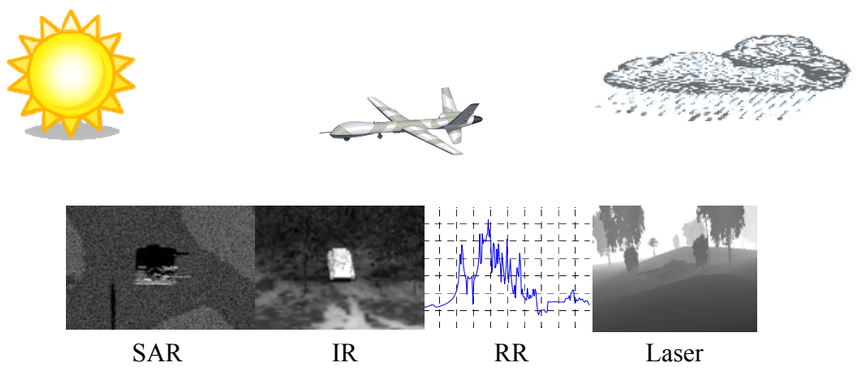
Simuleringsbaserade metoder för sensorvärdering

FREDRIK NÄSSTRÖM, MIKAEL KARLSSON, LEIF CARLSSON, TOMAS CHEVALIER,
PETER FOLLO, JOHAN HEDSTRÖM, NILS KARLSSON, ANDREAS PERSSON, AIN SUME



FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1000 anställda varav ungefär 800 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, bedömning av olika typer av hot, system för ledning och hantering av kriser, skydd mot och hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
Informationssystem
Box 1165
581 11 Linköping

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

www.foi.se

FOI-R--2629--SE Underlagsrapport **Informationssystem**
ISSN 1650-1942 December 2008

Fredrik Näsström, Mikael Karlsson, Leif Carlsson,
Tomas Chevalier, Peter Follo, Johan Hedström,
Nils Karlsson, Andreas Persson, Ain Sume

Simuleringsbaserade metoder för sensorvärdering

Titel	Simuleringsbaserade metoder för sensorvärdering
Title	Simulation based sensor system assessment

Rapportnr/Report no	FOI-R-2629--SE
Rapporttyp Report Type	Underlagsrapport Base data report
Sidor/Pages	51 p
Månad/Month	December
Utgivningsår/Year	2008
ISSN	ISSN 1650-1942
Kund/Customer	FM
Forskningsområde Programme area	4. Sensorer och signaturanpassning 4. Sensors and Low Observables
Delområde Subcategory	42 Sensorer 42 Above surface Surveillance, Target acquisition and Reconnaissance
Projektnr/Project no	E3097
Godkänd av/Approved by	Ola Kärvell

FOI, Totalförsvarets Forskningsinstitut
Avdelningen för Informationssystem
Box 1165
581 11 Linköping

FOI, Swedish Defence Research Agency
Information Systems
P.O.Box 1165
SE-581 11 Linköping

Sammanfattning

Projektet SIMSENS (Simuleringsbaserade metoder för sensorvärdering) har i uppgift att utveckla en gemensam simuleringsmiljö för värdering av sensorsystem i olika miljöer, väderförhållanden och tidpunkter. Simuleringsverktyget som vidareutvecklas för detta är MSSLab (MultiSensorSimuleringsLab).

För att en korrekt sensorvärdering skall erhållas med simuleringsbaserade metoder krävs noggranna simuleringsmodeller. I denna rapport beskrivs utvecklingen och aktuell status för simuleringsmodellerna i MSSLab.

Simuleringar i olika väderfall har gjorts för sensorerna IR och radar. Väderfallen som använts vid simuleringarna var vackert väder, regn, moln och snöfall. För IR och visuella simuleringar har beräkningsprogrammet CameoSim använts. För att beräkna de atmosfärsdata som används i CameoSim har transmissionsmodellen MODTRAN använts. Med MODTRAN kan atmosfärens egenskaper för olika väderfall beräknas.

För radarsimuleringar används beräkningsprogrammet SE-RAY-EM. Detta program har nu kompletterats med funktioner för att ta hänsyn till både dämpning och spridning för de olika väderfallen.

För att förbättra Norrköpingsmodellen har metoder utvecklats för att automatiskt extrahera och texturera byggnader utgående från data om byggnadernas byggytor och ortorektifierade flygfoton. Dessa metoder har sedan använts för att automatiskt rekonstruera byggnader i Norrköpingsmodellen.

För att göra det möjligt att simulera sensordata med animerade människor har HLAS, High Level Animation System, tagits fram. Med HLAS underlättas integrationen av animerade sekvenser för olika typer av simuleringsramverk. Med HLAS är det nu möjligt att styra animeringssekvenser på en högre abstraktionsnivå.

Nyckelord: simulering, värdering, IR, laser, radar, MSSLab

Summary

The task for the SIMSENS' project (Simulation based sensor system assessment) is to develop a modular sensor system simulation program that can simulate sensor systems in various environments, weather conditions and time periods. The simulation tool that is further developed is the MSSLab (MultiSensorSimulationLab).

To get a correct sensor assessment with simulation based methods you need accurate simulation models. In this report the development and current status of these models in MSSLab are described.

Simulations are made for the IR- and radar sensors in different weather cases. Used cases are fine weather, rain, clouds and snow fall. IR and visual simulations are made with CameoSim, a synthetic imagery software tool. The atmospheric data used in CameoSim are calculated in MODTRAN, in which the atmospheric transmission for the different weather cases can be calculated.

Radar simulations are made with the software tool SE-RAY-EM. Complementary functions are added to this software to take care of attenuation and scattering for the different weather cases.

To improve the Norrköping-model, methods are developed to automatically extract and texturize buildings emanating from data about building surfaces and unrectified aerial pictures. These methods are then used to automatically reconstruct buildings in the Norrköping-model.

To make it possible to simulate sensor data with animated persons, HLAS (High Level Animation System) has been developed. With HLAS the integration of animated sequences in different simulation frameworks is made easier. With HLAS it is now possible to handle animation sequences on a higher abstraction level.

Keywords: simulation, assessment, IR, laser, radar, MSSLab

Innehållsförteckning

1 Inledning 7

2 Terrängmodeller 9

2.1 Kvarnmodellen 9

2.2 Norrköpingsmodellen 11

3 Objektmodeller och animering av människor 13

3.1 Objektmodeller 13

3.1.1 Beräkning av IR-signaturen för objekt 13

3.2 Animering av människor 17

4 Sensorsimuleringar 21

4.1 Radar 21

4.2 IR 24

4.3 SceneServer 28

4.4 Laser 28

5 Sensordataanalys 30

5.1 Detektion 30

5.2 Målföljning 33

5.3 Återigenkänning 35

5.4 Ruttplanering och sensorstyrning 37

6 Nätverksdistribuerad simulering med HLA 39

6.1 Introduktion till FOM 39

6.1.1 HLA-objekt 40
6.1.2 Interaktioner 40
6.1.3 Datatyper 41

6.2 Tillägg till RPR-FOM:en för distribuerad sensorsimulering 42

6.2.1 Tillagda sensorobjekt 42
6.2.2 Tillagda sensorinteraktioner 44
6.2.3 Datatyper 46
6.2.4 JAVA- och C++-implementering av FOM-tillägg 46

6.3 Fortsatt arbete 46

7 Resultat 48

7.1 Status på utvecklingen av simuleringsbaserade metoder för sensorvärdering 48

7.2 Fortsatt arbete 49

8 Referenser 50

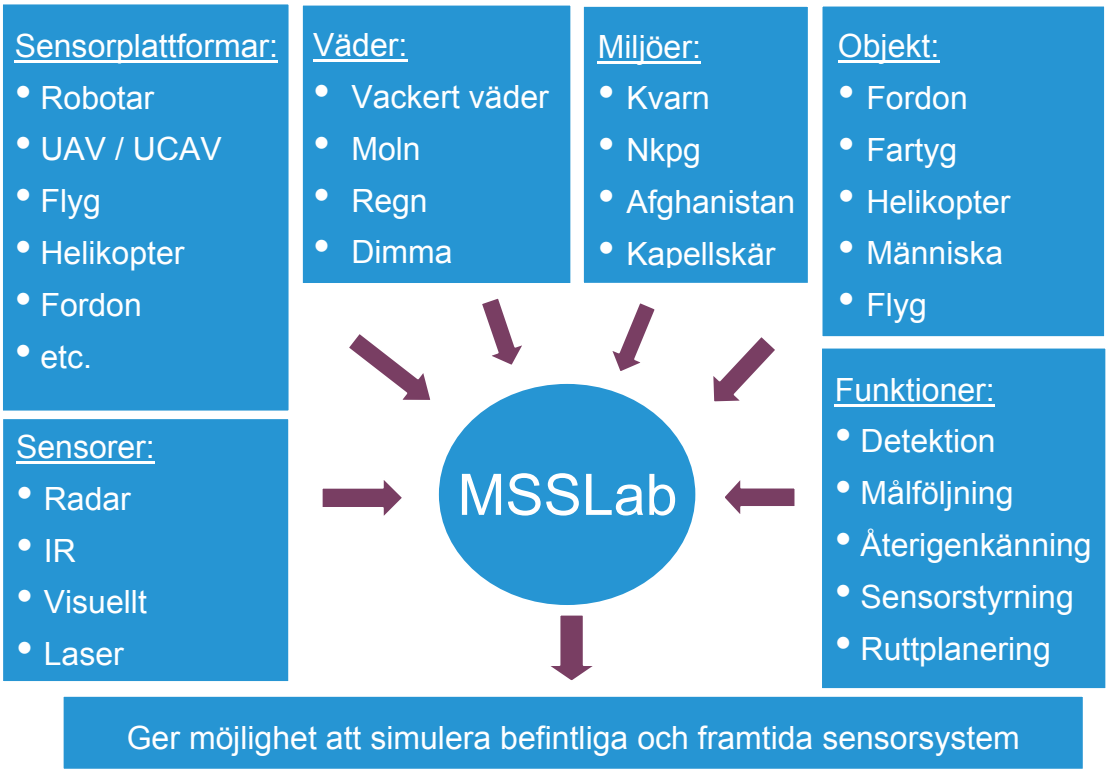
1 Inledning

Vid en värdering av ett sensorsystem krävs tillförlitlig information om sensorprestanda för den aktuella uppgiften. Spelkort eller tekniska specifikationer finns ofta för befintliga system, men information om sensorprestanda för många internationella och framtida sensorsystem, och sensorsystem i nya miljöer är ofta otillräcklig.

Ett vanligt sätt att skatta prestanda hos ett system är med hjälp av simuleringsbaserade metoder. För att få hög tillförlitlighet med simuleringsbaserade metoder krävs god kännedom om det aktuella systemet samt validerade modeller. Fördelen med simuleringsbaserade metoder är att när väl systemet är modellerat och omgivningsmodellerna finns, så kan simuleringar göras för en mängd olika fall till en rimlig kostnad och på relativt kort tid.

På FOI Informationssystem utvecklas en gemensam simuleringsmiljö för värdering av sensorsystem. Denna simuleringsmiljö ska även kunna bidra med information om sensorprestanda för alla nivåer från teknisk till taktisk värdering.

Simuleringsverktyget som vidareutvecklas för detta är MSSLab (MultiSensorSimuleringsLab), vilken är utvecklad vid FOI Informationssystem. MSSLab är en simuleringsmiljö där avancerade sensorsystem kan simuleras i olika miljöer och vid olika väderförhållanden och tidpunkter. Figur 1 visar olika beräkningsenheter som kan integreras vid en simulering med MSSLab.



Figur 1. Sammanställning av delsystem som kan integreras vid en simulering med MSSLab.

Simuleringsverktyget möjliggör simulering av dynamiska scenarier med rörliga plattformar, sensorbärare, målobjekt etc. Högkvalitativa sensorsimuleringar kan göras för IR, visuellt, laser och radar. Dessa sensorsystem kan simuleras både enskilt, som samverkande sensorsystem och som multisensorsystem. Avancerade algoritmer för objektdetektion, följning, återigenkänning, sensorstyrning och ruttplanering håller även på att integreras i simuleringsverktyget.

En svårighet vid simulering av sensorsystem är att sensorprestanda påverkas av många scenariovariabler. MSSLab har redan idag mycket bra funktionalitet, men mycket arbete återstår för att få hög tillförlitlighet vid värdering av sensorsystem i nya miljöer. Exempel på viktiga scenariovariabler som ofta påverkar sensorprestanda är:

- Målobjekt: konfiguration, signatur, kamouflage, skylda eller ej, mål på marsch eller stillastående etc.
- Terräng: bebyggelse, öppen, småbruten, betäckt, öken, djungel etc. Terrängen kan dessutom vara platt, kuperad eller bergig, samt ha inslag av sjöar och vattendrag.
- Vegetation: fördelningar av träd och buskar och dess årstidsförändringar.
- Väder/atmosfärstransmission: plats på jorden, årstid, dygnstid, lufttyp, luftfuktighet, temperatur, molnighet, molnbas, olika typer av nederbörd etc.
- Ljusförhållanden: tid på året och dygnet, aktuellt väder och plats på jorden.
- Datainsamling: observationstid, avstånd, höjd, aspektvinkelförändring på målet under observationen, hastighet på sensorplattformen etc.
- Hot och störning mot sensor och sensorbärare.
- Handlingsregler för uppdraget (ROE - Rules Of Engagement)

Målsättningen med denna rapport är att beskriva aktuell status för de olika delarna i MSSLab, samt beskriva arbetet som bedrivits i projektet Simuleringsbaserade metoder för sensorvärdering under 2008.

