surface structures, where each member belongs to a particular category. These members can be interpreted in a symbolic way and they will subsequently be called symbolic tiles. Furthermore

- *The symbolic tiles should also be used for identification of different terrain objects or features. This process is carried out by defining a set of symbolic filters. Each such filter should correspond to a composition of a set of connected symbolic tiles. To determine the requested objects a suitable and efficient search strategy must be provided as well.*

Examples of objects that should be possible to determine by means of the filters are, e.g. ditches and fields for landing of helicopters. However, the latter type requires also information about various types of land cover classes such as lakes or forests since a helicopter cannot land on areas covered by such object classes. The terrain must also be available in different resolution levels, which however, will not be discussed further in this paper. As already mentioned, the eventual query language must allow queries where the filters, which may be represented in multiple resolutions, will be used for determination of more or less arbitrary three dimensional terrain objects. The method should thus be applicable to the search for objects of all sizes.

3 SPATIAL CATEGORIES

3.1 The tiles

In order to determine the spatial categories of the terrain, the original sensor data is pre-processed to eliminate non-terrain data, e.g. trees and buildings (Elmqvist 2001a). The pre-processing step produces a surface defined on a rectangular grid with 0.5 m between grid points. Formally, given a compact and connected domain $\Omega \subseteq \Re^2$, a finite number of points $P = \{(x_i, y_i)\} \subseteq \Omega$ and a function $f: P \to \Re$, $\psi(P,f) = \{(x_i, y_i, f(x_i,y_i)) \,|\, (x_i,y_i) \in P\}$ is called the sampled surface corresponding to *f*. The purpose of this section can loosely be stated as to find a more suitable surface representation, that can be substituted for $\psi(P, f)$ in subsequent spatial reasoning tasks, e.g. filtering for detection and visualization of certain terrain objects.

The first step in creating such a surface representation is to partition the domain into sub regions and to define a class of sub functions with non-zero only support on those sub regions. Let $\Gamma = \{\gamma_1,\ldots,\gamma_n\}$ be a regular partition of Ω into square regions with sides 2 m and let $F_\Gamma = \{f_1, \ldots, f_n\}$ be any set of functions $f_j : \Re^2 \to \Re$ where

- f_j is continuous
- $\int_{\Re^2} f_j(x, y)\,dx\,dy = 0$
- $f_j(x,y)=0 \;\; \forall (x, y) \notin \gamma_j$

A pair (γ_j, f_j) is called a tile and the set of all tiles over Ω is *T*. A surface model consisting of tiles will be determined by substituting every (sampled) sub surface $\psi_j = \{(x, y, f(x, y)) | (x, y) \in \gamma_j \cap P\}$, *j=1,...,n* by a particular tile, chosen from a carefully selected set. The definition of that set will be presented in the following sections.

3.2 Overview of the process

The next step is to define a number of categories of tiles, i.e. the spatial categories. A spatial category is a set of tiles with similar features, e.g. with similar inclination or with edges at similar locations. The definition of the categories is carried out in three steps. Firstly, a set of representative tiles $REP \subseteq T$ is determined. Secondly, a suitable distance metric *dist:* $T \times T \to \Re_+$ for comparing the similarity of the tiles is needed. Finally, a category [r], $r \in REP$ can be defined, using the metric *dist*, as the set of all tiles that are more similar to *r* than to any other $k \in REP$. In practise, the categories are never calculated, only their representatives. After definition of the categories, every representative (and thus every category) is given a partially symbolic interpretation by a unique string, called a symbolic tile. Formally, this is done by a Symbolic Interpretation mapping *SI:* $REP \to M_\Gamma \times A \times B \times A \times A \times \Re_+$, where *A={0,...,8}*, *B={0,1,2}* and $M_\Gamma = \{(u_j,v_j)\}$, is the set of centre points of γ_i, *j=1,...,n*. A sub surface ψ_j is compared with every representative on $\gamma_j \cap P$ by first subtracting the mean of *f* over γ_i and then using a discrete counterpart of the above distance metric. The best match r_k is selected and ψ_j is substituted by the string *s* such that $SI(r_k)=s$.

3.3 Defining the representatives

The complete, formal definition of the representatives is beyond the scope of this work. Instead, an informal but more easily accessible presentation will be given. Three different representations are necessary to achieve a sufficiently accurate surface model. The types respectively describe sub-surfaces using:

1. Two linear functions with non-zero support on separate parts of the square, but where their composition is continuous.
2. One single linear function.
3. A function with a single extreme point in a specified location.

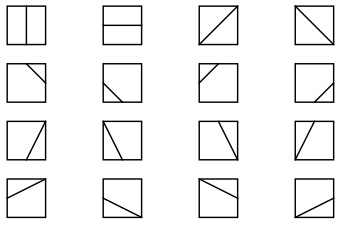


Figure 1. The 16 allowed partitions of a square region.

To describe the first type, consider the 16 partitions of any square γ_j into two parts shown in figure 1. Every representative of this type is completely determined by one of these defining partitions, together with a certain combination of linear functions g_j and h_j. In this work, there are *nine* such combinations for each partition. The functions g_j and h_j are chosen such that their composition is continuous and the sub surface associated with a part has a direction of the inclination (in the xy-plane) among those shown in figure 2. Only three directions are allowed for a partition, zero inclination and the two opposite

inclinations that are perpendicular to the direction of the partition edge. The allowed directions for two partitions are shown in figure 3. The representatives can be quite similar, only differing in the location of the edge that partitions the square. In fact, all representatives of this type can be seen as variations of the basic category forms, shown in figure 4, rotated and translated to "fit" the partition. The second type of representative can be seen as a sub type of the first. This can be seen by considering the case where the two functions have exactly the same partial derivatives, e.g. as for the top left tile in figure 3. Note that some representatives are given by more than one partition, e.g. the flat square is given by every partition.

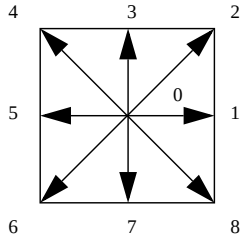


Figure 2. The allowed values for the partial derivatives for a sub function, displayed as vectors seen from the centre of the square. Also shown are the names of the points defining an inclination and a partition (at the head of the arrow).

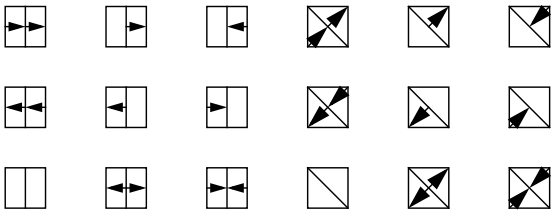


Figure 3. The allowed representatives for two partitions. An arrow indicates the inclination direction for that part. No arrow means the part is flat.

A common terrain feature that is poorly approximated by the above defined representatives is extreme points at other locations than the corners. Therefore ten additional representatives, five minima and five maxima, are allowed with extreme points at the middle of the four borders of the square and at the middle point of the entire square. All together this makes 115 different categories; 96 of type 1, 9 of type 2 and 10 of type 3.

3.4 The categories

A general distance metric for comparing similarity between tiles $(\gamma_j, k_j), (\gamma_l, k_l) \in T$ is:

$$dist((\gamma_j, k_j), (\gamma_l, k_l)) = \left\| (k_j / \|k_j\|) - (k_l / \|k_l\|) \right\| .$$

This metric will determine different categories depending on which norm that is used. For the examples in the following sections, the norm used is the integral over absolute value.

At last, the category determined by $r \in REP$ can be defined as $[r] = \{ k \mid k \in T \wedge ((\forall (r_l \in REP))(dist(k, r) \leq dist(k, r_l))) \}$. As mentioned above, there is no need to calculate the categories, but only the representatives. The calculation of which category a certain ψ_j belong to can be carried out straightforwardly by computing the distance to every representative tile. One important exception to this rule is the representative with partial derivatives all zero, i.e. the totally flat representative. To compensate for the fact that tiles with uninteresting, small height differences will be given undue consideration, all sub surfaces with norm below some pre-determined threshold will be considered a member of the flat category.

3.5 A symbolic interpretation

The symbolic interpretation SI is a composition of five sub mappings, *Position*: $REP \rightarrow M_\Gamma$, *Inclination:* $REP \rightarrow A$, *Featurestate:* $REP \rightarrow B$, *Featureorientation*: $REP \rightarrow A \times A$ and *Norm:* $REP \rightarrow \Re_+$. Position is defined as *Position*(γ_i , f_i)= (u_i,v_i) and $Norm(\gamma_i, f_i) = \|f_i\|$. Featureorientation is an encoding of the partition of the tile in the case the tile is of type 1. A particular partition is identified by the pair (*a*,*b*) of points on the border of the square that it divides. The points are named as shown in figure 3. For a tile of type 3, Featureorientation map the tile to (*a*,*a*) if *a* is the location of the extreme point. For type 2, the encoding is irrelevant. Inclination is an encoding of the average inclination direction of the tile. As in the case of Featureorientation, the set *A* is used to identify an inclination direction. Naturally, if two sub regions are inclined in opposite directions the average inclination is zero. Featurestate encode a deviation from Inclination and is used to differentiate between tiles with the same average inclination. The deviation can be positive, 1, negative, 2 or zero, 0. In figure 4 the top row corresponds to featurestate(r) = 1 and the bottom row to 2.

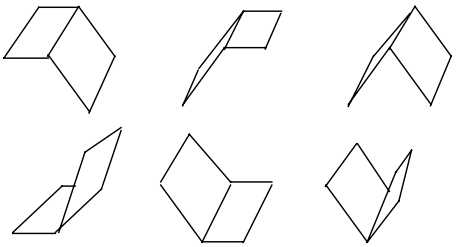


Figure 4. Basic category forms.

4 FILTER STRUCTURES

4.1 Finding segments

The filter matching technique is linear and the filters correspond to a sequence of connected symbolic tiles describing the feature that will be subject to the search. The search is carried out by applying the filters across the given area of interest (AOI), i.e. the given terrain model, to find the requested terrain features. The approach taken here is to use the specific and characteristic cross-sections, which exist for all terrain formations. For instance, a hill has always slopes going either upwards or downwards depending on the position of the viewer. Thus any cross-section of a hill can first be described with an upward directed slope followed by a slope directed downwards. A ditch can be described with an opposite structure. At a conceptual level the cross-section of a terrain formation consist of a start segment, a possible middle segment and an end segment, see figure 5.

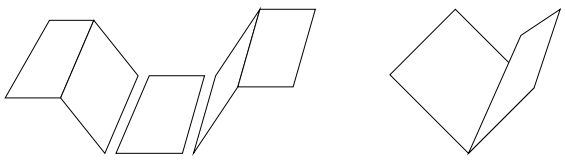


Figure 5. The filter structure to be applied to the cross-section of a ditch

The filter can be of different size depending on the type and size of the requested terrain feature. Consequently, the search strategy must be sufficiently flexible to accommodate these demands. It turns out that a two level search strategy is the most appropriate. The simplest high level search, i.e. tile indexing can be based on row-ordering (Worboys 1995). This type of search algorithm explores the tile structure row by row, from top to bot-

tom, and column by column, from left to right, see figure 6.

Exhaustive search (a) Simplified search (b)

Figure 6. All possible search directions of a 4 x 4 filter (a) and the orthogonal search directions of a simplified search (b).

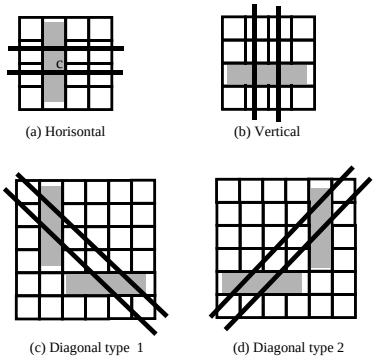


Figure 7. Approximate orientation of long thin objects (bold lines at the edge of the object). Filters applied in two directions (grey boxes) will find objects in all orientations.

By applying the simplified search in horizontal and vertical directions the two filters will find the objects corresponding to all possible orientations. Intuitively it seems like two more filters may be necessary to find the diagonal cases but as can be seen in figure 7, this is not the case. Some minor problems may arise when the simplified search is used. The filters cannot be given a width equal to the maximum width of the terrain formation if the terrain structure has an orientation that is diagonal. The width of the filter must in such a case be somewhat larger than the actual terrain formation. An approximation that is certain to find the formations in these cases is the sum of the width of the horizontal and the vertical filters, i.e. a choice that resembles the triangle inequality. Another obvious consideration in the search is that the filters should not be applied such that any part goes outside the edge of the AOI.

As a result of the first matching step a large number of segments will be found. To find out which of these segments that are part of the feature that is searched for adjacent segments must first be connected. To complete this process, two different approaches have been developed.

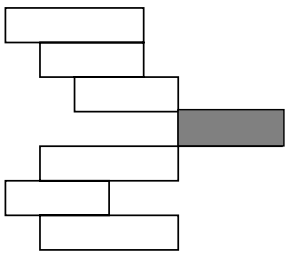


Figure 8. Illustration of the vertical segment search.