2 Terrängmodeller

För att kunna värdera ett sensorsystem med simuleringsbaserade metoder krävs realistiska och detaljerade omvärldsmodeller. En vanlig uppdelning av omvärldsmodeller är geospecifika resp. geotypiska modeller, där geospecifika modeller avbildar verkliga miljöer medan geotypiska miljöer är virtuella men realistiska.

På FOI Informationssystem har metoder utvecklats för att ur data från flygburna lasermätsystem automatiskt extrahera modeller för markytan, extrahera information om position, höjd, vidd och trädslag hos vegetation, samt rekonstruera byggnader [1, 2]. Den extraherade information integreras sedan tillsammans med data från andra källor i kommersiella verktyg till geospecifika omvärldsmodeller med hög geometrisk upplösning och detaljeringsgrad.

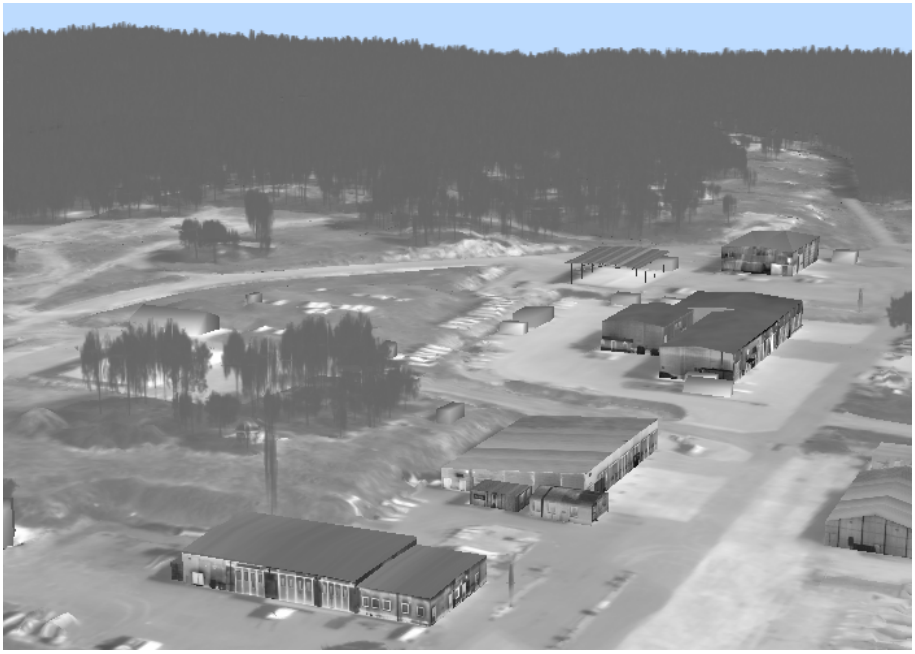
En stor fördel med geospecifika miljöer är att de representerar en verklig plats. Detta gör det möjligt att verifiera simulerade resultat med verkliga fältmätningar. De två mest utvecklade omvärldsmodellerna vid FOI Informationssystem är Kvarnmodellen och Norrköpingsmodellen, vilka kommer att beskrivas i detta avsnitt.

2.1 Kvarnmodellen

Kvarnmodellen representerar en modell av Prästtomta skjutfält vid Kvarn som ligger ca 3 mil nordväst om Linköping. Detta område har mätts upp med ett flygburet lasersystem och från dessa data har omgivningsmodellen skapats. Kvarnmodellen innehåller en högupplöst markyta, automatiskt rekonstruerade byggnader och modeller av enskilda träd inplacerade med korrekta dimensioner på korrekta positioner. Den rumsliga upplösning i modellen är ca 5-10 punkter/m^2 och med en koordinatnoggrannhet på 0.1 m (i rikets nät). Detta kan jämföras med Lantmäteriverkets standardprodukt med 50 meter mellan höjdvärdena.

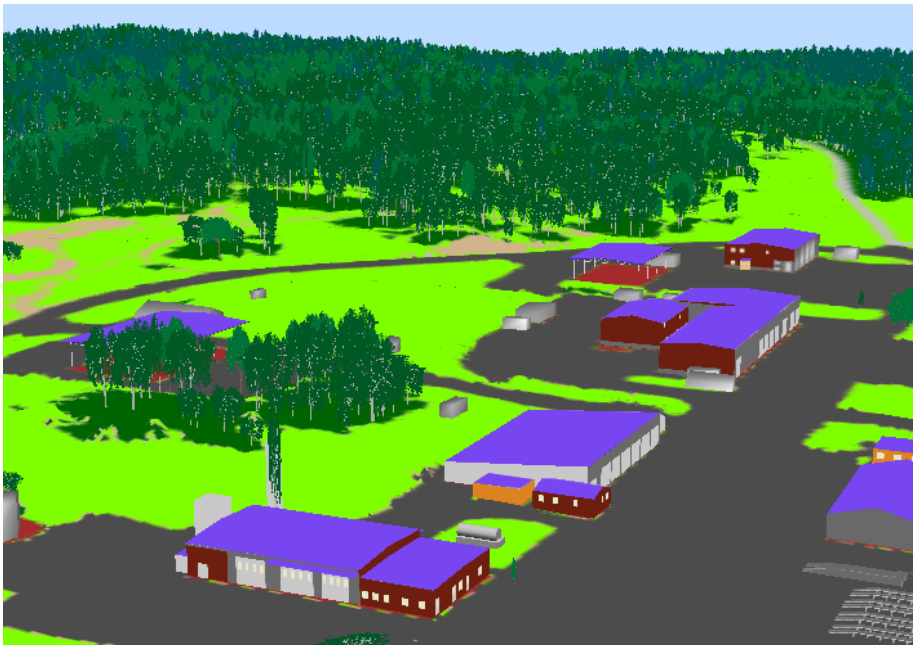
Figur 2. Bilden visar en del av Kvarnmodellen med uppmätt visuell textur.

Högupplöst visuellt och IR-data finns även insamlat över det laseruppmätta området. Dessa data har sedan använts som textur på den tredimensionella omgivningsmodellen. Detta ger en omvärldsmodell som ger en bra bild av hur terrängen såg ut vid det aktuella insamlingstillfället, vilket visas i figur 2 och figur 3.



Figur 3. Bilden visar en del av Kvarnmodellen med uppmätt IR-textur.

Det finns även en materialklassad modell som används vid högkvalitativa sensorsimuleringar av IR- och radarsystem. Terrängmodellen som visas i figur 4 är klassificerad med följande material; gräs, väg, undervegetation, lera, berg och vatten. De tre omgivningsmodellerna är alla geografiskt och geometriskt korrelerade.



Figur 4. Bilden visar en del av Kvarnmodellen med materialklassad textur. Färgerna i bilden motsvarar material / terrängtyp.

För varje träd i det uppmätta området har information om position, höjd, vidd och trädslag extraherats. Modeller av enskilda träd har sedan placerats in i terrängmodellen med korrekta dimensioner och på korrekta positioner. Använda trädtyper är tall, gran och björk. Undervegetation och gräs saknas i nuvarande modell. Ett problem med omvärldsmodeller med noggrann beskrivning av vegetation är att dessa blir mycket stora och kräver mycket minne. Förenklade trädmodeller har därför använts i modellen.

2.2 Norrköpingsmodellen

Norrköpingsmodellen består av en högupplöst markyta, modeller av enskilda träd och över 25 000 byggnader. I figur 5 visas en del av modellen som bygger på data som har extraherats från ett lasersystem och en visuell kamera. Den rumsliga upplösningen i modellen är på 5-10 punkter/m^2 och med en koordinatnoggrannhet på 0.1 m (i rikets nät).

Under hösten 2008 utvecklades i samarbete med projektet 'Basblock: Omvärldsmodellering 2007-2008' metoder för att automatiskt extrahera och texturera byggnader utgående från kommunal data om byggnadernas byggytor och ortorektifierade flygfoton. Information om väggarna saknas så istället har en uppsättning av generiska väggtexturer använts [3].

Figur 5. Del av Norrköpingsmodellen.

För taktextur hos modellen kan ortorektifierat flygfoto användas, men för att inte få ohanterliga mängder texturdata har husets takfärg skattats och används istället. Detta möjliggör realistisk om än bara delvis korrekt visualisering av mycket stora områden. I figur 6 visas en del av Norrköping med generiska väggtexturer och färgsatt tak där färgen är skattad från flygfoto.

Figur 6. Del av Norrköpingsmodellen. Husen är extraherade ur kommunens grunddata varefter generiska texturer applicerats på väggarna. Takfärgen har skattats från flygfoto.

3 Objektmodeller och animering av människor

3.1 Objektmodeller

Vid värdering av sensorsystem med simuleringsbaserade metoder är det mycket viktigt att modellerna för såväl bakgrund som objekt är av hög kvalité. På FOI finns mycket god kunskap om modellering av signaturen hos olika objekt. Några exempel på objekt visas i figur 7. Objekten som används har generellt mycket hög detaljrikedom, vilket är viktigt för att noggranna signaturberäkningar ska kunna göras.

Figur 7. Exempel på några tillgängliga objektmodeller

För radar är uppdelning i delobjekt och meshning av dessa inte avgörande för simuleringsresultatet, varför modeller i de flesta fall kan införskaffas via kommersiella kanaler. De viktigaste kraven på dessa modeller är att de är geometriskt någorlunda korrekta och att de är tillräckligt detaljerade.

3.1.1 Beräkning av IR-signaturen för objekt

I projektet SIMSENS har IR-signaturen för olika objekt beräknats under skiftande drifts- och vädertillstånd. Fördelen med att beräkna IR-signaturen är att man kan analysera signaturen hos ett fordon med hänsyn till inre värmekällor, väder, vinddrag, friktionsvärme, tidpunkt på dygnet m.m. Analyser av uppvärmnings- och nedkylningsförlopp kan analyseras. Inre värmekällor kan till exempel vara motor, kylare och avgassystem. I figur 8 visas några resultat från signaturberäkningar. Exemplen är tagna ur transienta dygnsberäkningar som är beräknade med 5 minuters intervall för full termisk historik.

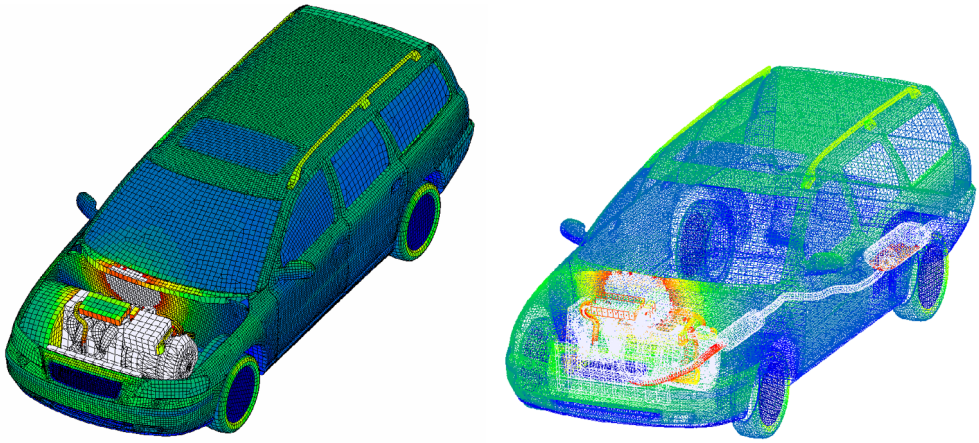
Figur 8. Exempel på beräknade IR-signaturer. Från vänster till höger och uppifrån och ner: Volvo V70, BTR70, MTLB, T72 och HKP10 är alla beräknade av FOI. Helikoptern AH-64 Apache i figuren längst ner till höger är beräknad av ©ThermoAnalytics Inc.

För att kunna göra en noggrann IR-signaturberäkning krävs en noggrann beskrivning av det aktuella objektet. Vanligtvis representeras de geometriska ytorna med en s.k. mesh, vilket är ett antal noder som kopplats samman. Den vanligaste polygontypen är triangel, men även kvadrat, pentagon etc. kan användas. Generellt gäller att högre upplösning på ytmeshen ger bättre beräkningsnoggrannhet. Vid en viss gräns konvergerar dock förbättringarna och man vinner väldigt lite i noggrannhet då upplösningen ökas ytterligare, utan resultatet blir istället onödigt långa beräkningstider. Flera av de framtagna objekten har olika ytmeshningsdensitet på olika delar av objekten. Detta för att både erhålla hög noggrannhet för viktiga detaljer och rimliga beräkningstider.

För yt- och volymsmeshning har ANSYS ICEM CFD 11.0 använts. Med detta program är möjligheterna mycket goda att styra mesh-genereringen. Olika elementtyper och storlekar kan användas för olika delar av objektet. Det finns även möjlighet att justera den ursprungliga geometrin som oftast är framtagen för att användas inom spelindustrin. I sådana modeller saknas inre värmekällor och de eventuella inre strukturer som finns stämmer oftast inte överens med verkligheten utan olika delar går rakt igenom varandra m.m. Allt detta måste ordnas innan en lämplig beräkningsmesh kan skapas.

För strömningsberäkningar (CFD - Computational Fluid Dynamics) krävs volymsmeshning och det finns möjligheter till bl.a. hybridmeshning med hexa och tetra element. För väggfunktionen som krävs för de flesta turbulensmodeller kan ett antal prismalager nära ytan byggas upp.

I figur 9 visas beräkningsmeshen för en modell av en Volvo V70. De inre värmekällorna i fordonet består av en motor och ett avgassystem. Fälgarna är plana och heltäckande och är i massivt blankt aluminium, vilket ger låga radiansnivåer. Railsen är modellerade med svart färg liksom vissa andra plastlister, vilket ger högre radiansnivåer under solbelysning.



Figur 9. I den vänstra figuren visas beräkningsmeshen för en Volvo V70 och i den högra bilden visas en signaturberäkning där värmekällorna är en motor och ett avgassystem.

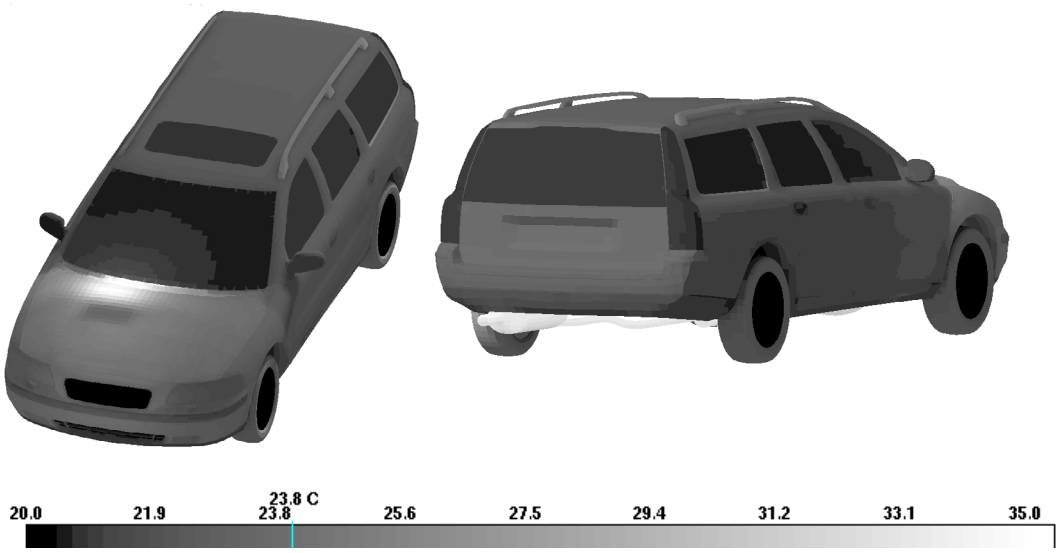
För noggranna temperaturberäkningar och radiansberäkningar av objekt har programmet RadThermIR 9.2 använts. Detta är ett kommersiellt program från ThermoAnalytics Inc. Med detta program kan värmeekvationerna beräknas i tre dimensioner och interna värmekällor är då inkluderade. Programmet löser således värmekevationerna för strålning, konvektion och ledning. Indata är bl.a. en beräkningsmesh, väderfiler, materialdata och reflektans. Man kan också lägga in driftsprofiler med t.ex. data från GPS med positioner och hastigheter för fordonet. Även markstigning m.m. kan läggas in liksom att fordonet stannar och motorn svalnar. Typiska signaturfenomen som att luftintag är kalla under drift, men blir varma då motorn stängs av kan simuleras. Temperaturberoende material och reflektansegenskaper kan också ges som indata. Vid behov kan BRDF (Bidirectional reflectance distribution function) renderingar göras.

Beroende på avsikten med modelleringen kan således allt från detaljstudier av motorrum, till komfort för människor i fordonet till objektets signatur i en hög-

upplöst terräng studeras. En modul för mänsklig värmekomfort kan även köpas till.

Konvektion beräknas förenklat i 1-D. För att förbättra kvaliteten på IR-signaturen kan man importera CFD till RadThermIR och därmed få värme-övergångskoefficienter på elementbasis. Under året har strömningsberäknigs-koden CFD++ köpts in och ingår i IRSSLab. Ett exempel på när det krävs en koppling till CFD är för att modellera avgasplymer (se helikoptern i figur 8) och dess konvektiva effekter på skrovet, samt effekten av fartvinden och vind-draget från rotorn. Behovet av att koppla RadThermIR och CFD beräkningar har tidigare visats i bl.a. [4].

I figur 10 visas en beräkning där en Volvo V70 är simulerad för vädret som var den 2 juni 2003 i Kvarn (ca 3 mil nordväst om Linköping). Simuleringarna är genomförda under ett helt dygn, med full väderhistorik, dvs. transienta simule-ringar. Bilden är i fysikalisk temperatur men finns även simulerad för våg-längdsbanden 3-5µm, 8-12 µm samt 8-9 µm. Modellen består av ett avgassy-stem och en förenklad motor som bygger på en hybridmodell från Toyota Pri-us. En torpedvägg har satts in i fordonet men vissa värmesköldar kring grenrör saknas, vilket ger en alltför kraftig strålningseffekt in i kupén genom torped-väggen.



Figur 10. Signaturberäkning av en Volvo V70 med hastigheten 70 km/h.

I tabell 1 finns en sammanställning av vilka inre värmekällor som är modelle-rade för några objekt.

Tabell 1. Aktuell status på modelleringen för några olika objekt.

Objekt	**Beskrivning**	**Inre geometri (värmekällor)**
Volvo V70	Personbil - civilt fordon	Motor och avgassystem.
BTR70	Pansarskyttefordon (hjulfordon)	Motor och hjullager.
T72	Stridsvagn	Motor, kylare luftintag, utblås, avgassystem.
PBV401 - MTLB	Pansarskyttefordon	Motor, luftintag, utblås, kylare, avgassystem
BMP1	Pansarskyttefordon	Hjullager.
M1A2	Stridsvagn	Hjullager.
HKP10.	Helikopter	

Under 2008 har ett modelleringsarbete av helikopter 10 (Super Puma) påbörjats. I det inledande arbetet har en ytmesh och partsuppdelning genomförts. En bild på ytmeshen visas i figur 11.

Figur 11. Ytmeshad och partsuppdelad Hkp10.

Under 2009 kommer arbetet att fortsätta för att göra det möjligt att simulera en flygande helikopter. Rotorns och fartvindens påverkan på avgasutblås och skrov kommer då att modelleras. För att få med dessa effekter kommer strömningsberäkningar att vara nödvändiga.

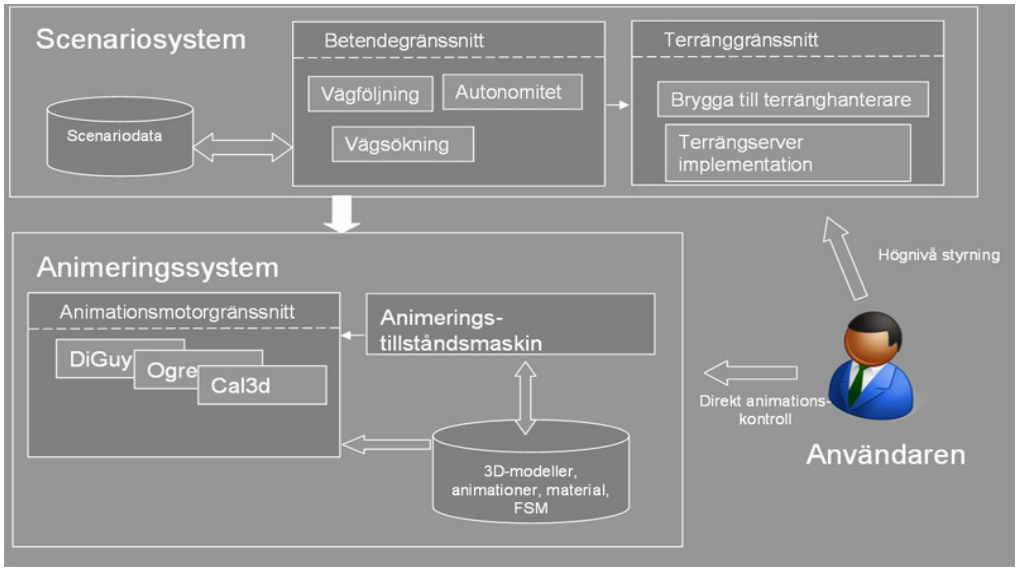
3.2 Animering av människor

Simulering av sensordata med animerade människor är en verksamhet som påbörjades 2008. För att göra detta möjligt har HLAS, High Level Animation System, tagits fram inom SIMSENS-projektet. Med HLAS underlättas integrationen av animerade sekvenser för olika typer av simuleringsramverk. För att skapa och visualisera skelettbaserade animationer finns det flera verktyg, men utbudet är inte lika stort då man vill kunna styra dessa animeringssekvenser på

en högre abstraktionsnivå. Ett exempel på ett sådant fall kan vara att en användare vill förflytta en karaktär från en punkt till en annan utan att behöva ta hänsyn till hur terrängen ser ut. För att en sådan förflyttning ska utföras på ett trovärdigt sätt behövs ett stöd för vilka animationer som ska spelas. I figur 12 visas några modeller som använts för animering av människor.

Figur 12. Modeller som använts vid animering av människor.

Grundtanken med designen av HLAS är att skapa ett plattforms- och renderingsoberoende system som på ett enkelt sätt ska kunna utvidgas, integreras och anpassas för olika typer av simuleringar. HLAS kan delas in i två moduler, scenariosystemet samt animeringssystemet. Animeringssystemet kan i sin tur brytas ner i två delar, en tillståndsmaskinmodul och animationsmotormodul, vilket visas i figur 13.

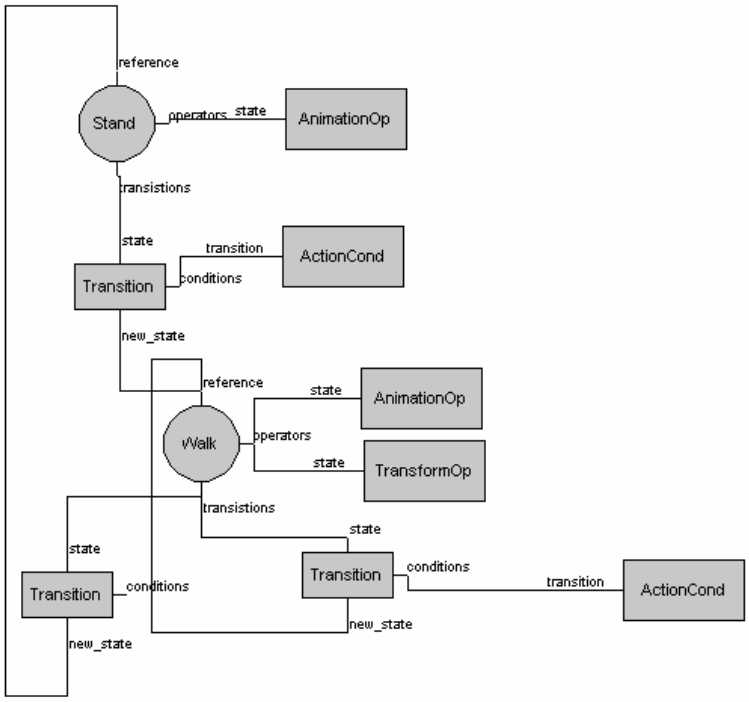


Figur 13. Övergripande modulindelning.

Animationsmotorn är den del som hanterar inläsning, uppspelning och rendering av animationer. Då det redan existerar några system som hanterar just detta har denna del abstraherats via ett gränssnitt så att användaren har möjlighet att integrera valfri mjukvara på denna nivå. Referensimplementeringen av detta gränssnitt använder sig av animeringsmotorn Cal3D [5]. Eftersom Cal3D i sig är ett plattforms- och renderingsoberoende system lämpar det sig väl som refe-

rensimplementering. Den renderingsimplementation som i dagsläget finns bygger direkt på OpenGL [6], men integration av andra system som t.ex. OpenSceneGraph [7] sker relativt enkelt genom att ett nytt renderingsplugin skapas.

En tillståndsmaskin i HLAS hanterar de animationer och förflyttningar en karaktär kan utföra. Tillståndsövergångar utförs då givna övergångsvillkor är uppfyllda. Exempel på ett sådant villkor kan vara ett kommando från användaren eller att karaktären uppnått en given hastighet. Då endast ett tillstånd kan vara aktivt vid samma tillfälle rekommenderas användaren att skapa separata tillståndsmaskiner för olika delar av karaktären. På detta sätt kan antalet tillstånd hållas ner till en hanterbar mängd. En tillståndsmaskin är konfigurerbar antingen via en xml-fil eller i ett grafiskt verktyg, vilket visas i figur 14.



Figur 14. Exempel på en tillståndsmaskin med två tillstånd, Stand och Walk. Dialogen visar parametrar för en animations operator.