4.2 Connecting segments

To connect adjacent segments into objects a search is applied from top to bottom and from left to right. Segments found in step 1 (section 4.1) are connected only if the segments horizontally or vertically overlap each other as can be seen in figure 8 where the white segments will be connected into two different object. The position of the shaded segment prohibits further connections.

4.3 Edges and inclination connections

The algorithm described above is not concerned with the underlying symbolic structure; it operates only on the tiles filtered out in stage 1. A more powerful technique is to take the details of the shape of the filtered tiles into account. To do this we first have to find adjacent tiles which are chosen with respect to some given constraints in the search process. To find tiles that we consider to belong to the same feature, we must also determine whether different existing edges and inclinations proceed into the neighboring tiles. In the case of a tile with both an inclination and an edge the edge takes precedence over the inclination. In each step of the search process a check is made to make sure that the current edge matches the previous edge. To expect an exact match of the edges is in most cases unrealistic so a tolerance of +/- 45 degrees will be allowed. If the check is successful the search direction will be strictly determined by the direction of the edge. Only one search direction will be chosen unless the edge divides the tile asymmetrically, in which case there are two search directions. The chosen directions are depending on the angle of the edge. A case where the tile contains an edge is illustrated in figure 9 by a category with an edge running from point “5” to point “8” as specified in figure 1. Some special cases exist here which are discussed further in (Sjövall 2002).

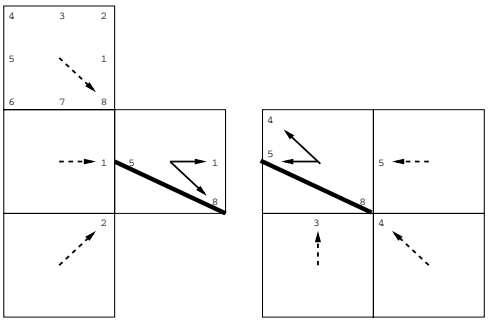


Figure 9. Illustration of the search method in the edge connection algorithm.

5 FILTER SPECIFICATION

The filters are specified in a plain text file that is parsed by a Java-program and applied to the symbolic tile data file. Each segment will first be searched for horizontally then all categories in the filter-specification are rotated 90 degrees and the search continues vertically. All filter lengths and widths are specified in terms of multiples of symbolic tiles. A filter segment as defined in section 4 starts with a *beginsegment* statement and ends with an *endsegment*. Everything between these statements represents the string that is to be searched for. An optional parameter for the segment is invert that will invert the whole segment, i.e. a positive state will become a negative state etc. This can, for instance, be used to quickly transform a ditch-filter into a ridge-filter and vice-versa. The fact that all segments need to be applied both vertically and horizontally requires special attention. Thus the statements *horizontal* and *vertical* specifies the actual search directions.

A segment may contain several sub segments surrounded by *beginsubsegment* and *endsubsegment*. The *beginsubsegment* statement takes three parameters. The first two are mandatory: *minimum length* and *maximum length*, which is the number of consecutive symbolic tiles that should used in this sub-segment. The third is the optional keyword *exclude* which means that this sub segment should not be part of the final segment although it still has to match. The sub segments may also contain the statement *maxerrors* that has a parameter that is the allowed maxi-

mum number of not matched tiles. If this is not specified, no errors are allowed. The sub segments contain a set of allowed symbolic tiles which are specified with the statements *begincategory* and *endcategory*. A symbolic tile is specified by the statements *inclination*, *state* and *orientation*. The keyword not is used to tell which values that are not be permitted. The keyword any will allow all possible values.

After the segment-specifications, the post-processing algorithms can be specified. The statement *connect* will connect tiles using the algorithm specified as the first parameter. Allowed algorithms are *edge* and *segment*. Parameters two to five is minimum/maximum length and width of the final object. These parameters are required but by using a value of -1 the checks can be disabled. If the segment algorithm is used a maximum error can be set as an additional parameter. The edge algorithm is implemented as a recursive algorithm which could lead to problems if the recursion depth is too great. Although this should not happen unless a filter is specified for very large objects, in which case the resolution of the tiles should be chosen differently. The segment algorithm first searches from top to bottom followed by a search from left to right. This means that the algorithm only handles objects that are laid out in a straight line. Objects that changes direction and are curved will not be found. The statement *h-and-v* uses the logical-AND algorithm to combine horizontal and vertical segments. This statement does not require any parameters.The statement *findrectangle* will find rectangles of a specified size. The first parameter is the *width*, the second the *height* and the third set the allowed relative error. An example of a filter with just a single sub segment is

```
beginsegment              /*Start the segment*/
  invert                  /*Invert the segment, i.e.
                            the real inclinations to be
                            searched for is 2, 3 and 4.*/
  beginsubsegment 1 3
                          /*Start a subsegment with a
                            minimum length of 1 and a
                            maximum length of 3.*/
    begincategory         /*Start category*/
        inclinations 6,7 and 8
                          /* Allow inclina-
                            tions 6,7 and 8*/
        state any         /*Allow any state*/
        orientation not 14 15 16
                          /*Allow all orientations
                            except 14, 15 and 16.*/
    endcategory
  endsubsegment
endsegment
connect edge 10 -1 -1 -1
                          /*Connect categories using
                            the edge-algorithm and a
                            minimum length of 10*/
```

The complete segment will have a possible length of 1 to 3 categories (the minimum and maximum length of the subsegment.

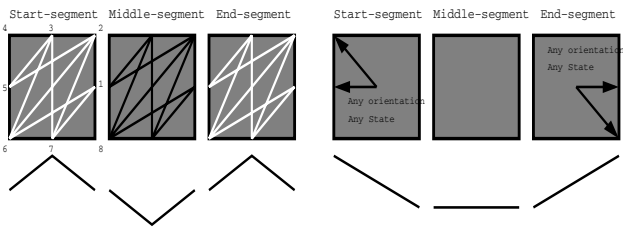


Figure 10. Categories allowed in the ditch-segment.

6 SOME ILLUSTRATIONS

A number of filter types for different features have been developed and tested. Among them can filters for determination of ditches, ridges, hill tops, ponds and flat (sub)- areas including roads be mentioned. We will here give some illustrations of the application of these filters including some of the short-comings.

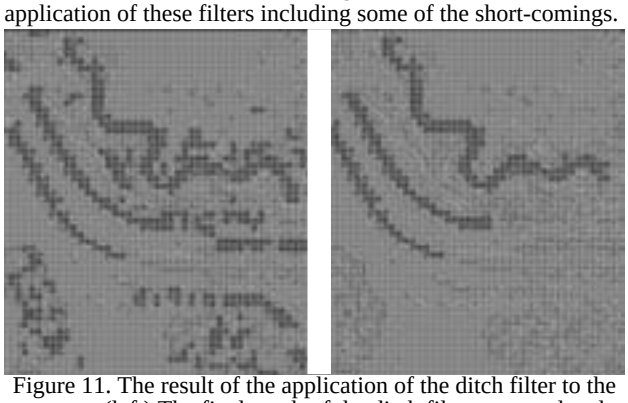


Figure 11. The result of the application of the ditch filter to the test area(left).The final result of the ditch filter as completed with the edge-connect algorithm (right).

6.1 Ditches

This filter finds narrow ditches and is also very tolerant to changes in direction of the ditch. The specification contains three different segment-types:

- The first is a single flat tile with a negative edge (i.e. all orientations except extreme points), see figure 5 at right.
- The second is a segment with a down-slope, flat and up-slope structure. This part finds segments that are part of a ditch that has a northeast-southwest orientation. Figure 10 shows all possible tiles for each sub segment and since the north-south (*3 to 7*) edge as well as the inclinations *1* and *5* are part of the start and end sub-segment, segments with a north-south direction will also be found, see figure 5 at left.
- The third segment is similar to the second, only the difference in the orientation, which is northwest-southeast.

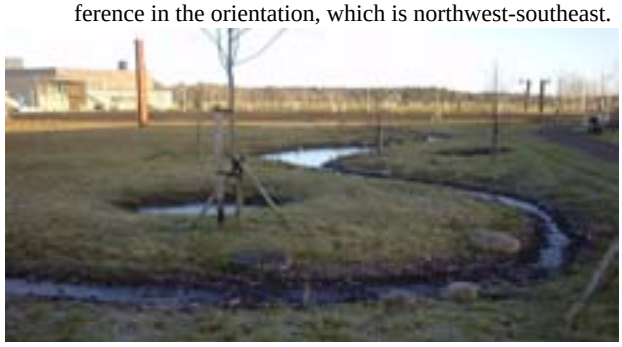


Figure 12. The small stream in real-life.

These three segment types are sufficient to find ditches oriented in all directions since the filters are also rotated 90 degrees. The minimum width of a ditch-segment is one tile and the maximum width is three tiles. Figure 11 shows to the left the result of the segment filter where a lot of false hit segments occur as well.

The segments are then connected using the connect-edge algorithm with a minimum allowed length of 10 (i.e. 10x2 meters). This will remove most of the smaller object hits that may not correspond to ditches. The results can be seen at right in figure 11 where three ditches can be seen. Two of these are in reality ditches along a road and the third a small stream, see figure 12.

Figure 13. The result of the road-filter

6.2 Roads

The principle of the road filter is to find segments that start and end with a tile that is not flat and with several flat tiles in between. The enclosing categories should not be part of the segment; this is done by applying the *exclude* parameter in the *beginsubsegment* statement to the non-flat categories. The only segments that will remain are those including just flat tiles. These segments become connected by using the connect-segment algorithm. A minimum length of 10 is used with an allowed error frequency of 20%. The filter is applied to the complete data set and the result is shown in figure 13. Several shorter and isolated road segments can be seen in several places and is most likely flat areas where houses were removed by the ground-segmentation. However, this should not be considered a serious problem since it is possible to mark these areas in the removal process. Furthermore, the connect segment algorithm does not handle changes in the direction of the elevation well, thus some parts of the roads are missing here and there.

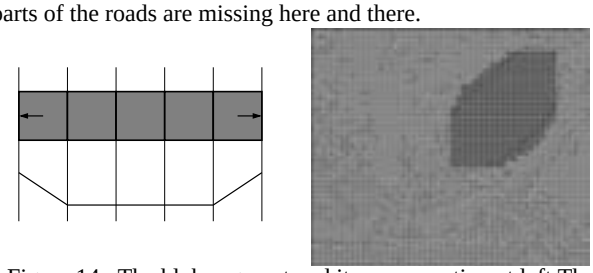


Figure 14. The blob segment and its cross-section at left.The resulting pond using the blob-filter to the right.

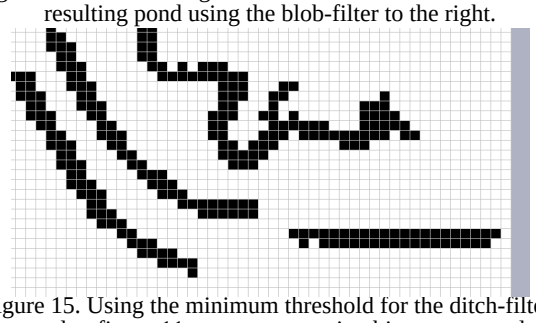


Figure 15. Using the minimum threshold for the ditch-filter. Compared to figure 11, one more terrain object appear at lower right.

6.3 Blobs

This filter will find a smaller blob-like object, e.g. a pond. This is done by looking at the cross-section which ideally will look like the illustration in figure 14. Since a long flat segment is bound to have errors in it, a certain error tolerance is used. To make sure the object is “closed” the cross-section has to be found both horizontally and vertically by using the post-processing filter h-and-v. The edge-connection algorithm is then applied to allow a minimum length and width of 15 tiles. The final result can be seen in figure 14.

6.4 Objects with different inclinations

The above described technique of finding terrain objects identified by their cross sections can not distinguish between tiles with different magnitude of inclination. Nevertheless, the symbolic tiles carry all necessary information to do so in the norm of a tile. In the current version, the previously mentioned filters can be added by a minimum threshold value used to set all tiles with a norm below the threshold as flat. The threshold can be changed dynamically as specified by a user request. The consequence of using a different threshold when searching for ditches can be seen in figure 15, where a curb is detected. This represents a significant improvement over the results in. (Jungert 2002) where cross sections with as small height differences as, in this case, 1.5 dm could not be detected.

7 Conclusions and future work

In this work, techniques that can be used for identification of different types of terrain object features have been described. The technique can be split into two different steps where the first step concerns development of a method for the description of the terrain by means of symbolic tiles. The second step includes methods for determination and connection of non-attached segments.

Future work must include improvement of the filters and further types of terrain-objects must be possible to identify as well. By using the formal filter-specification it is possible to more adequately and correctly describe the terrain-objects which will be the foundation for further work on the development of a query-language.

References

D. J. Maguire, M. F. Goodchild and D. W. Rind (Eds.), 1991, Geographical Information Systems, Longman Scientific and Technical, London, vol. 1, pp. 269-297.