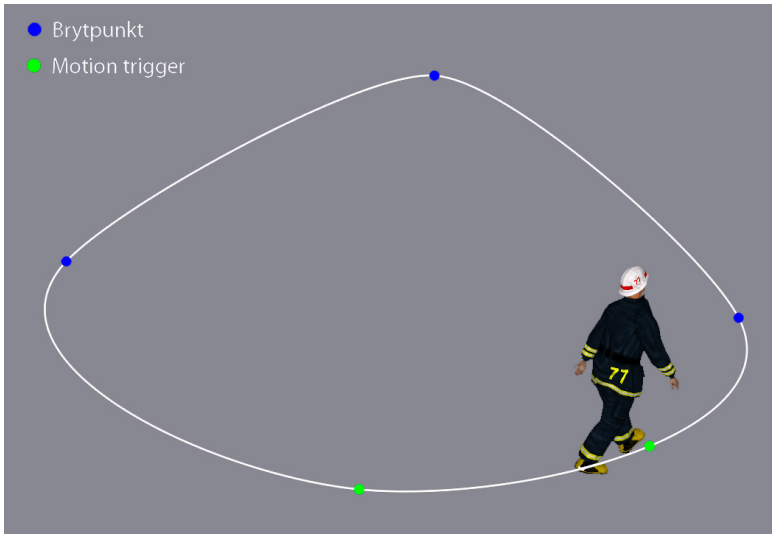
För att få en flexibel tillståndsmaskin har ett tillstånd brutits ner i två delar, operators och transitions. En operator är något som påverkar karaktären t.ex. att en animation spelas eller att en förflyttning föreslås. Ett tillstånd kan ha godtyckligt många animationsoperatorer för att på så sätt skapa en önskad rörelse. Animationerna kommer vanligtvis från ett motion capture system, men kan även vara skapad i ett animeringsverktyg som t.ex. Motion Builder eller 3D-Studio Max.

En transition är en möjlig övergång till ett nytt tillstånd. En transition består av olika conditions vilka tillsammans avgör om en tillståndsövergång är möjlig. Existerar inga conditions för en transition är det en ovillkorlig tillståndsövergång. Om flera transitions är uppfyllda i samma tillstånd är det den först listade övergång som väljs, processordningen är med andra ord sekventiell. En tänkbar vidareutveckling är att användaren har möjlighet att ändra processordningen, t.ex. kan en slumpmässig processordning vara av intresse.

Innan ett scenario kan skapas i HLAS måste varje deltagande karaktär ha en fungerande tillståndsmaskin som uppfyller önskade förflyttningar och rörelser. Användaren skapar ett scenario antingen via HLAS API (Application Programming Interface) eller genom att ladda in ett färdigt scenario från en xml-fil. Därefter kan man via API:et ändra delar i scenariot för att sedan spara ner det till fil igen.

Beteendet som finns implementerat i HLAS är en enkel brytpunktsnavigering, vilket visas i figur 15. Denna navigering bygger på att användaren först anger ett antal brytpunkter därefter några rörelsekommandon, s.k. motion triggers. En motion trigger är aktiv och förblir aktiv så länge tre parametrar är inom givna intervall. Dessa parametrar är tid, position och antalet skickade kommandon. När dessa tre villkor uppfylls skickas ett önskat kommando till karaktären som sedan tolkas av tillståndsmaskinen som i sin tur föreslår en förflyttning.

Brytpunktsnavigeringen utför sedan själva förflyttningen genom att fråga karaktären vilken hastighet den vill röra sig med och räknar därefter ut den totala tillryggalagda sträckan längs banan och placerar karaktären på denna position. Eventuella rotationer som tillståndsmaskinen föreslår tas inte hänsyn till av brytpunktsnavigeringen utan karaktären antas hela tiden vara roterad längs brytpunktsbanans riktning.



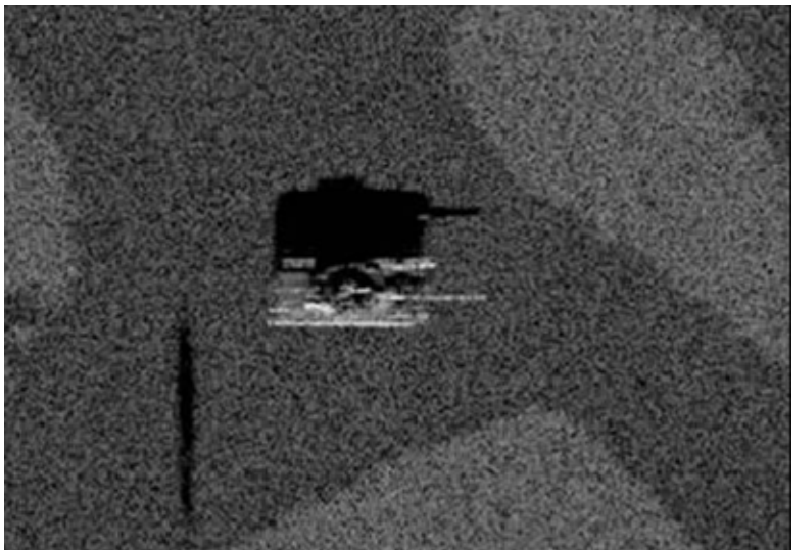
Figur 15. Scenario med brytpunktsnavigering.

4 Sensorsimuleringar

4.1 Radar

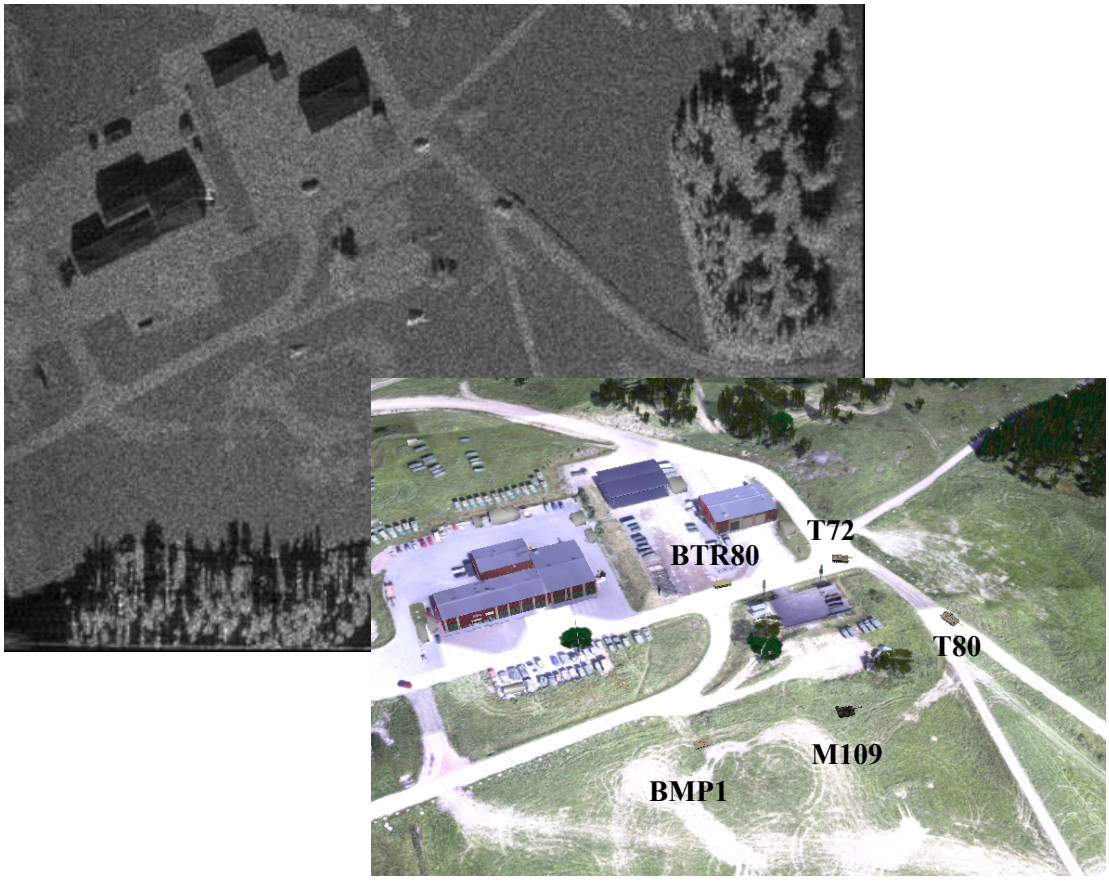
För simulering av radarsystem används den kommersiella programvaran SE-RAY-EM. Detta är en programvara som det franska företaget OKTAL-SE [8] har utvecklat. Programmet beräknar elektromagnetisk spridning och kan till exempel användas för att simulera olika radarsystem eller för att estimera radarmålarea hos objekt. Den primära funktionen för SE-RAY-EM är att beräkna elektromagnetisk spridning från objekt med geometrisk och fysikalisk optik som beräkningsmodell. För att kunna göra naturtrogna simuleringar behöver samtliga objekt och terrängbakgrunder vara materialklassade. Vid simuleringen ansätter radarsimuleringsverktyget materialparametrar till respektive material.

I figur 16 visas en bild på en T72-stridsvagn från ett simulerat SAR-system (Syntetisk AperturRadar). Upplösningen hos SAR-systemet är 0.1×0.1 meter. I bilden syns tydligt hur skuggan korrekt projiceras på den underliggande terrängen.



Figur 16. Beräknad SAR-bild av T72 med 0.1 meters upplösning.

I figur 17 visas en beräknad SAR-bild över en del av skjutfältet i Kvarn med en upplösning på 0.5×0.5 meter. I scenen står fem olika stridsfordon utplacerade, vilket visas i den visuella bilden.

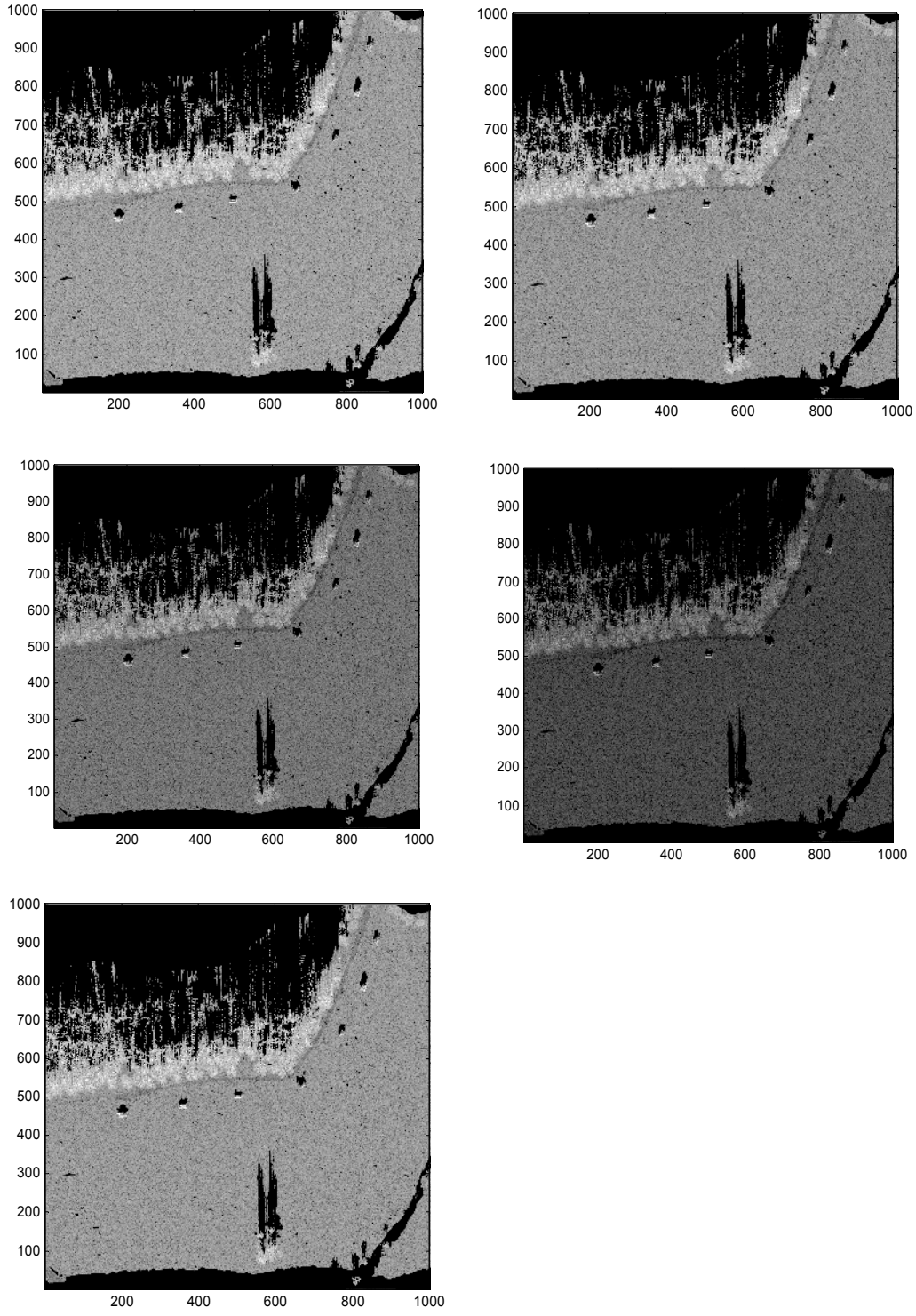


Figur 17. Beräknad SAR-bild och visuell bild med fem stridsfordon.

För att studera vädrets påverkan på radarsystem har olika atmosfärsmodeller tagits fram. De framtagna atmosfärsmodellerna tar hänsyn både till dämpning och spridning för radarvågor från ca 1 till 100 GHz, för väderfallen vackert väder, regn, moln/dimma och snöfall.

För att studera vädrets påverkan på radarsystem har ett SAR-system simulerats i olika väderfall. Det simulerade SAR-systemet arbetar på frekvensen 35 GHz och har en upplösning på 0.3 × 0.3m. SAR-data är simulerad för flyghöjden 1 km och på avståndet 4.1 km från målområdet. I scenariot som visas i figur 18 finns sju fordon utplacerade framför en skogsridå i Kvarn-modellen.

Övre vänstra bilden visar en SAR-registrering upptagen vid vackert väder. I den övre högra bilden är absorption i moln tillagt och man kan då se en svag försämring av kontrasten i SAR-bilden. I bilderna i mitten är dämpning och spridning från regn utmed hela sträckan tillagt mellan mål och sensor. Bilden till vänster visar regnintensitet 5 mm/tim och bilden till höger intensitet 10 mm/tim. För regnfallen fås en måttlig försämring av radarbilden i form av minskad kontrast. Den nedersta bilden har snöfall med ekvivalent regn-intensitet 1 mm/tim pålagts utmed hela vägsträckan från radarn till mål-området; någon molninverkan har inte antagits längs denna sträcka. På detta korta avstånd ger snöfallet endast mycket svag försämring av SAR-bilden. En mer detaljerad beskrivning av vädersimuleringar med SE-RAY-EM finns i [9].



Figur 18. Vädrets påverkan på radarsensorer. De olika väderfallen är från vänster till höger och uppifrån och ner: vackert väder, moln, regn (intensiteten 5 mm/tim), regn (intensiteten 10 mm/tim) och snöfall (ekvivalent regnintensitet 1 mm/tim).

En sammanfattande kommentar rörande väderpåverkan enligt dessa simuleringar är att på de relativt korta avstånd som studerats, är inverkan måttlig på en SAR-sensor för frekvensen 35 GHz. En grov tumregel för storleksförhållan-

dena hos utbredningsförlusterna i atmosfären för radarvågen är i minskande storleksordning: smältande snö, regn, moln och dimma, samt slutligen torr snö.

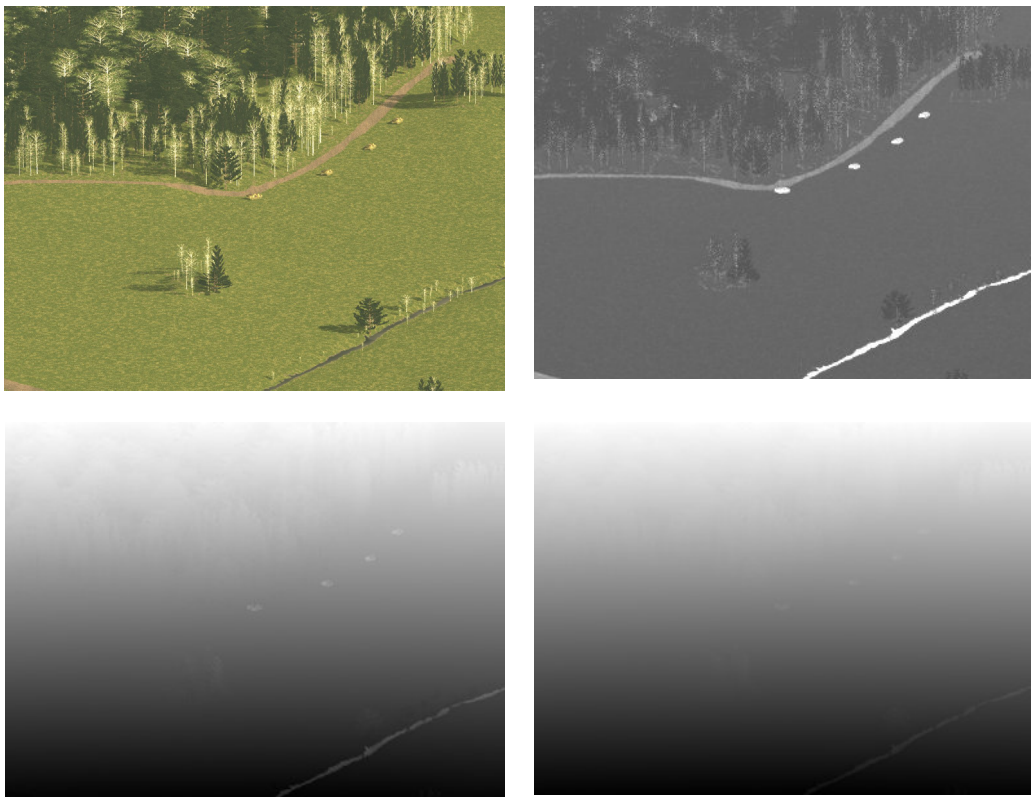
Modellerna kan numera ses som noggranna och är inte gränssättande för hur noga man kan beräkna utbredningsförlusterna i ett specifikt fall. Osäkerheten bestäms oftast till övervägande delen av att man inte har en noggrann kännedom av atmosfärens tillstånd utmed utbredningssträckan (t.ex. regncellers exakta utsträckning, intensitetsvariationer hos nederbörden och komplicerad dynamik som leder till inhomogeniteter). Beräkningarna ger därför oftast typiska värden, snarare än att vara giltiga för specifika fall.

4.2 IR

För noggranna simuleringar av optiska sensorer har CameoSim (Camouflage Electro Optic Simulation) använts. Detta är ett kommersiellt program som är framtaget av Insys Ltd i England och har använts på FOI sedan 2003. CameoSim kan simulera sensorer inom det optiska våglängdsbandet 0.4-14 μm och scenarierna kan vara både statiska och dynamiska. I CameoSim finns fysikaliskt baserade modeller för väder, termodynamik för beräkning av fysikalisk yttemperatur, avancerade renderingsalgoritmer för modellering av spridning, transmission, reflektans och absorption av ljus.

En begränsning i CameoSim är att termofysikaliska beräkningar endast utförs i en dimension. Detta medför att värmeflödet till och från en yta endast beräknas i djupled. För att få en mer detaljerad simulering av värmeledning, strömning och strålning från olika objekt i en scen kan CameoSim styra och hämta resultat från RadThermIR. RadThermIR som har beskrivits i avsnitt 3.1.1 kan utföra termofysikaliska beräkningar i tre dimensioner.

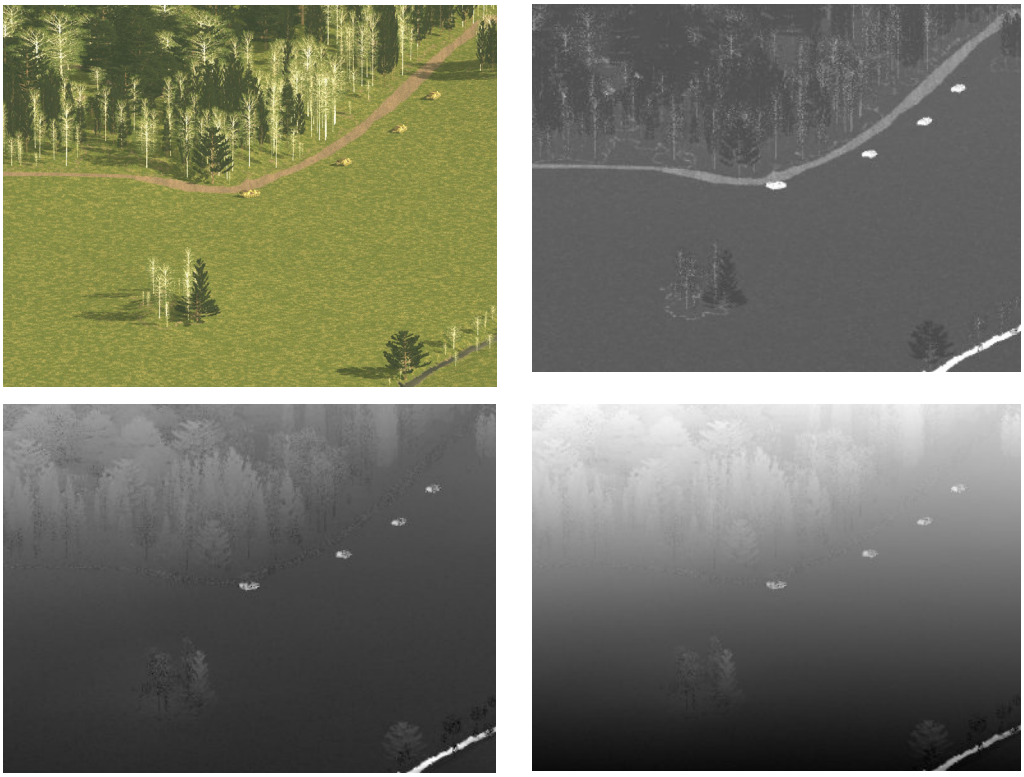
I SIMSENS-projektet har CameoSim använts för att simulera data från IR och visuell sensor. I de följande figurerna visas sensordata från en anflygning mot fyra fordon på ett gräsfält för väderfallen vackert väder, moln och regn. Stridsfordonen i scenen är av typ MTLB och dessa har stått stilla en längre tid så inga inre värmekällor bidrar till signaturen. En mer detaljerad beskrivning av scenarier och vädersimuleringar med CameoSim finns i [9]. Vid simuleringarna av sensordata i olika väderfall har CameoSim använt atmosfärsfiler som har beräknats med MODTRAN (MODerate resolution TRANsmission) [10].



Figur 19. En visuell bild och tre IR-bilder genererade för några olika väderfall. De översta bilderna är genererade i vackert väder; bilden nere till vänster är genererad för ett molnscenario och bilden nere till höger är genererad för ett regnscenario.

I figur 19 visas en visuell bild och tre IR-bilder som är simulerade för väderfallen vackert väder, moln och regn. Sensordata är simulerat för ett avstånd av 850 m och med en depressionsvinkel på 14 grader. FOV för sensorerna är 5×3.8 grader och IR-bilderna är simulerade för en våglängd på 3 - 5 µm. På långt avstånd till målet är dämpningen mycket stor i dåligt väder, vilket göra att målen knappt kan upptäckas.

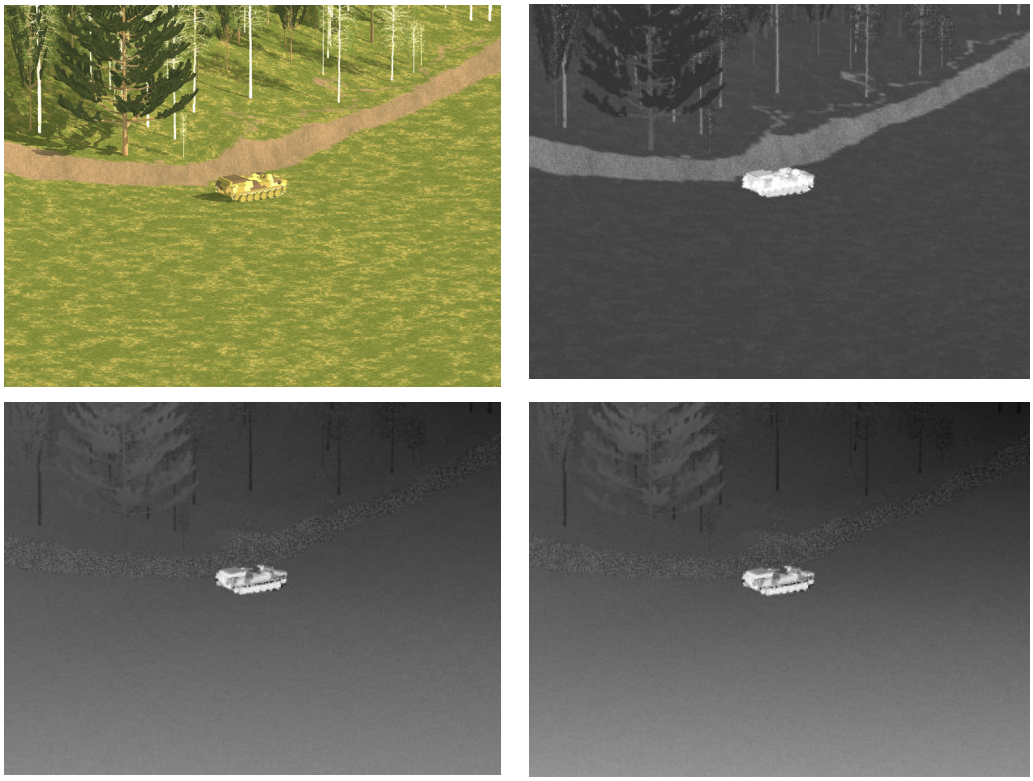
Data i figur 20 är genererat på ett avstånd av 650 m och depressionsvinkeln 14 grader. På detta avstånd befinner sig den simulerade sensorn fortfarande i molnet som har en molnbas på 160 meter över havet. Gångvägen i molnet är dock inte så stor, vilket göra att målen här är lätta att se. Väderfallet med regn ger dock lite högre dämpning än enbart moln även på detta avstånd.



Figur 20. En visuell bild och tre IR-bilder genererade på 650 m avstånd till målet. De översta bilderna är genererade i vackert väder; bilden nere till vänster är genererad för molnscenariot och bilden nere till höger är genererad för regnscenariot.

I figur 21 befinner sig sensorn precis under molnet och på ett avstånd av 450 m från målet. På detta korta avstånd är dämpning inte så stor utan målen framträder tydligt i alla väderfall. I de övre bilderna har solen värmt upp området med jord i dagen.

Simuleringar visar att de olika väderfallen kraftigt påverkar IR-signaturen. En begränsning med MODTRAN är att atmosfären inte kan vara delvis molnig. Det är alltså inte möjligt att ange en atmosfär som är molnig till t.ex. 2/8. Ett alternativ till detta är att använda en tunnare molntyp t.ex. cirrusmoln. Det är emellertid inte enkelt att översätta en delvis molnig hemisfär av en molntyp till en helt täckt hemisfär av annan molntyp. Ett annat sätt att lösa problemet är att göra egna molnmodeller och placera ut dessa som objekt i scenen som skall simuleras.