F. Lantz, E. Jungert, 2000, Dual aspects of a Multi-Resolution Grid-Based Terrain Data Model with Supplementary Irregular Data Points, In: *Proceedings of the 3rd International Conference on Information Fusion*, Paris, France, July 10-13.

J. Bradley, 1999, An Efficient Modularized Database Structure for a High-resolution Column-gridded Mars Global Terrain Database, *Journal of Software - Practice and Experience*, vol. 29, no 5, pp. 437-456.

A. Clematis, A. Coda, M. Spagnuolo, S. Spinollo, T. Sloan, 1998, Parallelising Fuzzy Queries for Spatial Data Modelling on a Cray T3D, In: *Proc. of the 4th Int. Workshop on Applied Parallel Computing, Large Scale Scientific and Industrial Problems (PARA’98)*, pp. 76-81.

D. Fritsch, D. Schmidt, 1994, DTM integration and three-dimensional query spaces in geographic information systems, In: *Proceedings of the SPIE - The International Society for Optical Engineering vol. 2357, pt 1, pp.* 235-242.

M. Elmqvist, 2001a, Ground Estimation of Laser Radar Data using Active Shape Models. In: *OEEPE workshop on Airborne Laser-scanning and Interferometric SAR for Detailed Digital Elevation Models*, Stockholm, Sweden.

M. Worboys, 1995, GIS A Computing Perspective, Taylor and Francis, London.

M. Elmqvist, E. Jungert, F. Lantz, Å. Persson, U. Söderman, 2001b, Terrain Modelling and Analysis Using Laser Scanner Data, In: *Proceedings of the Workshop on Land Surface Mapping and Characterization Using Laser Altimetry*, Annapolis, Maryland, October 22-24.

M. Sjövall, 2002, Object and feature recognition in a Digital Terrain Modell, Master thesis, University of Linköping, Sweden, LITH-IDA-Ex-02/16.

E. Jungert, F. Lantz, M. Sjövall,2002, Determination of Terrain Features in a Terrain Data Model Using Symbolic Filters, In: *Proc of the 8th int conf on Distributed Multimedia Systems*, San Francisco, CA, Sept 26-28, pp. 664-667.

Appendix J

Context Fusion for Driveability Analysis

Information Fusion, Philadelphia, PA, July 25-29, 2005.

Lantz, F., S. Edlund, Jungert, E.

Context Fusion for Driveability Analysis

Fredrik Lantz, Susanne Edlund, Erland Jungert
FOI (Swedish Defence Research Agency)
Box 1165, S-%81 11 Linköping, Sweden
{flantz, jungert}@foi.se

Abstract - *Driveability analysis is a quite complex problem that for its solution depends on several factors. One of these factors concerns the type of vehicle for which a drive-way should be determined. Besides this, the terrain structure, the type of vegetation but also the ground type and its conditions play important roles. Driveability analysis will consequently include analysis of primarily geographical information and the outcome of this analysis can be used to support decision making in command and control systems. However, quite often the required geographical information is represented in a resolution that is either too low and/or is represented with a high degree of uncertainty that cannot be neglected. In this work, an approach to driveability analysis is presented in which geographical information is regarded as context information that eventually is fused to generate paths, that may be drivable for certain types of vehicles. This information is fused by means of a knowledge-based technique that determines the driveability from a set of qualitative driveability impact factors.*

1 Introduction

Driveability analysis (also called trafficability analysis) of terrain and geographical data offers an important technique for decision support for all kinds of movements in the terrain. This type of analysis is needed for the judgement of possible movements of targets (target tracking) and for the planning of future movements. An important data source for such an analysis is a digital terrain model [1], [2], which in this case is a high resolution digital terrain model generated from laser radar. In [2] is a method for finding essential terrain objects such as ditches and ridges described. The geographical data are generally represented as map data that in order to become useful need to be in a high resolution. Parts of this work has earlier been described in [3].

The research on driveability has been subject to fairly intense studies e.g. Donlon and Forbus [4] have developed a domain theory for trafficability to partition regions according to some criterion. They discuss both complex factor overlay and combined obstacle overlay as a means to do this. A complex factor overlay partitions regions according to the type of terrain and combine them into areas of homogenous characteristics. A combined obstacle overlay characterizes the terrain according to its effect on the vehicular movement. Bonasso [5] presents a trafficability theory using first-order predicate calculus to enable reasoning about trafficable paths without the need for a detailed terrain model where only a limited set of qualitative values are used, which e.g. allows a desert to have the value *SOFT* given to a rigidity attribute, and *FREE* to a forest density attribute. Much existing work on trafficability has focused on real-time applications, mainly path planning for unmanned ground vehicles (*UGV*s), which move in unknown areas [6], [7], [8], [9], [10] and [11]. In the work by Johnson et. al. [12] and Kruse et. al. [13] hyperspectral or multispectral data are used to classify areas according to their surface composition, but with limited success. Kruse et. al. improve the usefulness of their approach by using data fusion to identify areas which e.g. have both high clay content and high slopes. Another fusion oriented approach is given by Glinton et al. [16]. Sapounas et. al. [15] use an object-oriented approach and cost functions to calculate bounding regions and shortest paths. The cost functions represent the effect of the terrain on the travel speed. Finally, Slocum et al. presents an approach based on a trafficability engine [14].

The rest of this paper is organized as follows. In section 2 context fusion is defined. This section includes also a definition of the problem addressed in this paper. A discussion of existing driveability impact factors is discussed in section 3. The digital terrain model used is described in section 4. Driveability analysis, basically including the used measures of driveability and the final cost function with its impact factors is discussed in section 5. Finally some results are presented in section 6 followed by the conclusions and some further research topics in section 7.

2 Context Fusion

The main objective of this work is to develop a method for driveability analysis that determines

whether a certain path through a terrain area is possible to travel given a certain type of vehicle. The result of this analysis should be visualized on a map where the driveability is highlighted. To be useful in real-world applications information from a large number of sources is required. The focus in this work, however, is concerned with the overall method and basic structure of driveability. Still, in addition to the digital high resolution data model other geographical information is used as well, e.g. forest data including the density of the forest. Finally, information about the vehicle is used, both with respect to its properties (e.g. if it has wheels or tracks, its weight etc.), and its capabilities (e.g. how steep slopes it can climb). This problem turns out to be complex where the various types of information must be fused and presented to enable generation of complete driveability maps. Here context information refers to the geographical background information as well as the terrain elevation data model. The different sources of context information needs to be fused with the vehicle information to solve the problem.

The approach to data fusion taken in this work refers to fusion of the context information. To carry out this, the impact factors of all the various types of context information must first be determined and then, finally, the driveability cost is determined through a cost function. The impact factor of a particular context data type refers to the factors in the context that impact driveabilty.

3 Driveability impact factors

Driveability is a complicated matter which does not lend itself to simple solutions. It is affected by many factors, including the vehicle type, soil, weather, slope, etc. Although outside the scope of this work, a particular problem is to collect all the required information and represent it in a reasonably high resolution. Driveability is path dependent. It may e.g. be possible to drive down a slope, but not the other way around. Most terrain features cannot be regarded in isolation. Below are some examples:

- A road barrier, a large stone, or a tree can only be seen as obstacles if it is impossible to drive around them.
- A wide ditch may be seen as an obstacle to a driver who must cross it, but not to a driver who can drive inside or around it, or use a bridge.
- A single tree may not be an obstacle, but a dense forest can be a great impediment.
- A slope may not be too steep, but if the soil rigidity is too yielding it may be an obstacle. Vice versa, a mud field may be driveable on a plane area, but not at a slope.
- A ditch may not be driveable if it is filled with water, nor if trees or other obstacles are present in its proximity.
- Vegetation and crops have an impact on the driveability of soils. E.g., grass and grain often improve the driveability, while vineyards decrease it.

There are also other features, which are not intuitively possible to represent in a geometric model, but which may still affect the driveability. For example surface roughness or vegetation may decrease the speed, damage the vehicle, or bring it to a stop. Both stickiness and sinking soil properties affect the driveability, but possibly in different ways. Sometimes the effect is not strictly physical. Mine fields and radar installations may be examples of non-terrain features which decrease the wish to drive through such areas, whereas a forest may be a good place to hide. Strictly speaking, these aspects should instead be dealt with in threat analysis, although they still deserve to be mentioned in this context.

Many terrain features change their properties over time. Forests are growing denser and higher, rivers are bending, etc. During a war bridges, roads, etc., may be destroyed.

Weather properties may affect the terrain properties, thereby affecting the driveability. For example, soil rigidity is weather dependent. During a dry period a mud field may turn into a passable field. A cold winter may turn a river, that is otherwise impossible to pass, into a frozen road. However, it is not sufficient to know what the weather is like right now; historical values must be taken into account as well. The lake will not be frozen simply because the temperature falls below -10˚ C right now; it must have been cold enough for a longer period. Hard winds or floods may also affect the driveability.

Example of vehicle properties include width, length, height, override diameter, maximum gap to traverse, ground clearance, maximum step, maximum gradient, maximum tilt, specific ground pressure, and maximum straddle. Properties such as maximum speed is not of any direct interest to the driveability reasoning performed in this work, although it can be used as a basis for driveability calculations [15]. Sometimes the terrain features and the vehicles interact in unexpected ways. E.g., even if a vehicle can override small trees, the resulting pileup of vegetation may be a too great obstacle for it to pass. This effect is greater for wheeled vehicles than for tanks [17]. The properties are also not constant for the same feature. A ditch or a road, etc., may for instance grow wider, or fork, which is due to changes across the space.

To start with, it may not be necessary to know the exact value of a property. It may be enough to use qualitative values (e.g. *width* is equal to *large*), which can later be transformed into a real value, as is suggested by Bonasso [5], or by an interval as will be suggested in this paper. As a consequence of the complexity of the problem, the complexity must be reduced with respect to basic assumptions and simplifications, which will be discussed further subsequently.

4 Digital terrain model

The data used for creation of the 3D terrain model is registered by a scanning airborne laser-radar called Top-

Eye[1]. Uncertainties of the terrain elevation is in this case mainly dependant on the sampling density, which varies depending on flight elevation and speed. The data used in this work has an average density between 0.3 and 0.4 m. In order to detect significant terrain features, the original sensor data is firstly pre-processed to eliminate non-terrain data, e.g. trees and buildings. The pre-processing step produces a surface defined on a rectangular grid with 0.5 m between grid points.

As a first step, the surface is partitioned regularly into *tiles*, which is a sub surface that covers a square with sides 2 m. The set of tiles is called *T*. These tiles are sorted into a number of categories of tiles, the *spatial categories*. A spatial category is a set of tiles with similar features, e.g. with similar inclination or with edges at similar locations. The categories is defined by using a set of *representative* tiles $REP \subseteq T$. Three different types of representative tiles are necessary to achieve a sufficiently accurate surface model. The most important of which can be described by rotations and translations of the *basic category forms*, shown in figure 4. For further characterization see [2]. A distance metric $dist: T \times REP \rightarrow \Re_+$ for comparing the similarity of a tile to a representative is defined. A category $[r]$, $r \in REP$ is defined, using the metric *dist*, as the set of all tiles that are more similar to *r* than to any other $k \in REP$.

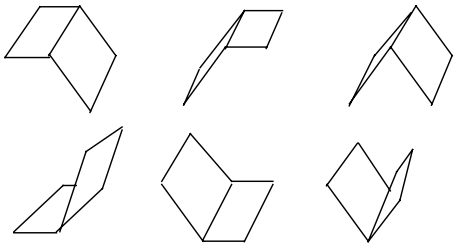


Figure 1: Basic category forms.

Every category corresponds to a unique string, called a *symbolic tile.* A tile is compared with every representative and are substituted by the corresponding string. The tiles can be stored in a database in accordance to a relatively simple structure [1], [18] from which data can be efficiently accessed.

4.1 Finding and connecting segments

A filter correspond to a particular sequence of connected symbolic tiles describing the feature that will be subject to the search. The approach taken here is to use the specific and characteristic cross-sections, which exist for all terrain formations. For instance, a cross-section of a hill can first be described with an upward directed slope followed by a slope directed downwards. At a conceptual level the cross-section of a terrain formation consist of a start segment, a possible middle segment and an end segment, see figure 2. The filters can be of different sizes depending on the requested terrain feature. Consequently, the search strategy must be sufficiently flexible to accommodate these demands.

It turns out that a two level search strategy is the most appropriate.As a result of the first matching step a large number of segments will be found. To find out

1. A property of TopEye AB, Sweden

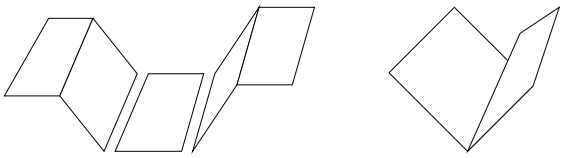


Figure 2: The filter structure to be applied to the cross-section of a ditch.

which of these segments that are part of the feature, adjacent segments of similar cross-sections must first be connected.Segments found in step 1 are connected only if the segments midpoints are sufficiently close and if the orientation of the cross-section segment is similar enough. Some methods for doing so is described in [2].

The filters are specified in a plain text file that is parsed by a Java-program and applied to the symbolic tile data file. A number of filter types for different features have been developed and tested. Among them can filters for determination of ditches, ridges, hill tops and flat areas including roads be mentioned.

At the same time as determining whether an object is present or not, an initial estimate of feature area and some impact factors can be calculated. In particular, slope angles of slopes, convexities and concavities, as well as elevation difference and width of concavities, fig 3. Apart from serving as impact factors for the driveability analysis, these values are used for post-detection refutation of features to exclude small and shallow features of negligible influence. Algorithms for acquiring better estimates of impact factors are under development, some possibilities are given in [3].

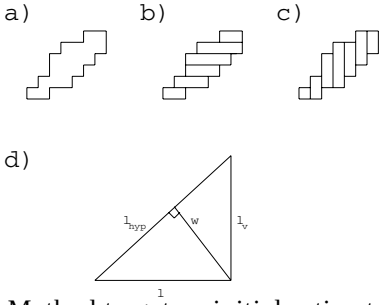


Figure 3: Method to get an initial estimate of feature width. a) The feature. b) Horizontal width. c) Vertical Width. d) Calculating feature width from horizontal and vertical width.

4.2 Geographic data

The major source of data besides the 3D-data is the real-estate map, which contains data about the geographic classification of the covered area in scale 1:10000. These data are organized in overlays, where each overlay represents a certain geo-class, e.g. buildings and ground classifications are found in different overlays. Apart from providing the driveability analysis with necessary data, the map provides a highly structured representation of an area of interest (AOI), which is adequate for user presentation and interaction. Consequently, when fusing the map data with other sources it is desirable to keep most map structures that are consistent with the more recently collected data. Although small, the

data uncertainty due to imperfections in the collection process can not be disregarded when fusion with other high-resolution sources is at hand. More importantly, generalizations due to aggregation, simplification, smoothing, exaggeration, displacement etc., [19], are typically made with consequences for the locational accuracy that is difficult to assess. Problems concerning the relative positions of the terrain features versus the map features inevitably arises, producing apparent contradictions. When serving as a basis to driveabilty analysis, another problem concerns the very rough, sometimes none, estimate given for most impact factors, e.g. the width of roads or the density of trees in a forest. In the absence of such values either defaults must be used or experienced users must be consulted for more specific estimates.

Even though the map is a general and important source of geo-class data there may be other sources of such data. For example [18], provides a way to locate roads, buildings and individual trees by using the elevation and intensity returns of the laser radar. Such data will improve the accuracy of some locational estimates as well as provide the driveability analysis with some of the otherwise absent values of impact factors, but will not be considered further in this paper.

5 Driveability Analysis

Driveability is a measure of the possibility of a certain vehicle to follow a path $p = (p_1,p_2)$ from the start p_1 to the end p_2. There are off course infinitely many such paths between p_1 and p_2 and there are infinitely many start and end points in a given AOI. Using a finite, but high resolution of 0.5 m still makes the analysis prohibitively expensive in computational terms. To reduce the computational complexity, as well as to lay a foundation for user interaction and spatial reasoning, another approach is suggested. Instead of viewing the AOI as an image, the AOI shall be viewed as a set of *objects.* These objects have attributes and belong to different classes. Consequently, in order to be able to calculate the driveability for a given AOI, a commitment to which objects, classes and attributes the driveability analysis recognizes has to be made. The objects of concern in this application will be called terrain objects (TO). The complete enumeration of the entities that constitute the terrain objects in this application differs depending on the available data sources and their implicit ontology. Any physical map entity that has a spatial extension, including roads, buildings, fences and similar line objects that direct influence driveability must be considered terrain objects. Every 3D terrain feature is also a terrain object. In analogy with the map, the result of applying a certain filter to the laser-radar data defines a separate overlay for each different filter. The terrain objects from different overlays must then be fused to construct a relevant basis for a driveability analysis, i.e. a *driveability segmentation*. The requirements of such *terrain object fusion* will be discussed below. Fusioned terrain objects will be called *compound terrain objects* (CTO).

A suitable driveability segmentation provides a complete segmentation of the AOI, where each CTO belong to three different types of classes. The types of classes are *3D classes*, *cover classes* and *obstacle classes*.

- 3D-classes = {Concave, Convex, Slope, Flat, Concave&Slope, Convex&Slope, Undetermined}
- Cover-classes = {Water, Mash, GroundVegetation, OpenGround, Road}
- Obstacle-classes = {Building, Forest, NoObstacle, DenseUrbanArea, Remains, LineObstacle}

Evidently, the Cover-classes and the Obstacle-classes contain (essentially) a subset of the classes available from the map. These classes are the necessary and sufficient classes to express the map data complexity in the current version of the driveability analysis. Each classification influences which calculations that are necessary and that which makes sense at all. The classes also differ in their default properties, in the way they are influenced by weather and their compactness. Furthermore, the obstacle classes can be both elements that are considered single entities (Building, LineObstacle) and elements, which are considered *aggregations* of smaller obstacles (Forest, DenseUrbanArea, Remains). Building, LineObstacle and DenseUrbanArea are compact objects, whereas Forest and Remains are not. The 3D-class “Undetermined” must be included as the terrain feature filtering can not be expected, as the map, to provide a complete segmentation of the AOI.

5.1 Path selection

Driveability, as defined above, is a concept that is meaningless without a path to refer to. In this work, the focus is on the analysis of meaningfully, i.e. qualitatively, different paths that are representative of an AOI. A path is qualitatively different than another path if it traverses a different CTO or if the traversal of a certain CTO is made in a qualitatively different way. The qualitatively different ways of traversing a CTO depends on its classifications. For instance, it is only meaningful to differentiate between paths going from A to B and from B to A if the area is sloping substantially. Also, it is only meaningful to consider “leaping” over an object if it is a concavity, not if it is a convexity. In this paper, however, the same type of paths are considered for every CTO. In fact, the paths are identified with traversal directions. The possible directions are D = {E,NE,N,NW,W,SW,S,SE}, representing travelling east, north-east etc.

The directions D can also be considered default paths that should be used for the 3D class Undetermined, i.e. if the CTO is labelled Undetermined, the qualitatively different paths traversing the CTO are members of D.

5.2 Requirements of Terrain Object Fusion

As mentioned, the data can be viewed as organized by a set of overlays, where each overlay contains a partial segmentation of the AOI. The fusion task at hand is to provide the driveability analysis with a complete, non-conflicting segmentation of the AOI, where all areas are CTOs. An

approach to constructing a driveability segmentation would be to simply intersect all overlays [4]. The problems that arise due to uncertainties and generalizations in the data, changes in the terrain and the fact that when some classifications coexist at a location a conflict is at hand. Examples of conflicts are CTOs classified as roads with crossing convexities or as concavity and water. These types of conflicts must be detected and resolved if the desired segmentations are to be achieved. As such, the terrain object fusion problem includes association, change detection and decision fusion problems commonly encountered in spatial data fusion [20]. An important consideration in this context is that the desired representation is not a driveability segmentation where the location of the objects from several sources are combined to give a more accurate locational estimate. Rather, the map representation should be kept for the earlier mentioned reasons. In particular, topological relations between TOs are of importance in the driveability segmentation. Even if not conflicting, without proper consideration, the driveability segmentation may consist of unnecessary many, small CTOs which defeat the entire idea of reducing the AOI into a manageable number of homogeneous entities. In the current version, the map are assumed to be correct and accurate in all aspects. If a conflict with 3D-data occurs, the map has precedence.

5.3 Impact Factor Attributes

Apart from determining which class of TO that are present at a certain location, some attributes, i.e. the impact factors, of the objects must be estimated as well. In this case *convexity gap width, maximum slope angle, minimum cover rigidity, obstacle rigidity, maximum obstacle rigidity* and *minimum obstacle distance*. The intended physical interpretation is indicated in table 1. Rigidity is used as a collection term for material properties, [5]. The corresponding attributes for the vehicles are *gap capability, slope up capability, slope down capability, ground pressure, force limit* and *vehicle width.* The use of maximum slope angle, minimum cover rigidity, maximum obstacle rigidity and minimum obstacle distance is connected to the fact that the slopes, covers and aggregated objects have different values at different parts of the single path under consideration. However, the value that determine the driveability of the path is the most disadvantageous value of the encountered values at any part of the path. A path between A and B going through a solid rock obstacle is not drivable not withstanding how easy the path part until encountering the rock may be.

Table 1:

Impact Factor	Interpretation
convexity gap width	Obvious.
maximum slope angle	The maximum angle of slope for all parts of a single path.
minimum cover rigidity	The surface ability to withstand pressure from above. Measured in pressure units.
obstacle rigidity	The obstacle ability to withstand force when being driven into. Measured in force units.

Table 1:

Impact Factor	Interpretation
maximum obstacle rigidity	The ability to withstand force when being driven into. Relevant for CTOs belonging to aggregated obstacle classes.
minimum obstacle distance	Relevant for CTOs belonging to aggregated obstacle classes.

5.4 Impact Factor Variation

As described earlier, driveability depends on a complex set of factors, many of which are difficult to assess and for which data is frequently missing. The actual formulation of the conditions that must be met for driveability is, however, not difficult to express once the particular case is determined. If it is the case that the path is going down a slope, through open ground and no obstacles are present, the slope angle must be compared with the vehicle capacity in this regard and the rigidity of the ground must be compared with the pressure to the ground exhorted by the vehicle. The result is true or false and the driveability is thus determined. This conclusion is only appropriate if the CTO is completely uniform over the entire object in all properties, e.g. the object has the same maximum slope angle for all paths of the considered class. Clearly this is rarely the case. This is a case of *spatial* variation for path s at individual TOs. Also, the performance characteristics of the vehicles of the same type is not uniform, i.e. *type* variation exists. Apart from the type variation, the vehicle capacity is dependent on, for instance, the carried load and the maintenance of the vehicle, i.e. *temporal* variation exists. Type and temporal variation is typically a result of the query of driveability not being specific enough, i.e. the query is put in terms of the driveability of a certain type of vehicle at some unspecified time instant.

In order to accommodate for the variations exhibited by terrain objects, as well as by the vehicles, an extension to the approach must be considered. Instead, consider the attributes to be variational quantities, i.e. the attribute value is different at different parts of the CTO. For instance, the concavity gap width varies somewhat at different locations and the maximum slope angle varies at different parts of a slope. In this work, variation will be described by an interval, [a,b], for every impact factor. This is equivalent to saying that considering different parts of the CTO, the minimum value of the impact factor is a and the maximum value is b. The variation of the maximum obstacle rigidity and the minimum obstacle distance connected with the aggregated classes are mainly connected to the fact that there are different types of smaller objects in the area and that the obstacle rigidity and obstacle distances varies between objects.

The variations described above can not, in general, be considered to describe the uncertainty of the true value of an impact factor. An estimate of the true width of a ditch at a certain location can be subject to uncertainty depending on the resolution of the data etc., but this is not what is described by an impact factor interval [a,b]. Neither should that interval be interpreted as saying that the probability of

the width at different locations is uniformly distributed over the interval. As already mentioned, the driveability is a measure of the *possibility* of travelling a certain path. Hence, it is not concerned with the length of the path, or, given a random choice of paths or vehicles, with the probability of success or with average performance. The choices are not random, but made by skilled operators. Therefore, the statistics of random choices are uninteresting, only possibilities are needed.

Still, a confidence measure that reflects the accuracy of the data and of the estimation process is useful to provide operators with an idea of the robustness of the analysis. Deciding that the variation of a width of a ditch is [a,b] from measuring two locations of a 100 m long ditch would clearly justify low confidence in such a description. For the 3D data, the confidence is essentially connected to the density/resolution of data after the ground filtering step, the properties of the ground filtering process and the age of the data. Currently, no measure of confidence is calculated, but all data necessary to do so is available.

Expert users or default reasoning can often provide a qualitative value for an impact factor or a vehicle capacity as an approximation. These qualitative values can be interpreted as intervals as well, i.e. using the value *SOFT* for a rigidity attribute can be interpreted as saying that the rigidly is in some interval [a,b]. Using this interpretation, the same theory can be used regardless of the source of the estimates. Hence, the absence of specific estimates as well as unspecifically made queries and variation in individuals can be modelled using interval valued variables.

5.5 Impact Conditions

As already mentioned, the conditions that need to be evaluated to determine driveability are not complicated. Due to the interpretation of impact factors and vehicle capacities as varying, the evaluation of the conditions are, however, not evident. As mentioned, the variation of vehicle capacities are both class variation and temporal variation, i.e. the entity for which driveability is determined is a generic instance of a certain class of vehicles and the determination is valid in some normal, possibly temporally changing, but wide range of conditions. In accordance with the above terminology, the conditions to evaluate are called *impact conditions* and the resulting values are called *impact costs*. An *impact condition* is a conjunction of relations between impact factors and vehicle capacities of the type:

(*impact factor*$_i$ relation *vehicle capacity*$_i$).