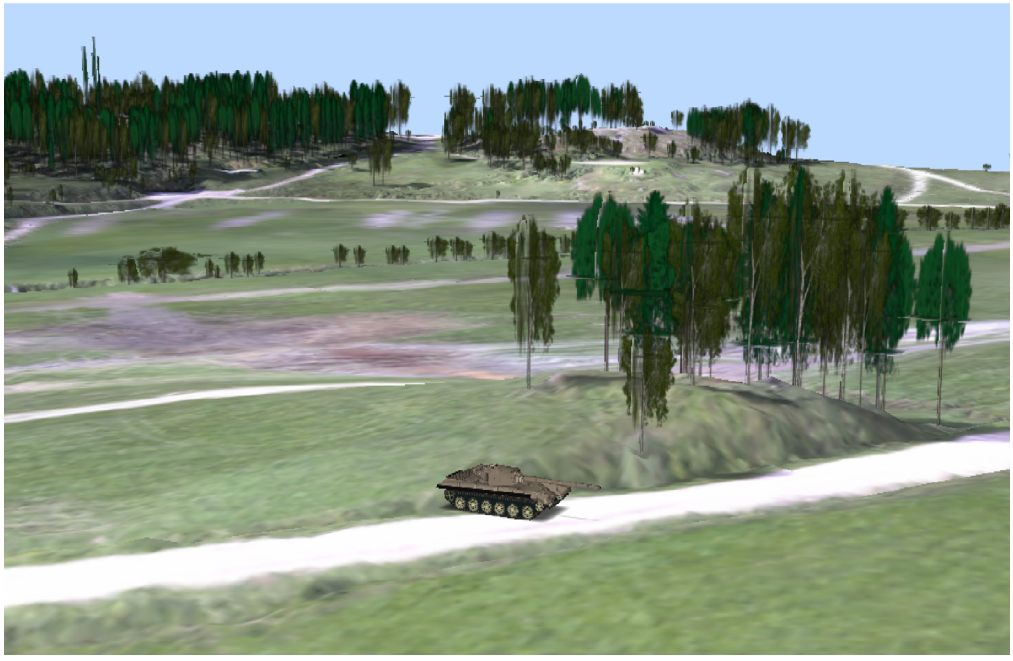
Figur 21. En visuell bild och tre IR-bilder på 450 m avståndet till målet. De översta bilderna är genererade i vackert väder; bilden nere till vänster är genererad för ett molnscenario och bilden nere till höger är genererad för ett regnscenario.

Modellerna som används i CameoSim och MODTRAN kan numera ses som noggranna och är inte gränssättande för hur noga man kan beräkna transmissionen i ett specifikt fall. En begränsning är dock att egenskaperna för materialen som finns i CameoSims materialdatabas endast är giltiga för några väderfall. Här kommer betydande arbetsinsatser att behövas för att utöka materialdatabasen med fler material som är uppmätta under olika väderförhållanden. En annan osäkerhet vid vädersimuleringar är att man inte har en noggrann kännedom om transmissionen utmed utbredningssträckan (regncellers exakta utsträckning, intensitetsvariationer hos nederbörden, komplicerad dynamik som leder till inhomogeniteter etc.). Beräkningarna ger därför oftast typiska värden, snarare än att vara giltiga för specifika fall.

4.3 SceneServer

SceneServer [11] är ett program som utvecklats av FOI för realtidsvisualisering av syntetiska omgivningar. Eftersom programmet är utvecklat av FOI har vi full insikt i koden, vilket gör det möjligt att skräddarsy applikationer och anpassningar. SceneServer är baserat på OpenSceneGraph [12] som är ett scengraf-API för OpenGL.

SceneServer är inte baserat på fysikalisk modellering utan scenen representeras av 3D-geometrier med påklistrade texturbilder. På terrängmodellen som beskrevs i avsnitt 2.1 har texturbilder från visuellt data respektive IR-data klistrats på, vilka insamlades samtidigt med laserskanningen av terrängen. Detta ger en mycket god representation av hur terrängen såg ut vid datainsamlingstillfället, men ger endast begränsade möjligheter att variera scenen för olika väder och tidpunkter. Vid kvalificerade simuleringar av visuella och IR-sensorer används därför helst CameoSim i kombination med RadThermIR.



Figur 22. Visuell bild genererad med SceneServer.

4.4 Laser

Ett program för simulering av aktiva 3D-avbildande lasersystem har utvecklats vid FOI under flera år. Programmet är i dagsläget baserat på SceneServer, vilket ger full insikt i koden och möjligheter att skräddarsy applikationer och anpassningar. Programmet bygger på en ray-trace-liknande princip och är uppdelat i fyra huvuddelar [13]; laserkällan, scenen, atmosfären samt mottagaren.

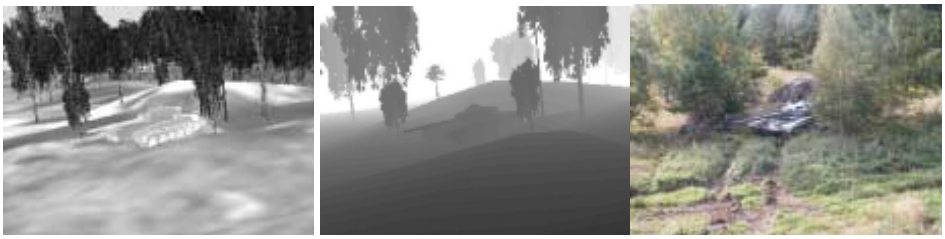
För laserkällan finns möjlighet att använda en godtycklig analytisk eller diskret beskrivning av pulsformen, såväl i temporal (i ljusets utbredningsriktning) som spatiell (tvärs emot utbredningsriktningen) utbredning. I denna fas bestäms även vilken underliggande upplösning som används för att få med egenskaper mindre än ett bildelement.

För att generera ett tidsupplöst eko sänds laserenergi mot scenen i en stor mängd strålar. Ekot genereras sedan genom att analysera hur varje stråle påverkas på sin färd mot scenen. För varje stråle bestäms avståndet till respektive del av scenen samt reflekterad ljusenergi. Den reflekterade ljusenergin bestäms genom att analysera vilket material och i vilken vinkel som strålen träffar respektive material.

När optisk energi färdas genom atmosfären påverkas den av turbulens och partiklar i luften. Atmosfärseffekterna har delats upp och simuleras var för sig. Bland annat sker en dämpning, en scintillation och en utsmetning av energin samt även en strålvandring som får ljusets centroid att avvika från sin utbredningsriktning. Modeller för att simulera vädrets påverkan på lasersystem saknas för tillfället, men med underlag från exempelvis MODTRAN kan även atmosfärspåverkan på lasersensorer simuleras.

Avslutningsvis träffar strålarna mottagarens optik och resulterar i en tidsupplöst elektrisk signal i detektorn. Genom parametrar som beskriver mottagaren, bestäms degraderingen av signalen. Hur den slutliga signalen ser ut bestäms genom systemdefinitionen. Vissa system resulterar i en vågform, medan andra endast ger ett eller flera avstånd i varje bildelement. Modellen tillåter godtyckliga system.

En simulering av ett 3D-avbildande lasersystem mot en stridsvagn T72 i småbruten terräng visas i figur 23. En simulerad reflektansbild visas till vänster och en simulerad avståndsbild visas i mitten. Till höger visas synfältet från sensorn i form av ett foto.



Figur 23. Till vänster visas en simulerad reflektansbild för en stridsvagn T72. I mitten visas en simulerad avståndsbild där ljusare bildpunkter är längre bort. Till höger visas ett foto som visar 3D-lasersensorns synfält.

Under året har programmet effektiviserats och gjorts mer generellt och modulärt. En nackdel med dagens program är att man tydligt kan se att träden är billboard-träd och att marken är platt och obrusig. Terrängmodellen kommer så småningom att förbättras med mer geometriskt korrekta träd. Dessutom kommer ett avståndsbrus motsvarande sensorns riktiga egenskaper att implementeras i modellen.

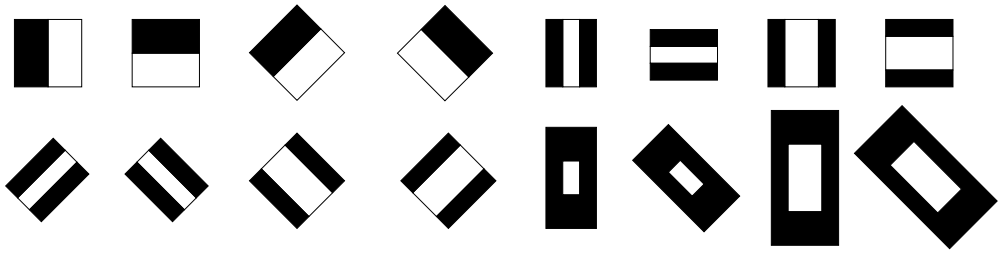
5 Sensordataanalys

I simuleringsverktyget som håller på att utvecklas har flera olika signalbehandlingsalgoritmer integrerats. Signalbehandlingsalgoritmerna är framtagna i andra projekt och har integrerats i MSSLab.

5.1 Detektion

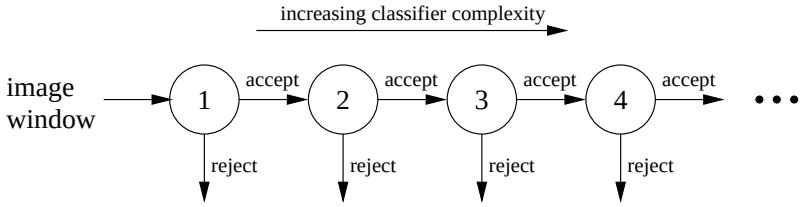
Detektionsalgoritmen som har integrerats till MSSLab togs från början fram för att användas på främst IR-bilder [14]. Det har dock visat sig att samma algoritm kan användas även för att detektera mål i SAR-bilder.

Denna detektionsalgoritm fungerar vanligtvis så att man för varje bildpunkt testar om ett lokalt fönster kring punkten innehåller ett mål. Detektionsalgoritmen genomför testet genom att applicera ett antal operatorer eller filter som känner av lokala strukturer, särdrag, i fönstret. I det aktuella fallet har filter av de typer som visas i figur 24 använts; dessa är känsliga för kanter och linjer av olika slag, samt punkter vars gråvärde avviker från omgivningen. Detektionsalgoritmen viktar sedan samman filtersvaren, och om värdet överstiger en tröskel beslutas att fönstret innehåller ett mål.



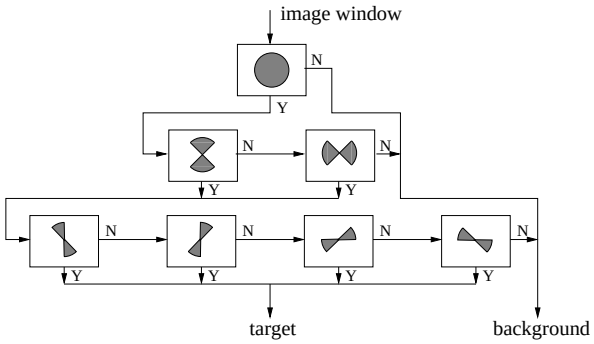
Figur 24. Filtertyper som används av den implementerade detektionsalgoritmen. Genom att variera höjd, bredd och position kan mer än 100000 olika filter genereras i fönster av storleken 20×20 bildelement.

Träningen av detektionsalgoritmen består i huvudsak av att bestämma vilka filter som ska användas, samt hur filtersvaren ska viktas samman. Detektion av markmål är på grund av den svåra klottermiljön ett utomordentligt svårt problem, så för att uppnå acceptabla prestanda med avseende på falsklarms- och detektionssannolikhet krävs ett mycket stort antal filter (flera tusen). Även om svaren från filtren i figur 24 kan beräknas mycket effektivt, är det omöjligt att i realtid applicera flera tusen filter i varje bildpunkt. En elegant lösning på detta problem är att använda en sekvens av detektorer, så som visas i figur 25. Först bearbetas bilden med en enkel detektor som har en viss förmåga att förkasta bakgrunder och en hög sannolikhet att acceptera ett verkligt mål. Om en signifikant andel (kanske hälften) av alla bildpunkter på detta sätt kan avfärdas, återstår en mindre del att bearbeta i nästa steg. Man har därför råd att använda en mer komplex detektor för att ytterligare reducera antalet bakgrundsomgivningar. Sekvensen förlängs sedan på samma sätt tills önskad falsklarmsannolikhet uppnåtts. Med denna metodik kan antalet filter som i snitt måste appliceras i varje bildpunkt reduceras med mer än 90%.



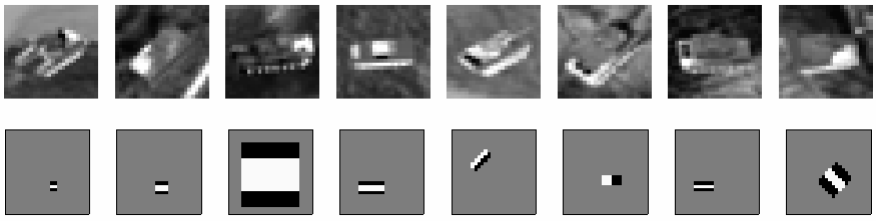
Figur 25. Sekvens av detektorer.