The only relations needed are "<=" and ">=". The possible relations between two intervals [a,b] and [c,d] are 13 as described in [21]. In this case, six out of these can be given distinct interpretations regarding driveability, see figure 3 for the relation "<=", u as an impact factor and v as a vehicle capacity. The interpretations of these interval relations are, in decreasing order of driveability:

1. All vehicles can pass at every location of the CTO.
2. Some vehicles can pass at every location, but some vehicles only at some locations of the CTO.
3. All vehicles can pass at some locations and none can pass at all locations of the CTO.
4. Some vehicles can pass at every location, some vehicles only at some locations and some vehicles can not pass at any location of the CTO.
5. Some vehicles can pass at some locations and some at no locations of the CTO.
6. No vehicle can pass at any location of the CTO.

Evaluating a single impact condition thus gives an impact cost in {1, 2, 3, 4, 5, 6}. Combining the values of impact costs from conjunctions of impact conditions are done by taking the maximum of the values from the individual conditions. The currently used CTO classifications and the impact conditions that are evaluated can be seen in table 2. The last entry in the table is a conjunction of the second and fifth entries.

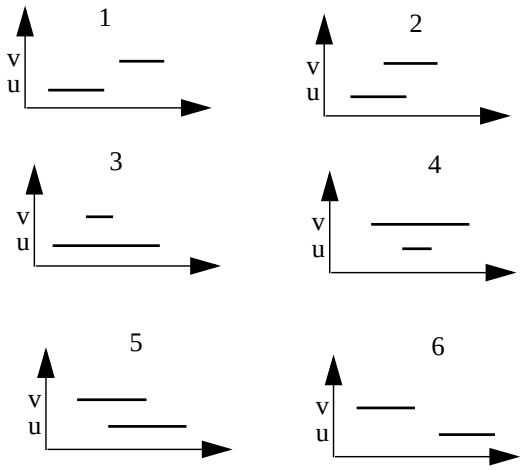


Figure 4: The interpretation of "u<=v" when considering u and v as intervals.

For any of the traversal directions D currently under consideration, the correspondence of that direction with a certain way to traverse the CTO must be determined, e.g. whether the direction corresponds to going down or up a slope. This is solved by approximating the orientations of the TOs as one of the directions in D. The 3D-class "Undetermined" must be included for the areas where no uniform feature can be determined by the filtering process. A CTO belonging to this class has e.g. different slope angles associated for the 8 directions in D.

3D class	cover class	obstacle class	way of traversing	impact condition
Concave	Any	NoObstacle	across	(gap width <= gap capability)
Convex, Concave, Convex&Slope, Concave&Slope	Any	NoObstacle	on surface	(maximum slope angle <= slope up capability) & (maximum slope angle <= slope down capability) & (minimum surface rigidity >= pressure)
Slope	Any	NoObstacle	on surface	(maximum slope angle <= slope up/down capability) & (minimum surface rigidity >= ground pressure)

3D class	cover class	obstacle class	way of traversing	impact condition
Any	Any	Building, LineObstacle	through obstacles	(obstacle rigidity <= vehicle force limit)
Flat	Any	Aggregated	on surface and/or through obstacles	(maximum obstacle rigidity <= vehicle force) & (minimum obstacle distance >= vehicle width)
Not Flat	Any	Aggregated	on surface and/or through obstacles	(maximum obstacle rigidity <= vehicle force) & (minimum obstacle distance >= vehicle width) & (maximum slope angle <= slope up capability) & (maximum slope angle <= slope down capability) & (minimum surface rigidity >= ground pressure)

Table 2: Some of the impact conditions for certain CTO classifications.

6 Results

An experimental tool for driveabilty analysis has been developed. Two types of vehicles has been used in the experiments so far. One wheeled troop carrier and one tank. The capacities of these vehicles are sometimes very difficult to obtain. This fact can be modelled by using a qualitative approximation. The value of a driveability analysis is off course dependent on the estimation accuracy of vehicle capacity but, as shown in for instance [4] and [5], qualitatively valued approximations can still be valuable.

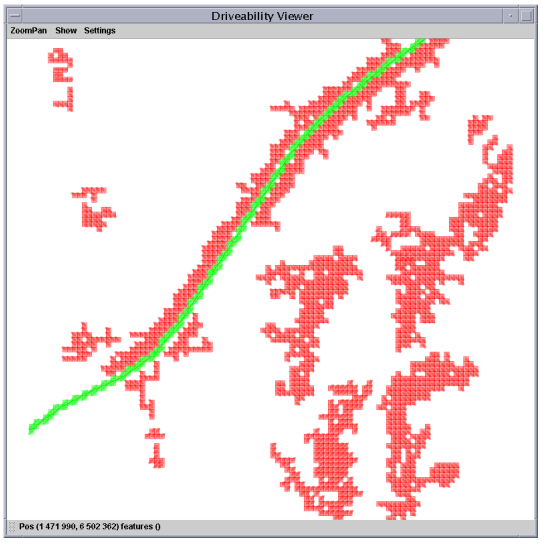


Figure 5: The driveability of a carrier. The direction of highest impact cost is selected for display.

As examples of driveability, a square AOI of sides 200 m is given in figures 5, 6 and 7. The area contains a road and some open areas in connection with these. The road is colored green in the figures because of its low driveability cost. Following the road on each side is ditches. Other concavities and slopes can also be seen. Visualization of the driveability is handled by color coding of the tiles belonging to the terrain objects. In this case impact costs = 6 are colored red and impact costs = 1 are colored green. All other impact costs are white.

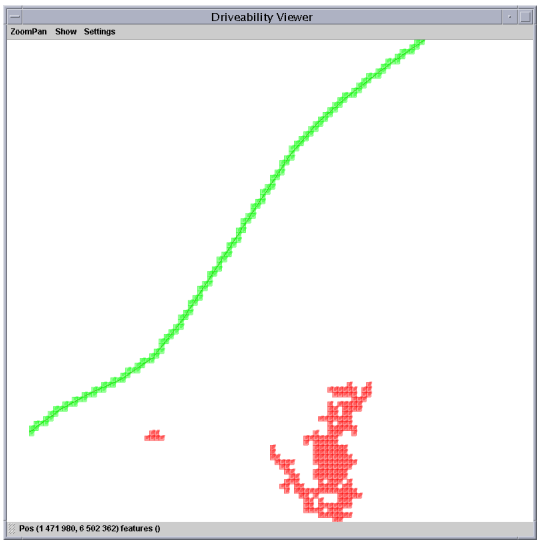


Figure 6: The driveability of the carrier in direction north-east.

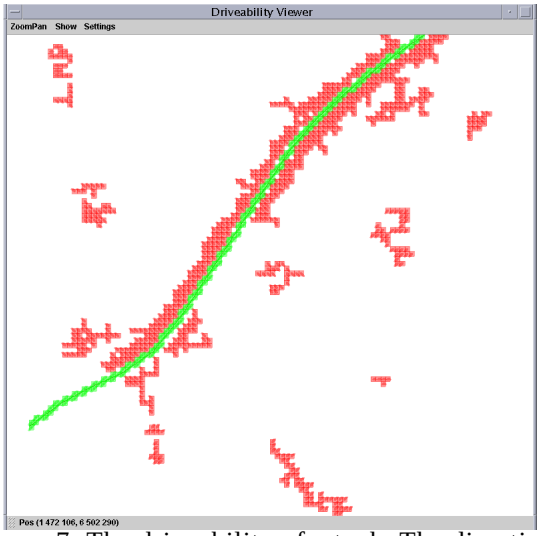


Figure 7: The driveability of a tank. The direction of highest impact cost is selected for display.

7 Conclusions and Further Work

Many data sources are necessary if successful driveability determination shall be possible. In particular, 3D data in high resolution is important to determine terrain features and their attributes. Fusion of context sources and vehicle information is required to enable adequate analysis of the properties of the context in which the vehicles move, i.e. context fusion is needed. Requirements on data sources and processes are described. A formulation of

driveability analysis in terms of terrain objects, impact factors and impact conditions is presented. The suggested method accounts for the dependence on both path and vehicle characteristics. Most terrain features and vehicles exhibits variation that can not be neglected. An approach to handling this phenomenon is presented. Handling confidence measures in the estimate of such variational qualities is still to be resolved. A prime candidate for locational uncertainty representation in this context is rough set theory, which is increasingly popular in handling geographic information.

References

[1] Lantz, F., Jungert E., *Dual aspects of multi-resolution grid-based terrain data model with supplementary irregular data points*, Proceedings of the 3rd International Conference on Information Fusion (Fusion'2000), Paris, France, July 10-13, 2000.

[2] Lantz, F., Jungert, E., Sjövall, M., *Determination of Terrain Features in a Terrain Model from Laser Radar Data*, Proceedings of the ISPRS Working Group III/3 Workshop on #-D Reconstruction from Airborne Laser scanner and InSAR Data, Dresden, Germany, October 8-10, 2003.

[3] Edlund, S., *Driveability analysis using a digital terrain model and map data*, LITH-IDA-EX-04/031-SE, Linköpings Universitet, Linköping, Sweden, Mars, 2004. Also available as Technical Report, FOI-R--1242--SE, Department of Data and Information Fusion, Linköping, Sweden, May, 2004.

[4] Donlon, J. J., Forbus, K. D., *Using a geographic Information System for qualitative spatial reasoning about trafficability*, Proceedings of the on Qualitative Reasoning (QR'99), Volume 4364, 1999.

[5] Bonasso, R. P., *Towards a naive theory of trafficability*, Proceeding of the Annual Conference on AI Systems in Government, 1989.

[6] Broten, G. S., Digney, B. L., *Perception for learned trafficability models*, Proceedings of the SPIE, volume 4715, 2002.

[7] Chaturvedi, P., Sung, E., Malcolm, A. A., Ibañez Guzmán, J., *Real-time identification of drivable areas in a semi-structured terrain for an autonomous ground vehicle*, Proceedings of the SPIE, volume 4364, 2001.

[8] Digney, B. L., *Learned trafficability models*, Proceedings of SPIE, volume 4364, 2001.

[9] Ducksbury, P. G., *Driveable region segmentation using a Pearl Bayes network*, In IEE Colloquium on Image Processing for Transport Applications, Digist No. 1993/236, 1993.

[10] Grunes, A., Sherlock, J. F., *Texture segmentation of defining drivable regions*, Proceedings of the British Machine Vision conference (BMV'90), 1990.

[11] Jasiobedzki, P., *Detecting drivable floor regions*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1995.

[12] Johnson, A. J., Windesheim, E., Brockhaus, J., *Hyperspectral imagery for trafficability analysis*, Proceedings of the IEEE Aerospace Conference, 1998.

[13] Kruse, F. A., Boardman, J. W., Lefkoff, A. B., *Extraction of compositional information for trafficability mapping for hyperspectral data*, In Algorithms for Multispectral, Hyperspectral and Ultraspectral Imagery IV, 2000.

[14] Slocum, K. R., Surdu J. R., Sullivan, J., Rudak, M., Colvin N., Gates, C., *Trafficability Analysis Engine*, Cross Talk, The Journal of Defense Software Engineering, June 2003, pp 28-30.

[15] Sapounas, D., Kreitzberg, T., Johnson, M. L., *Terrain trafficability model*, In Military, Government and Aerospace Simulation, Proceedings of the 1996 Simulation Multi Conference, 1996.

[16] Glinton, R., Grindle, C., Giampapa, J., Lewis, M., Owens, S., Sycara, K., *Terrain-based information fusion and inference*, Proceedings of the 7th International Conference on Information Fusion (Fusion'04), Stockholm, Sweden, June 28-July 1, 2004.

[17] Department of the Army, FM 5-33, *US Army Field Manual*, *Terrain Analysis*, URL http:// www.global sequrity.org/military/library/policy/ army/fm/5-33/default.htm, July 1990.

[18] Elmqvist, Jungert, Lantz, Persson, Söderman, *Terrain Modelling and Analysis Using Laser Scanner Data*, Proceedings of the Workshop on Land Surface Mapping and Characterization Using Laser Altimetry, Annapolis, Maryland, October 22-24, 2001.

[19] Dunkars, M.,*Multiple representation databases for topographic information*, Ph.D. Thesis TRITA-INFRA 04-036, Royal Institute of Technology, Stockholm, Sweden, December, 2004.

[20] Waltz, E., *The principles and Practice of Image and Spatial Data Fusion* in Handbook of Multisensor data fusion, Hall, D., Llinas, J., (Eds), CRC Press LLC, Boca Raton, Florida, USA, 2001.

[21] Allen, J., F., *Maintaining knowledge about temporal intervals*, Communications of the ACM 26, pp. 832-843, 1983.

Appendix K

Towards a Query Assisted Tool for Situation Assessment

Information Fusion, Annapolis, MD, July, 2002.

Fransson, J., Jungert, E.

Towards a Query Assisted Tool for Situation Assessment

Jörgen Fransson, Erland Jungert
Swedish Defence Research Agency (FOI)
Box 1165, SE–581 11 Linköping, Sweden
{jorfra, jungert}@lin.foa.se

Abstract - *Abstract: A critical problem when designing systems to support interpretation of real-world situations is how to handle possible alternative world views. This is especially true in situation assessment (SA) where the complexity of the task often requires many diverse methods to be used. If combined in an ad-hoc way, there is little or no chance of finding alternative solutions in a structured and reliable way. To overcome this problem a novel framework for SA, capable of exploring alternative interpretations in cooperation with a user is proposed. The specific problem addressed in this framework is compilation of a tactical ground picture in a situation with sensor observations that are highly discontinuous in space and time. To assist the users a query language for multiple heterogeneous data sources is attached to the framework. The query language includes methods for target recognition and for fusion of the acquired information. Terrain information in both high and low resolution will also be available. Finally, it will be demonstrated how information acquired from the query language can be used to support the analysis in the situation assessment process with the aid of the proposed framework.*

Keywords: Situation modelling, situation analysis, evolutionary algorithms, decision support, command and control system, query languages, information fusion.

1 Introduction

A critical problem when designing systems to support interpretation of real-world situations is how to handle alternatives. This is especially true in military applications, where many types of uncertainties exist and the cost of mis-interpretation can be very high. Implementation approaches in the area of Situation Assessment (SA) are often patchworks of different methods such as rule-based deduction, clustering, neural network classification, etc. The use of such diverse methods may be necessary due to the complexity of the task, but combined in an ad-hoc way it becomes very difficult to consider alternative solutions. Therefore, in this work a novel framework for SA capable of exploring alternative interpretations in cooperation with an operator is proposed.

The specific problem under consideration in this paper is that of tactical ground picture (TGP) compilation, which in the longer perspective will be extended to cover a broader range of SA tasks. In order to design a framework for high-level user support of this task, there is a need to define what the important problem characteristics are and what consequences they should have on the framework level. Since TGP compilation is both a very complex and in some aspects not a well-defined problem, there has to be assumptions made on different levels.

Query languages are traditionally instruments for accessing information from various types of databases. Lately, there has been a growing interest for the use of query languages for acquisition of information generated by multiple sensors. Data from multiple sensors are generally of heterogeneous type. Consequently, a query language for multiple data sources where the data are generated by different types of sensors must include, not only, methods for target recognition but methods for fusion of the acquired information are required as well. An approach to such a query language, called σql that can be used for recognition of ground targets such as vehicles can be used as a tool for this purpose. Beside this, there is also a need for acquisition of ground data including terrain information in both high and low resolution. This work will also demonstrate how sensor data can be used for the determination of terrain models to support the analysis in the situation assessment process through a specific query technique. It will thus in this work be demonstrated how these two query techniques together can be used to assist the users in the situation assessment process.

This work is organized as follows. In section 2 the objectives the work are presented and in section 3 the characteristics of the TGP compilation problem and the consequences for framework design is discussed. In section 4 the situation assessment system introduced including also the two query language approaches integrated to support the SA process. The proposed framework for situation assessment is presented in section 5 and finally in section 6 the conclusions and the future research are discussed.

2 Objectives

Computer based SA in a ground scenario is a very complex problem with several layers, each of which has unresolved problems. We have two objectives at this stage. The first one is to design a general information system for SA which supplies the necessary basic information, i.e detected targets, terrain information, etc. This is based on query languages to achieve the flexibility needed to handle different types of tasks and user needs. The second objective is to design a framework for high-level user support, where a basically autonomous problem-solving capacity is coupled with mechanisms for man-machine cooperation.

3 Problem discussion

Data- and information fusion in support of SA in military ground situations is a complex area where relatively little progress has been made. In view of these difficulties the somewhat more limited task of TGP compilation is chosen as a starting point. Both share the basic problem of limited information due to imperfect sensing functions, i.e all targets are not visible (detected) all the time. There are also common problems in determining and formalize the knowledge there is about expected capacity, behavior and organization of military units. The notion of a TGP usually means some form of description of the current status and history of enemy forces; where they are, where they have been and their type. This description can be of different levels of detail. Traditionally it has been prepared as a graphical overlay on a map, where it usually, as far as possible, refers to known organizational units, readily depicted with military symbols. The process by which intelligence analysts manually compiles a TGP is described in military handbooks as part of military doctrines.

3.1 The problem approach

The main task for TGP compilation in this work is defined to be that of object correlation. By this we mean association of observations originating from the same entity, i.e a vehicle, a group of vehicles or a military unit. If aggregation and subsequent classification of groups of entities into units is performed, it is only of supportive nature. This may seem somewhat surprising as the task of correlation often is viewed as a supporting, lower level task. A somewhat philosophical motivation for this is the following. Suppose that we had ”perfect” sensors which could detect, track and classify individual vehicles. In such a case it could be argued that classification of military units would only be of secondary importance; the most relevant information would be found in a presentation of the objects with good visualization techniques. This would imply that the primary reason for aggregation and subsequent classification of military units is that we do not have these perfect sensors. With more and better sensor information the possibilities of object correlation will increase. By not having the goal of hierarchical aggregation and classification of military units there is also less dependence on that type of a priori knowledge. A further argument is that by setting correlation in the foreground, the task of TGP compilation can be seen as a generalization of the concept of group tracking.

In this framework the input to the TGP compilation comes from a subsystem for target recognition, which includes multisensor data fusion capabilities, including tracking. This means that local classification and correlation of individual objects has been performed. To understand the nature of the correlation task faced at the higher level the following assumption is introduced:

Assumption 1: *Fragmented sensor observations are the key characteristic of the problem*

This is a fundamental assumption of this approach. We assume that situations where the sensor coverage is highly discontinuous in time and space is the single most important characteristic of the problem. While some degree of fragmentation almost always can be expected, we here push this issue to the limit. There is a growing interest in relatively inexpensive, expendable sensors and sensor platforms, e.g ground sensor nets, micro-UAV’s, reconnaissance grenades, etc. It seems very likely that distributed use of such systems will result in highly fragmented TGPs. Even in the case of more traditional sensor systems, the ability to handle fragmentation, resulting from such things as terrain masking, limited coverage, etc., will be a key issue.

The primary interest here lies in small scale scenarios. A typical example could be the following. A battalion-sized mechanized unit is advancing, both on- and off road. A few sensor systems make some observations of parts of the unit over a significant time-span. Sensor systems could, for instance, be a UAV - with different combinations of radar, infrared- and laser-imaging sensors - and acoustic ground sensors at a few selected places. The non-continuous observations give rise to hard correlation problems, e.g ”Do the group of vehicles observed at road-junction X belong to the same group as was observed 10 minutes before in area Y?”. Typical questions are about numbers and types of vehicles in the observation area. TGP compilation in a mini-scenario such as this bears some resemblance to the task of group tracking, [9], [10], [11]. One similarity is the task of correlating observations of a *group* of objects. There are however at least two important differences. The first one is that correlation in TGP compilation, due to the larger time-gaps, must be based on (or augmented with) other types of domain knowledge and factors than the kinematic models which typically dominate in group tracking (although there are exceptions here).

It is outside the scope of this paper to make an analysis of what domain knowledge that can be used for TGP compilation of the type defined above. We do however envisage the use of the following general areas. The concept of *group similarity* is central. What makes this a very difficult problem area is that we must account for the possibility of partial observations of groups. In special circumstances, i.e flat, open terrain with high detection rates, this may be less of a problem. But in general, the effects of terrain masking, limitations in detection processes, sensor field-of-view limitations, etc., make partial observations a thing to be expected. An example of a bayesian model for handling partial observation of a group is given in [20]. The problem is not only in mathe-

matical models, but also in the data needed. For instance, to take terrain masking into account there must be a terrain database with high resolution available, detection rates must come from somewhere, etc. Models of *terrain trafficability* and *vehicle/unit speed* is obviously needed. In the area of *military unit behavior* our intent is to strive for as simple models as possible; more of "rational behavior" rather than military doctrines.

3.2 Framework design issues

Designing a system to support a user requires consideration not only of the problem and its technological solution, but also of human factors. An interesting discussion of this can be found in [19], where a triad model is proposed that establishes the relationship between task, technology and human in the design of systems that support human decision making. The starting point is an analysis of task characteristics and the constraints imposed from the environment. Possibilities and limitations should then be identified both in the technology-task and the human-task relation. It is then in the trade-off spectrum of identified requirements on both technology and human that the roles of man and machine are defined.

3.2.1 Key factors in TGP problem characteristics

The problem characteristics as discussed in 3.1 is based on the assumption of fragmented sensor observations. Partly based on that assumption, the following assumption can also be made:

Assumption 2: *The existence of relevant alternative interpretations is not uncommon*.

The ambiguities created due to fragmentation, in combination with uncertainties in the knowledge applied to make an interpretation, can generally result in more than one plausible interpretation. The assumption is that these plausible interpretations are enough different (from the user perspective) that we can speak of different interpretations rather than a single interpretation with minor uncertainties.

Assumption 3: *The domain knowledge is often uncertain in a hard-to-quantify way.*

This especially applies to domain knowledge in the area of military doctrines.

3.2.2 Key factors in problem solving technology in relation to the TGP problem

Assumption 4: *The combinatorial explosion caused by uncertain correlations and aggregations make exhaustive search impossible*.

This is an assumption only for very small situations, otherwise it is a fact.

3.2.3 Key factors in user needs and capabilities in relation to the TGP problem

Assumption 5: *The user must somehow gain an overview of plausible alternative interpretations.*

The notion of alternatives is central to the military decision making process. It is therefore not enough to find a single very plausible interpretation, if not other possibilities has been checked and rejected.

Assumption 6: *The relevance of differences between alternative interpretations is context- and task dependent in such a way that human control is needed.*

This means that if we have a system that searches for alternative interpretations, the user must somehow be able to influence the underlying definition of "relevant alternative interpretation". This has to do both with computational priorities (the "Focus of attention") and with the user needs of not getting overwhelmed with unimportant alternatives.

Assumption 7: *The user has domain knowledge at least part of which is not represented in the system.*

Assumption 8: *The user has superior judgement compared with the system.*

These two general assumptions holds for almost any real-world decision support system with an experienced user.

Assumption 9: *The user has enough time available to be able to interact with the system.*

While the time available for analysis has task dependent limits, we do not think that this is the most critical problem.

There are several important consequences for framework design that can be derived from these assumptions. One of the most fundamental is that *some form of overview of plausible alternative solutions must be given to the user*. This is a direct consequence of assumption 4, but it is also motivated by the other assumptions. If the domain knowledge was more statistical in nature, it might be possible to rely on the system to make the most optimal interpretation. The opposite is a system where only one solution is given. In such a case it is more of a autonomous problem solving system, where the users domain knowledge and intuition is of less use.

Another important consequence is that *the user must be able to influence the search for solutions*. The most obvious reason is that the limited computing resources available must be used in the most efficient way. In the end it is only the user who can decide where it is *relevant* to search. The more autonomous the system is, the greater the risk is that it will spend its time searching among alternative solutions where the differences are irrelevant to the user. It also has to do with superior judgement, and sometimes domain knowledge, on part of the user. In case of conflicts between the opinions of the user and the system, there must be mechanisms which allows the user not only to override the conclusions of the system, but also to make the system *revise* its own reasoning with the new pieces of information in consideration. There must be

mechanisms that allows *cooperative problem solving* between man and machine.

If user judgement and domain knowledge is to be used effectively, the *proposed solutions must be as transparent as possible to the user*, preferably down to knowledge base level. This is because the user in case of conflicting views with the system must have a chance of pinpointing the reasons for this. For this to be of any value, the user must also have mechanisms to *control the systems knowledgebase.* If the user decides that a specific part of the systems knowledge base leads to wrong conclusions, in general or in a specific context, there must be mechanisms that allows modification (or at least blockading) of that piece of the knowledge base. An example of this could be if the actual behavior of military units deviates heavily from (expected) doctrine.

3.3 Alternative interpretations

The notion of alternative interpretations is central to the proposed framework. The general aim is to explore plausible interpretations, which is a kind of *learning process*.

Definition: *Alternative interpretations are a set of solutions to the interpretation optimization problem, where the pairwize differences are great enough according to some specified metric.*

From a practical point of view, what is of interest is to find and present a limited set of plausible solutions which in some way are representative of the interpretative possibilities. There is a trade-off here between degree of plausibility and originality. Even if there is no precise definition of this, we can at least state that solutions must be different enough from one another to be of interest. The solution difference metric and the threshold for relevant differences must be problem dependent. In the case of TGP compilation, it depends on the perceived situation, user task, etc. Generally, what information the user is interested in should be reflected in the difference metric. It could for example be the overall number of vehicles present, organizational structure, etc. Ideally, the user should be able to dynamically shift this kind of focus. It should be a user *tool* for exploration.