För att ytterligare effektivisera beräkningarna kan man även dela upp varje steg i ett antal detektorer, var och en specialiserad på ett visst vyintervall. Det är rimligt att anta att ett flygande spaningssystem med hjälp av navigeringssensorer och en terrängmodell har en god kunskap om avståndet till och orienteringen av markplanet. Markfordon undviker också i största möjliga mån branta lutningar. Sammantaget medför detta att osäkerheten om elevationsvinkeln mellan markplanet (och därmed fordonet) och spaningsplattformen är relativt liten. Azimutvinkeln, dvs. fordonets orientering i markplanet, är dock i allmänhet helt okänd. För att underlätta detektionen kan man därför träna detektorer att känna igen mål inom ett visst azimutintervall, och sedan kombinera dem enligt figur 26.



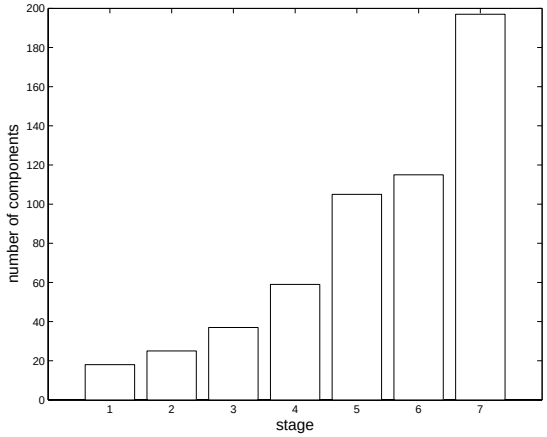
Figur 26. Hierarki av detektorer. Den sekventiella bearbetningen sker nu uppifrån. För varje steg görs en allt finare uppdelning i vyintervall för att klara det allt svårare diskrimineringsproblemet. Varje modul består i sin tur av en sekvens av detektorer.

Låt oss i mer detalj studera resultat för en modul (ett vyintervall) på den finaste nivån i figur 26. Exempel på träningsdata och några valda filter för IR-fallet visas i figur 27. Den träningsmetod som använts väljer successivt ut de filter som på bästa sätt särskiljer mål och bakgrund.



Figur 27. Exempel på träningsdata och några valda filter för att särskilja mål från bakgrund.

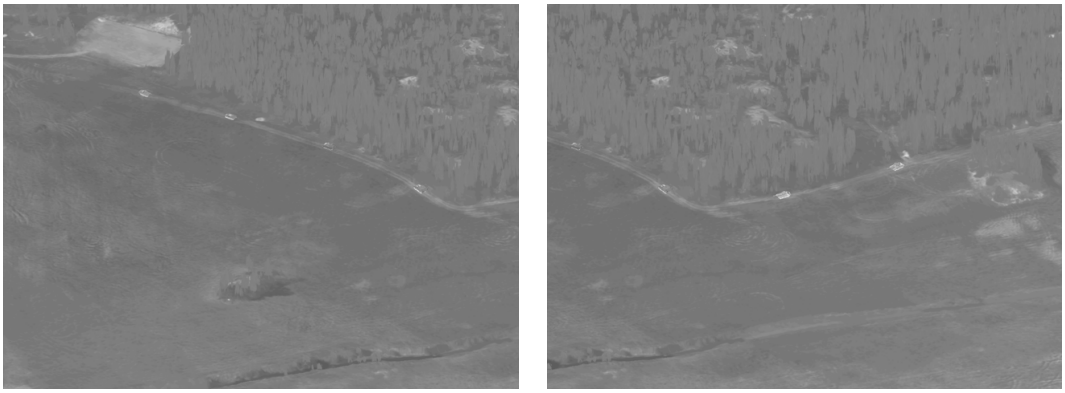
I figur 28 visas hur antalet filter i successiva steg ökar, som följd av att det blir allt svårare att särskilja återstående bakgrunder från målen.



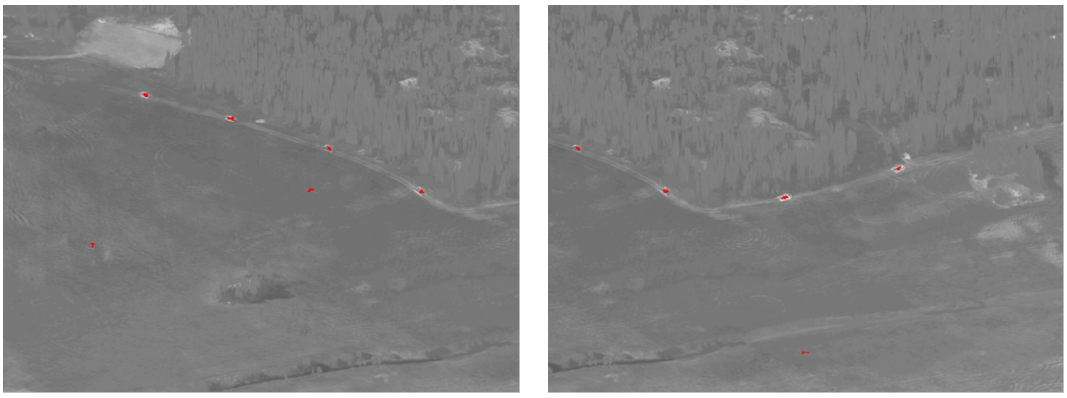
Figur 28. Antalet filter som krävs i de olika stegen i detektorsekvensen för att uppnå prestandakraven.

Algoritmen är utformad för att uppnå hög detektionssannolikhet (mer än 90 %) vid låg falsklarmsintensitet (någon enstaka indikering per bild) för signaturreducerade mål med låg kontrast mot bakgrunden. Den skall således kunna fungera även mot mål som inte kan detekteras med enkla ”hot spot”-detektorer (t.ex. fordon med avstängd motor) och i komplexa klottermiljöer. För att uppnå dessa prestandakrav krävs en minsta upplösning som kan skattas till omkring 0.5 meter per pixel.

I figur 29 visas bilder ur en testsekvens med hög bildkontrast, motsvarande en solig sensommardag. Bilderna simulerar en högkvalitativ IR-sensor känslig i våglängdsbandet 8-9 μm. Sensorn har en fokalplansmatris med 640×480 pixlar. Synfältet är 5 grader i horisontell led, vilket på det aktuella målavståndet resulterar i en pixelstorlek på ca 0.4 m. Utdata från detektionsalgoritmen visas i figur 30. Detektionsprestandan är mycket god och samtliga mål detekteras. Falsklarmsintensiteten är låg med endast någon enstaka felaktig indikering. Man bör dessutom notera att falsklarmsintensiteten kan minskas ytterligare genom att applicera en målföljningsalgoritm på indikeringarna, eftersom de flesta falsklarm bara förekommer i enstaka bilder.



Figur 29. Bilder ur simulerad testsekvens.

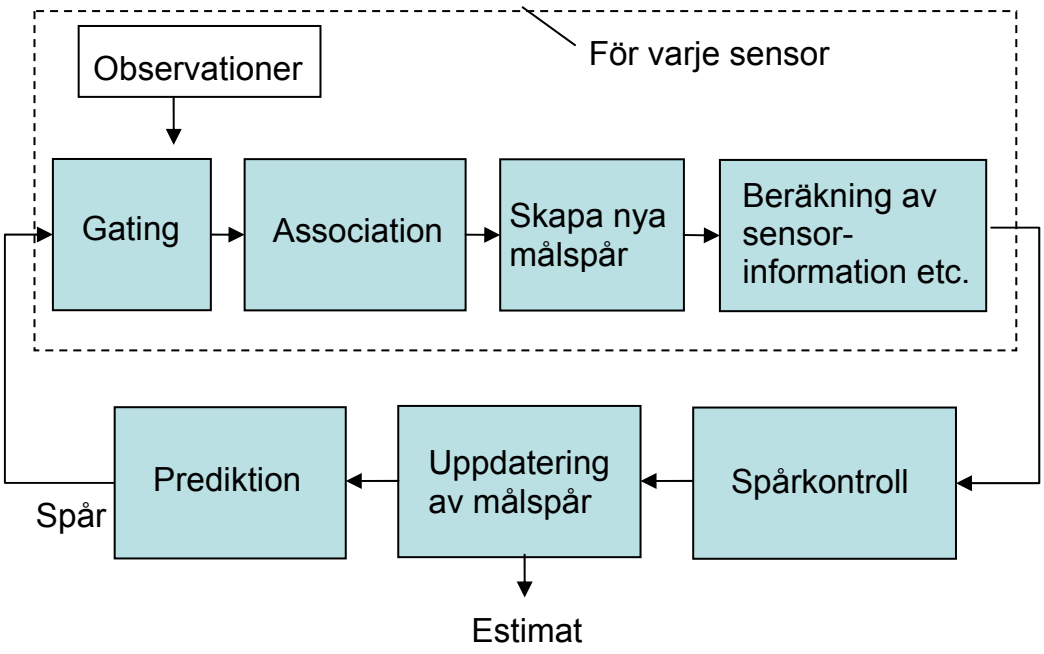


Figur 30. Indikeringar från detektionsalgoritmen för bilderna i figur 29 .

5.2 Målföljning

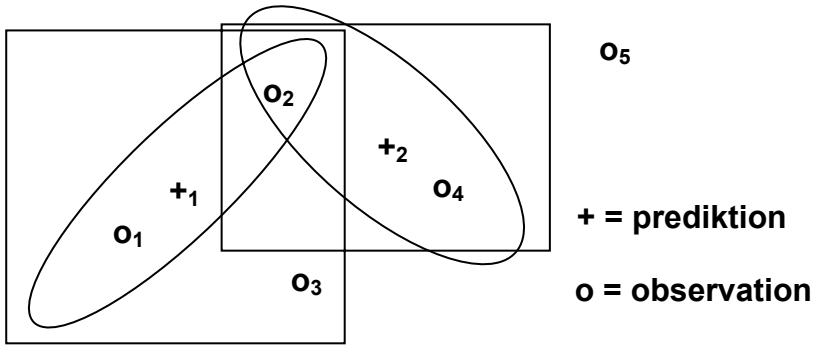
En målföljningsalgoritm har integrerats till MSSLab. Denna kallas MSMT-målföljaren (MultiSensorMultiTarget) och är utvecklad för multisensorsystem. MSMT-trackern är implementerad i C++ och grundtanken med dess design är att den ska vara allmän. Den klarar flera olika sensorer, många mål, olika rörelsemodeller och olika målföljningmetoder. Nedan följer en kort beskrivning av den integrerade MSMT-målföljaren.

En målföljare har som uppgift att utifrån observationer av ett mål skapa *målspår* över tiden. Dessa *målspår* ger information om objektens rörelse och kan t.ex. användas för att estimera målets position och hastighet. I figur 31 nedan visas en schematisk bild av MSMT-målföljaren. Denna målföljare är uppbyggd för att kunna hantera flera olika sensorer och kunna följa flera mål samtidigt.



Figur 31. Målföljningsloop för MSMT-målföljaren.

Målföljningsalgorimen startar med att observationer från ett mål kommer in till gatingen där en grovsortering av data sker. Detta för att underlätta för den efterföljande associationen som är beräkningstung.



Figur 32. Gating

Figur 32 visar ett exempel på hur gatingen går till. De fem observationerna ska associeras till prediktionerna för de två målspåren. I detta fall används två gatar i sekvens, en rektangulär och en elliptisk gate. Den rektangulära gaten är minst tidskrävande och sållar snabbt bort observation nummer fem. Därefter används de bättre men mer beräkningstunga elliptiska gatarna på de återstående observationerna och observation 3 sållas bort. Efter detta återstår 3 observationer vilka ska associeras till de två målspåren.

Associeringen av observationer till målspår kan ske enligt olika principer. I MSMT-målföljaren finns två associationsmetoder implementerade: *Global Nearest Neighbor* (GNN) och *Joint Probabilistic Data Association* (JPDA).

GNN (Global Nearest Neighbor) är en enkel och mycket vanlig associationsalgoritm. Den fungerar så att observationen som har det kortaste avståndet till den predikterade positionen anses med denna algoritm vara den rätta. När det finns flera objekt i området som observationerna kan härröra från väljs de kombinationer som minimerar det totala avståndet mellan observationer och prediktioner. I exemplet ovan kommer då observation ett att associeras till målspår ett och observation fyra kommer att associeras till målspår två, eftersom dess avstånd till prediktionen är kortast.

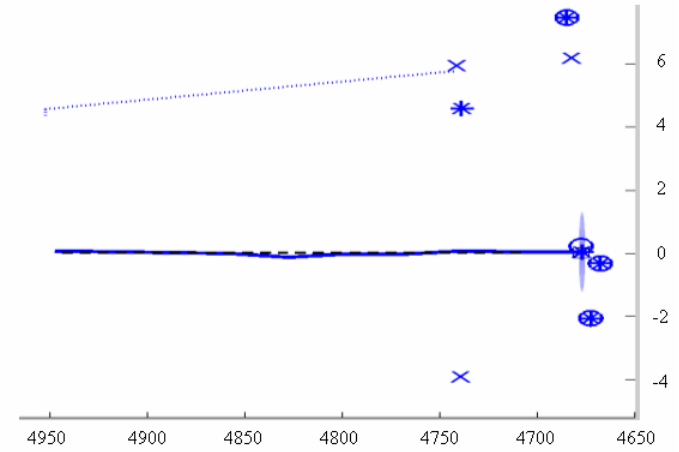
En annan mycket vanlig associationsmetod är JPDA (Joint Probabilistic Data Association). Vid uppdateringen av målspår med denna metod viktas alla observationer inom gaten samman så att observationer som ligger närmast prediktionen får störst betydelse. I exemplet ovan kommer både observationerna ett och två att användas vid uppdateringen av målspår ett, och på samma sätt kommer observationerna två och fyra att användas vid uppdateringen av målspår två.