In order to choose or develop techniques that makes it possible to find alternative solutions in an optimization problem, there is a need to understand the reasons to why the alternatives exist, i.e why the user could be interested in solutions other than the “optimal” one found by the optimization process. There can of course be many different reasons for this. In interpretation applications, and especially military ones, there is a need to take all possibilities of high enough plausibility into consideration to be able to make an optimal decision for some action. Another reason is imprecision in the optimization model, which means that a near-optimal solution may be the best one in reality. A third case is when there are several factors in the optimization model which are either hard to combine in an objective way, or of uncertain quality. If summed up in a single optimization objective in an ad hoc way there is a risk of losing interesting solutions because of the factors being integrated in a “wrong” way or because one or more factors being misleading.

In TGP compilation, all these reasons seems to be applicable. One important aspect is that it can be expected that the user has domain knowledge which is not present in the optimization model. Another aspect is that there are very different kind of factors which may be considered. On one side there is concrete evidence from sensors. On the other side there are different kind of a priori knowledge (or expectations/assumptions) of vehicle and unit behavior, organizational structure, etc. Much of this knowledge is non-statistical and hard to quantify in a meaningful way. If these different factors are combined to a single optimization objective, information is lost. The risk is that some piece of knowledge turns out to be misleading. It could for example be that the observed units are not organized or do not behave according to the expected doctrine.

4 Situation assessment system

The system described here is primarily intended for situation assessment. To support this type of operations two different types of query systems will be integrated. The first one is concerned with acquisition of vehicles from multiple heterogeneous data sources, i.e. basically sensors. As in all systems with multiple sensor data input sensor data fusion is needed here as well to gain more reliable results from the query process. Other information of importance to support the situation assessment process is terrain data; basically for determination of various kinds of terrain features, trafficability, change detection etc. To support these requirements a particular query system with the ability to fulfill such requests is being designed.

Input data to the SA module will be delivered from the query systems and is consequently not an entirely automatic process. Instead data are first collected by a set of sensors that may vary in type depending on availability and the application at hand. Once data have been collected by the sensors they will be transformed into a unified structure and objects of interest will be identified together with their most important attributes such as type, size etc. Besides, status information concerning, for instance, the object location, orientation and speed are determined. Such information as, for instance type, is always associated with some degree of uncertainty and for this reason a confidence value that mirrors the actual level of uncertainty must be associated with the targets. The information acquired from the various types of sensor data will be delivered to a database and stored for future use. Laser-radar data is used for generation of the terrain data model, which can be represented in a very high resolution.

4.1 The system structure

The structure of the situation assessment system can be seen in figure 1. The two query systems will both be reached from the graphical user interface. Queries for both query languages will be given in terms of a high level language and will include such aspects as type of vehicle or various *aggregated units*, *area of interest* and *time interval of interest* as well as *attribute* and *status variables of interest*. The queries may also include various constraints with regards to the attributes of interest. The queries are in a first step translated into a medium level

representation that in the next step will be transformed into a structure that can be used as input to the query interpreters. The former step is performed by the query decoder and involves reduction of the occurring high-level concepts or terms into low-level terms that are more suitable for the queries. Examples of such transformations are to determine the number and types of vehicles included in a specific unit, i.e. if the user query is concerned with a special unit then it is transformed into a specification including type and number of the vehicles; this specification is then used by the query system to answer the query, e.g. to determine whether these types of vehicles are present in the data and how many. The result of the queries may then be transformed by one or more of the information fusion agents, depending on the type of the query. The two query approaches will be discussed further subsequently while the methods used by the SA module will be discussed in section 5. Other special purpose modules may also be integrated into the SA system, for instance, a module for threat analysis can be integrated here if there is a need. The structure of this system is related to the JDL process architecture as described in e.g. [1].

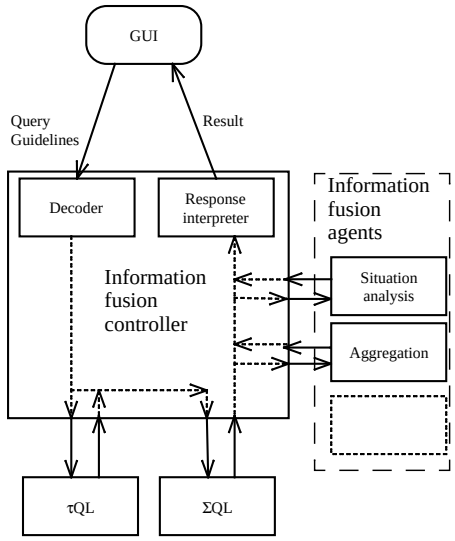


Figure 1: The structure of the situation assessment system.

4.2 The ΣQL query language

The query language, ΣQL, described in e.g. [2][3]and [5] can be seen as a tool for the handling of spatial/temporal information for sensor-based information fusion, because most sensors are generating spatial information in a temporal sequential manner. A query language of this type must be able to handle large volumes of data because most sensors can generate large quantities of data within very short periods of time. Another aspect to consider is that user queries may include data from more than one sensor, which consequently will lead to complex query structures, because the use of data from more than one sensor may require fusion of multiple sensor information. The strength of the query structure is its simplicity: the query language is based upon a single operator type, i.e. the σ–operator. Another advantage of the concept is the natural and simple mapping of ΣQL-structures into an SQL-like query language. However, the SQL-like query language is primarily useful just in theoretical investigations, while the σ–query language is easy to implement and also a step towards a user-friendly visual query language.

The σ–query language can be seen as tool that is applied to a data source corresponding to a multidimensional space. This *source, R*, is also called a *universe.* Each query is made up by a sequence of σ-operators that primarily should allow operations on a sensor-data-independent level, i.e. the acquired sensor data should be transformed into an information structure at a high abstraction level that is sensor independent. To accomplish this, the queries should be expressed in terms of operator sequences where the operators carry out the transformations stepwise. Basically, the operators reduce the dimensions of the multi-dimensional search space to which each new operator is applied with respect to the dimensions in focus of the query. The reduced search space is subsequently called a *cluster*. Thus, as new operators are applied, the clusters become more and more refined until eventually a final cluster is returned and this cluster corresponds to the answer of the applied query.

An illustration of the query language could be a video sequence, i.e. the universe R, from which a limited set of frames can be extracted. The σ-operators correspond to a select-command in SQL. Thus if we are interested in three frames at different predetermined times, t_1, t_2, and t_3, along the time axis, this will correspond to the σ-operator σ_t (t_1 , t_2 , t_3), which means that the three frames should be selected from the time axis of the universe R. In the SQL-like language the ΣQL query is expressed as:

SELECT t
CLUSTER t_1, t_2, t_3
FROM R

A new keyword "CLUSTER" is introduced, so that the parameters for the σ–operator can be listed. The word "CLUSTER" indicates that objects belonging to the same cluster must share some common characteristics (such as having a value of the value set of the same time parameter).

The dual representation of the ΣQL language means that a query can be formulated as an SQL-like query [2] or as a sequence of *generic* operators (the σ-operators introduced above). Translation from one representation to the other is straightforward. However, the main purpose of the operator sequences is to serve as an intermediate representation between the graphical user interface and the query interpreter.

The purpose of the concept of sensor data independence that was introduced above is to simplify the use of the system and to let the system take the responsibility of deciding which sensor and which sensor data analysis algorithm that should be applied under given circumstances as a response to a particular query. To support this activity an ontological knowledge base system [4] has been developed. This is a step towards general techniques to generate/refine queries based upon incomplete knowledge about the real world. However, the knowledge stored in the ontology differs from knowledge in other domains in that it includes not just object knowledge but sensor and sensor data control knowledge as well. An important consequence of the ontological knowledge base is that it

permits refinement and optimization of the queries since external information, such as weather and light conditions, can be maintained by the ontology as well.

A general and most important aspect of any query system but in particular in sensor data fusion systems, is the confidence in the query result, which must be acknowledged by the user. This is due to the fact that data acquired from sensors are always mapping the reality with some level of uncertainty. The uncertainties are due to technical imperfections in the sensors. Generally, these uncertainties can be represented with some kind of confidence value that may be normalized, i.e. they are given values within the interval [0,1]. Confidence values of this type should be interpreted as the confidence a user may have in a query result. This way of representing uncertainties in the data becomes even more necessary in the sensor data fusion process. Consequently, when evaluating the result from a query applied to data from multiple sensors the confidence value corresponding to the uncertainties of the fused result is required. This kind of confidence structure is used in ΣQL to support the user in interpreting the query result.

4.3 Terrain data query support

The situation assessment process should be concerned not just with vehicle information, their type and class membership as well as their status and attribute information. Terrain information is also needed to complete the SA process. For instance, determination of whether a particular part of the terrain is drivable is important and in practice this means that existing obstacles, such as ditches, must be determined. Here as well, a query language is required to support these demands. The query language, called τql [6], uses both a traditional terrain structure and a symbolic terrain representation, which are generated from ladar data. This terrain structure is represented in a very high resolution. The motivation for symbolic terrain structure is due to the observation that when terrain models in very high resolution are used, the number of data points tend to become extremely large which is very unpractical. Thus execution times tend to become very long. To handle this problem there is a need for a considerable reduction of the stored data. Basically, in this work this has been accomplished by development of a terrain model based on a grid structure that is combined with a set significant irregular data points [7]. This combined structure allows the terrain model to be represented in a resolution that reduces the number of stored data points with a bout 80 to 90% without loss of any considerable amount of information. The resolution of the terrain model is approximately 0.5 m.

The symbolic terrain data model can be viewed as a set of tiles where each tile is spanned up by the four corners of the grid structure including also the significant irregular data points. Consequently, each tile describes a part of the terrain in a characteristic way that is symbolically interpreted. Thus, each tile corresponds to a mapping of the terrain, which reduces the original data even further. The symbolic tiles are called categories and 115 such tile categories have been identified. To identify a certain terrain feature, e.g. a ditch, a filtering technique has been developed. The filters are made up by a set of symbolic tiles that are organized as a linear sequence that forms a segment. This segment, or filter, is matched against the tile structure. The matching process is performed both horizontally and vertically. The filter matching work is part of a recent masterwork [8], which is a step towards the development of a query language for terrain data, which is subject to on-going research.

5 A framework for support of tactical ground picture compilation

The general purpose of the proposed framework, see fig 2, is to allow flexible man-machine co-operation in the interpretation of sensor observations from a military situation, when viewed as a combinatorial optimization problem. The main feature of the framework is that it supports the exploration of alternative interpretations
.

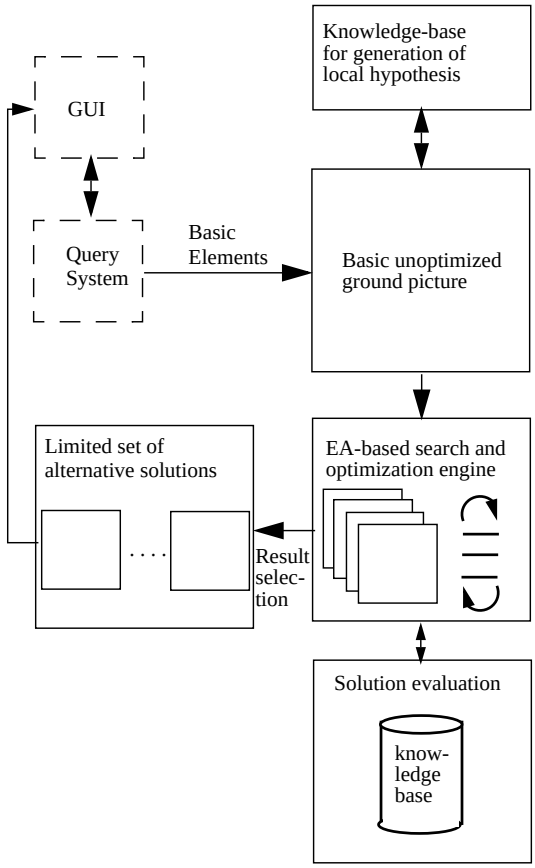


Figure 2: The framework for support of tactical ground picture compilation.

5.1 Search and optimization methods

The framework supports two types of optimization, both of which generates multiple solutions in some way. *Multi Modal Optimization (MMO)* [16], is concerned with optimization where the search space exhibits many local optima. The meta-heuristic here is that these local optima are interesting alternative solutions to the user if they are of good enough quality. There is a trade-off between solution quality and solution uniqueness. The other optimization technique is *Multi Objective Optimization (MOO).* Most real-world optimization problems has several

incommensurable and often conflicting objectives. MOO is concerned with the task of optimization with several objectives simultaneously. In single-objective optimization approaches such tasks can be handled by combining all objectives into a new objective according to some formula, e.g a weighted sum. The problem with such approach is that it can be difficult to combine different types of objectives in a quantitative way. In MOO objectives are kept separate. The optimization process then searches for solutions which are *pareto-optimal*. These solutions are optimal in the sense that no other solutions are superior to them when all objectives are considered. Some good overviews are given in [13] and [15].

Both MMO and MOO are implemented within the general paradigm of Evolutionary Algorithms (EA). This is an umbrella term used to describe computerized problem solving systems which use computational models of some of the known mechanisms of evolution as key elements in their design and implementation. Basic characteristics are that they maintain a population of tentative solutions to the problem to be solved, and that the population evolves according to rules which simulates evolution. A typical EA-loop consists of *mutation*, *selection* and *recombination*. Mutation means application of a local search operator to a solution. Selection is the equivalent of "survival of the fittest". Recombination means generating a new solution from two existing solutions, where the new one inherits properties from both "parents". EAs belong to the class of randomized search algorithms, but are often mixed with other methods to create hybrid systems. An introduction to the basic ideas of EA can be found in [12]. The paradigm of EA suits both MOO and MMO, partly because it works in parallel on a population of solutions, partly because of its inherent flexibility [14].

5.2 Framework structure

The central building-block of the framework is an EA-based *search and optimization engine*. It has a population of solutions to the TGP problem, which evolves in an ongoing loop of mutation, selection and recombination. The detected objects (including a track history) are a static background common to all solutions. A solution is represented as a set of hypothesized aggregations and correlations which links the detected objects in a consistent way. Mutation in this context means modifying (adding/removing/substituting) one of these hypothesis.
The techniques of MMO and MOO are treated as different *modes* of the optimization process, sharing the same basic solution structure. The optimization is an ongoing process where the user at any moment can inspect the most interesting solutions and change the focus of the continued search/optimization by changing optimization mode and/or their parameters.

The building blocks for the optimization process are stored in the *basic unoptimized ground picture* database. Here we have 1) the data from the multisensor data fusion system which has been requested by the user, i.e detected targets and tracks within a time and space frame 2) aggregation and correlation hypothesis. The hypothesis may be, and generally are, in conflict with each other. The main task of the basic unoptimized ground picture database is to feed the optimization engine with search alternatives, but it should also be visually inspectable by the users.

The aggregation and correlation hypothesis are generated in the *knowledgebased system for local hypothesis generation*. It is here possible to have multiple independent knowledge sources that generate hypothesis independent of each other, as there is no consistency requirement. Each hypothesis is given an estimated confidence value to enable the search and optimization process to implement different strategies, e.g to search among the most plausible alternatives first in order to quickly generate a solution.

Solutions in the search and optimization process are evaluated by a separate subsystem for *solution evaluation*. Solutions are evaluated in a hierarchical manner, from individual aggregations and correlations up to tracks of aggregated entities, groups of tracks, and finally the whole solution. To enable multiobjective optimization, the solutions are evaluated with multiple criteria. The two basic criteria are "plausibility in relation to sensor observations" and "plausibility in relation to expected behavior/doctrine". There is flexibility to define new, more specialized criteria.

The output of the search and optimization process is a limited set of solutions presented graphically to the user. They are intended to give an overview of different interpretations (i.e different TGP). By choosing optimization mode and its parameters, the user should be able to effect the criteria by which these solutions are selected.

5.3 Research issues

A *user-defined solution distance metric* is a high priority research issue. Another primary concern is about *computational complexity*. Both multi modal and multi objective optimization have high computational cost, and there is much research going on how to improve the efficiency of these. There has however been successful applications even in the case of combinatorial optimization. The concern about computational efficiency also applies to the evaluation of solutions. Another area which needs research is *solution representations and corresponding evolutionary operators.* There are a number of representations and operators for different kinds of combinatorial optimizations in the literature; the most applicable one in this context being that of set partitioning [18]. An open research issue of great importance, which is quite application specific, is how to design evolutionary operators so that both aggregation and correlation can be combined in the same solution structure.

5.4 Framework properties

Of the demands discussed in 3.2, it is the notion of *exploration of alternatives* which has been given most explicit attention. The challenge is both to do this in an efficient way and to get alternative solutions that are reasonably representative of the inherent possibilities. *User influence over search* is to be achieved by selecting optimization mode and by a user defined similarity metric. *Transparent solutions* can in principle be achieved by storing partial results from the solution evaluation process in the solutions. Some of this is needed for the MOO method. How to achieve *user control of applied domain knowledge* is

unclear, but simple blockage of parts of the knowledge is trivial with the intended modularity.

There are also other potential advantages with the proposed framework, largely attributable to the used paradigm of EA. The modularization of the problem solving process gives great flexibility to integrate other methods in the framework. For instance, if there is a solution generated by some other method, it can in principle be directly inserted into the population of the EA search machinery, as long as the solution structure is the same. Likewise, it is straightforward to use multiple methods for generation of local hypothesis, as there is no need for consistency.

6 Conclusions and future work

In this work a system for situation assessment has been proposed. The approach taken includes, in order to assist the end-users in their working processes, two different query systems. The first of these is designed to recognize various types of ground vehicles registered by multiple sensors, in a sensor data independent way. The second query system is more specialized and is primarily used for visualization of high resolution terrain data models and for determination of terrain features of importance, e.g. for trafficability and generally to support the situation assessment process with necessary terrain knowledge. The situation assessment system is augmented with a framework for generation of TGP. An important issue in this framework is the ability to identify alternative interpretations of a situation, which should be done with the user in an active role.

The system proposed here is still in its infancy and a lot of work remains to be done; nevertheless the result so far is promising. The most substantial contributions to the system and its framework are currently demonstrated by the two query systems, although more work is required here as well, for instance integration of other sensor types.

References

[1]“Handbook of Multisensor data fusion”, D. L. Hall, J. S. Llinas (Eds.), CRC Press, Boca Raton, Florida, 2001.

[2]S.-K. Chang, E. Jungert, “*Query Languages for Multimedia Search*”, Principles of Visual Information Retrieval, M. S. Lew (ed.), Advances in Pattern Recognition, Springer Verlag, Berlin, 2001, pp 199-217.

[3]S. K. Chang, G. Costagliola and E. Jungert, “Querying Multimedia Data Sources and Databases”, Proceedings of the 3rd International Conference on Visual Information Systems (Visual’99), Amsterdam, The Netherlands, June 2-4, 1999.

[4]S.-K. Chang, G. Costagliola, E. Jungert, "Multi-Sensor Information Fusion by Query Refinement", Proceedings of the 5th International Conference on Visual Information Systems (VISUAL 2002), Hsin Chu, Taiwan, March 11-13, 2002.

[5]S. K. Chang and E. Jungert, “A Spatial/temporal query language for multiple data sources in a heterogeneous information system environment”, The International Journal of Cooperative Information Systems (IJCIS), vol. 7, Nos 2 & 3, 1998, pp 167-186.

[6]M. Elmqvist, E. Jungert et al., “Terrain Modelling and Analysis using Laser Scanner Data”, Proceedings of Conference on Land Surface Mapping and Characterization using Laser Altimetry, Annapolis, MD, USA, October 22-24, 2001, 219-226, published by Dept. of Geography, University of Maryland, MD, 2001.

[7]F. Lantz and E. Jungert, ”Dual Aspects of a Multi-Resolution Grid-Based Terrain Data Model with Supplementary Irregular Data Points”, Proceedings of the 3rd Int. Conf. on Information fusion, Paris, July 0-13, 2000.

[8]Mats Sjövall, “Object and Feature Recognition in a Digital Terrain Model”, Master Report, University of Linköping, LiTH-IDA-Ex-02/N

[9]T. Kirubarajan, et. al., “Ground Target Tracking with Variable Structure IMM Estimator”, IEEE Transactions on Aerospace and Electronics Systems, Vol. 36, No 1, January 2000.

[10]D. J. Salmond and N. J. Gordon, “Group Tracking with Limited Sensor Resolution and Finite Field of View”, Proceedings of SPIE, Vol 4048,Signal and Data Processing of Small Targets 2000, April 2000.

[11]M. Lodaya, R. Bottone, “Moving Target Tracking Using Multiple Sensors”, Proceedings of SPIE, Vol 4048,Signal and Data Processing of Small Targets 2000, April 2000.

[12]D. E. Goldberg, “Genetic Algorithms in Search, Optimization & Machine Learning”, Addison Wesley, 1989.

[13]C. A. C. Coello, “A comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques”, Knowledge and Information Systems. An International Journal, 1(3):269-308, August 1999.

[14]E. Zitzler, “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications”, PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[15]J.Teghem, “Multiobjective Combinatorial Optimization”, In PPSN/SAB Workshop on Multiobjective Problem Solving From Nature (MPSN), Paris, France, September 2000.

[16]B. Sareni and L. Krähenbühl, “Fitness Sharing and Niching Methods Revisited”, IEEE Transactions on Evolutionary Computation, Vol. 2, No 3, September 1998.

[17]M. Ehrgott and X. Gandibleux, “An Annotaded Bibliography of Multi-objective Combinatorial Optimization”, Technical Report 62/2000, Fachbereich Mathematik, Universitat Kaiserslautern, Germany, 2000.

[18]E. Falkenauer, “Genetic Algorithms and Grouping Problems”, New York: Wiley, 1998.

[19]S.Paradis, R. Breton and J. Roy, “Data Fusion in Support of Dynamic Human Decision Making”, Proceedings of the Second International Conference on Information Fusion (FUSION 1999), Las Vegas, USA, 1999.

[20]J. K. Johnsson and R. D. Chaney, “Recursive Composition Inference for Force Aggregation”, Proceedings of the Second International Conference on Information Fusion (FUSION 1999), Las Vegas, USA, 1999.

Utgivare
FOI - Totalförsvarets forskningsinstitut
Ledningssystem
Box 1165
581 11 Linköping

Rapportnummer, ISRN
FOI-R--1787--SE

Klassificering
Användarrapport

Forskningsområde
4. Ledning, informationsteknik och sensorer

Månad, år
September 2005

Projektnummer
E7089

Delområde
42 Spaningssensorer

Delområde 2

Författare/redaktör
Erland Jungert
Martin Folkesson
Jörgen Fransson
Tobias Horney
Fredrik Lantz
Karin Silvervarg

Projektledare
Erland Jungert

Godkänd av
Martin Rantzer

Uppdragsgivare/kundbeteckning
Försvarsmakten

Tekniskt och/eller vetenskapligt ansvarig
Erland Jungert

Rapportens titel
Ett frågebaserat beslutsstödssystem för nätverksbaserade ledningssystem

Sammanfattning
Olika typer av beslutsstödssystem för militära och krisrelaterade tillämpningar måste finnas tillgängliga för integration i nätverksbaserade ledningssystem. Dessa beslutsstöd måste i de flesta fall kunna utnyttjas för att samla in, analysera, fusionera, hantera, lagra och i slutligen visualisera information hämtade från en mängd olika typer av sensorer. Den på detta sätt insamlade och bearbetade informationen skall väsentligen ge användarna stöd i deras beslutsfattande. I moderna ledningssystem är dessa beslutsstöd oftast tjänsteorienterade. Speciella krav på generalitet, användbarhet och flexibilitet måste också ställas på dessa beslutsstöd. I detta arbete beskrivs ett frågebaserat och tjänsteorienterat informationssystem anpassat till nätverksbaserade ledningssystemtillämpningar, främst för markspanings-uppgifter med utnyttjande av sensordata. Andra väsentliga egenskaper i detta tjänstebaserade system är sensorinteroperabilitet och sensordataoberoende.

Nyckelord
frågespråk, situations analys, digital terräng model, framkomlighet,interoperabilitet, sensordataoberoende, tjänster

Övriga bibliografiska uppgifter

Språk Svenska

ISSN 1650-1942

Antal sidor: 120 s.

Distribution enligt missiv

Pris: Enligt prislista

Issuing organization
FOI – Swedish Defence Research Agency
Ledningssystem
Box 1165
581 11 Linköping

Report number, ISRN
FOI-R--1787--SE

Report type
User report

Programme Areas
4. C4ISTAR

Month year
September 2005

Project no.
E7089

Subcategories
42 Above water Surveillance, Target acquistion and Reconnaissance

Subcategories 2

Author/s (editor/s)
Erland Jungert
Martin Folkesson
Jörgen Fransson
Tobias Horney
Fredrik Lantz
Karin Silvervarg

Project manager
Erland Jungert

Approved by
Martin Rantzer

Sponsoring agency
Swedish Defence

Scientifically and technically responsible
Erland Jungert

Report title (In translation)
A query based decision support system for net-based command and control systems, services

Abstract
Different types of decision support systems for military as well as crisis management related applications must be available for integration in network based command and control systems. These decision support tools must in most cases be possible to use for collection, fusion, handling, storage and finally also for visualization of information from a large number of sensor data sources. The information collected and manipulated in any of these ways should primarily support the end-users in their decision making activities. In modern command and control systems this type of decision support tools will essentially be service oriented. Special requirements concerning generality, usability and flexibility must be determined for these services as well. In this work is a query based and service oriented information system adopted to network based command and control applications, primarily for ground target reconnaissance using sensor data input, described. Other important properties in this service based system are sensor interoperability and sensor data independence.

Keywords
Query language, Situation analysis, digital terrain model, driveability, interoperability, data independence

Further bibliographic information

Language Swedish

ISSN 1650-1942

Pages 120 p.

Price acc. to pricelist