Många studier visar att JPDA är bättre än GNN på att bibehålla enstaka målspår i en miljö med mycket klotter. En nackdel med JPDA är dock att den har svårt att hålla isär målspår för objekt som rör sig nära varandra. Därför är det vanligt att på olika sätt kombinera GNN och JPDA för att dra nytta av båda algoritmernas styrkor.

För observationer som inte kan associeras till något målspår skapas helt nya målspår. Sensorinformation som kovariansmatriser, kalman gain etc. beräknas för respektive sensor, vilket sedan används vid den globala uppdateringen av målspåren.

För att målföljningsalgoritmen ska fungera bra behövs spårkontroll för att ta bort dåliga spår, slå ihop spår som tillhör samma objekt etc. Kvalitén på ett spår har i MSMT-målföljaren analyserats med en metod som heter SPRT (Sequential Probability Ratio Test) som beräknar ett ”track score” för de olika målspåren. Målspår som associeras med en observation får då högre “track score” och målspår som inte associera med en observation får lägre ”track score”. Om track score överstiger en övre tröskeln sägs spåret vara konfirmerat och om track score understiger en undre tröskel tas målspåret bort. Målspåren som är kvar uppdateras sedan och predikteras framåt i tiden.

Nedan visas ett exempel på ett målspår. I detta exempel har målet befunnit sig på 4950 meters avstånd till sensorn och målet har sedan rört sig mot sensorn som är kopplad till en målföljningsalgoritmen.



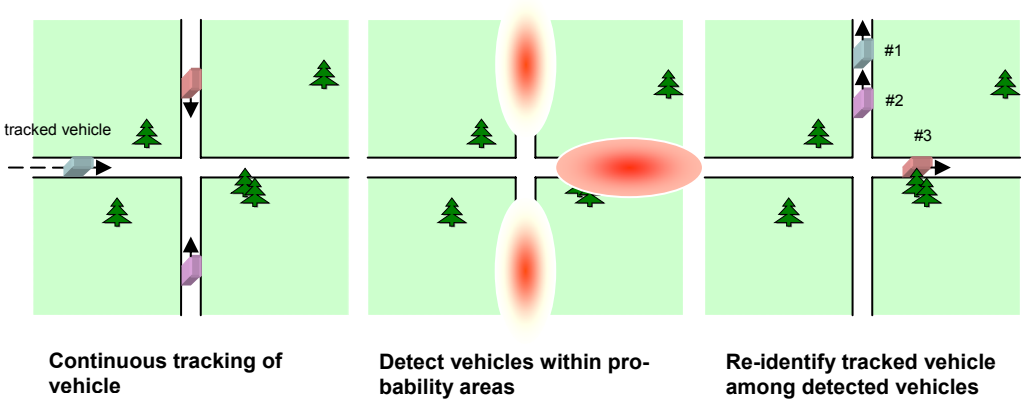
Figur 33. Ett målspår under anflygningen.

Den svarta streckade linjen visar börvärdet som en ideal målföljningsalgoritm skulle få, och den blå linjen visar målspåret. De blå stjärnorna visar spårens nuvarande estimat och de blå kryssen visar spårens prediktioner. De blå cirklarna visar observationer som detektorerna har levererat till målföljningsalgoritmen. Den tunna streckade linjen är ett målspår som ej är bekräftat och som till slut tas bort.

5.3 Återigenkänning

En ny funktion som integrerats i MSSLab under 2008 är återigenkänning [15]. Ett typiskt scenario för återigenkänningen visas i figur 34. Ett (rörligt) fordon på marken följs från t.ex. en UAV under ett visst tidsintervall. Under tiden fordonet är synligt extraheras robusta spatiella, temporala och spektrala särdrag från fordonet. För att lösa andra spaningsuppgifter behöver UAV:n sedan avbryta den aktiva följningen under en tidsperiod. Avbrottet kan t.ex. bero på skylning eller andra högprioriterade uppgifter. UAV:n försöker därefter återfinna det rörliga fordonet. De särdrag som tidigare extraherats används för att med hög signifikans avgöra om det är samma fordon som tidigare följts.

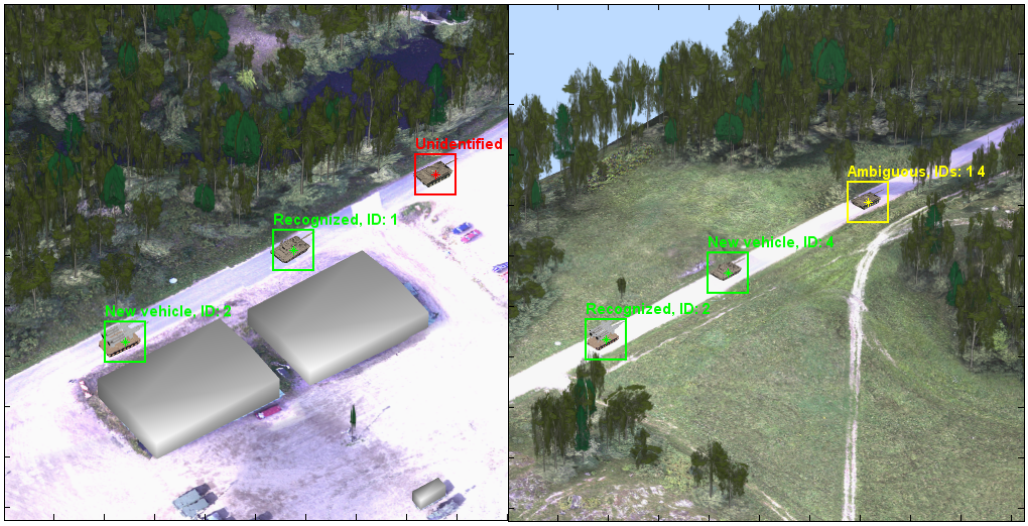
UAV:n fortsätter sedan att följa fordonet och dynamiskt uppdatera relevanta egenskaper.



Figur 34. Typiskt scenario för återigenkänning.

En svårighet med igenkänning av fordon är att skillnaden mellan olika objektinstanser kan vara mycket liten. För bilar kan det vara dörrhandtagets läge, form hos sidoruta etc. Återigenkänningsalgoritmen måste även kunna hantera olika belysningseffekter och att fordon är synliga från olika vinklar.

Den återigenkänningsmetod som integrerats i MSSLab bygger på Color Cooccurence Matrix (CCM) [16]. De särdrag som används i CCM är baserade på färg och textur, och representeras som histogram. En fördel med metoden är att den ger rotationsinvarianta matchningar. CCM kan användas både på naturliga färgbilder (RGB) och IR. Metodens känslighet kan justeras genom att välja färghistogrammets upplösning (antalet kvantiseringsnivåer) Om många nivåer används, kommer matchningarna att bli mer exakta under bra förhållanden, men samtidigt ökar känsligheten för förändrade ljusförhållanden och brus. Ett resultat av återigenkänningen visas i figur 35.



Figur 35. Återigenkänning av fordon på march.

5.4 Ruttplanering och sensorstyrning

En ny funktionalitet som håller på att integreras till MSSLab är automatisk ruttplanering och sensorstyrning. Målsättningen med detta arbete är att rutt och pekriktningen för sensorn ska väljas automatiskt för att så mycket relevant sensordata som möjligt ska kunna samlas in. Vad som är relevant sensordata är beroende av målet för uppdraget och inkluderar t.ex. upptäcktssannolikheter samt prestandamått för målpositioner och målidentiteter. Dessutom bör även hänsyn tas till restriktioner som kommer ur hot- och säkerhetsaspekter.

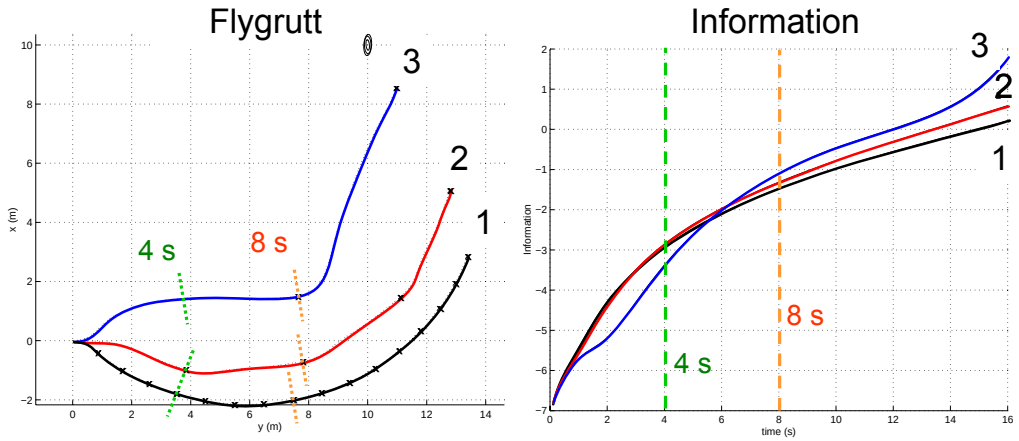
Sammantaget kan planeringsproblemet formuleras som ett optimeringsproblem med bivillkor. Tyvärr medför dock detta angreppssätt en rad problem. För det första kan det vara svårt att kvantifiera vad som är relevant information och hur man sammanväger konkurrerande önskemål från olika intressenter. Även om man lyckas formulera en relevant målfunktion så är denna funktion i praktiken alltid icke-linjär och icke-konvex, och förmodligen även stokastisk, vilket generellt innebär att det är extremt svårt och beräkningskrävande att hitta optimum med de numeriska metoder vi är hänvisade till.

Ett sätt att hålla nere komplexiteten hos planeraren är att skapa en planeringshierarki där problemet delas upp i delproblem som var och en är mer hanterbar. Man måste dock vara försiktig hur denna hierarki definieras. Om detta görs på ett olämpligt sätt finns risk att bara dåliga planer genereras. Här följer ett exempel på en uppdelning av en planering i tre olika moder. Första moden innebär att man på uppmaning från t.ex. återigenkänningen eller målföljningen följer ett givet mål som befinner sig inom en (visuell) sensors vy samtidigt som UAV:n flyger på ett sätt som maximerar inmätningsprestandan. Den andra moden styr UAV:n och sensorn så att målspåret för kända mål uppdateras tillräckligt ofta för att de inte ska tappas bort. Slutligen den tredje moden är ren spaning eller, med andra ord, yttäckning för att leta efter nya intressanta objekt.

Första moden är baserad på informationsteori och är inspirerat av forskning inom optimal styrning för målföljning med passiva sensorer [17]. Antag att fordonets position är p och att vår kunskap om positionen representeras av storheten $\tilde{p}$. Eftersom vi inte är helt säkra på positionen p betraktar vi $\tilde{p}$ som en stokastisk variabel – säg för enkelhets skull en normalfördelad variabel med medelvärde m och kovariansmatris C. Mätningar vi gör ska syfta till att reducera osäkerheten så mycket det går, eller med andra ord, maximera informationen om målet. Frågan är dock hur vi kvantifierar information? I detta fall kan informationsinnehållet helt enkelt definieras som storleken av inversen på variansen $\hat{P}^{-1}$, vilket ofta kallas för Fisher-information. En fördel med detta informationsmått är att det har additiva egenskaper – det är lätt att lägga till information som kommer från nya observationer. Denna praktiska egenskap utnyttjas flitigt i de informationsfilter som är vanliga i datafusionssammanhang.

Förväntad informationsvinst kan användas vid planering av spaning, övervakning och utforskning. Om ett problem formuleras så att intressanta egenskaper representeras som tillståndsvariabler, så är en lämplig strategi att maximera informationen, eller ekvivalent, minimera osäkerheten hos variablerna. Antag att en UAV med en passiv optisk sensor ska mäta in positionen hos ett objekt på marken på kortast möjliga tid – helt i analogi med exemplet ovan. Problemet är att välja lämpliga styrsignaler så att detta krav uppfylls. Notera att en passiv

sensor bara har förmåga att mäta en relativ vinkel till objektet, den har ingen möjlighet att direkt mäta avståndet. Uppgiften kan formuleras som ett optimeringsproblem där man söker en styrsignalsekvens som maximerar (minimerar) en viss målfunktion. Målfunktionen kan exempelvis vara determinanten av objektets informationsmatris $Y = \hat{P}^{-1}$ som alltså är ett mått på tillförlitligheten eller informationsinnehållet i skattningen.



Figur 36. Lokalisering av objekt med passiv sensor i fallet med en rörelsefrihet i två dimensioner. Till Vänster: Objektet är stationärt i (10, 10), farkosten startar i (0,0). Tre olika fall med olika lång planeringshorisont visas, omplaneringar indikeras med kryss. Till höger: Informationsvärdet för de tre fallen som funktion av tiden.

Planeringen har en viss planeringshorisont och det är i regel informationsvärdet vid denna horisont som ska maximeras. Olika tidshorisonter kan ge vitt skilda resultat. I figur 36 visas planeringsresultatet för tre olika planeringshorisonter i det ovan beskrivna inmätningsexemplet. Till höger visas informationsvärdet för objektet i de tre fallen. En kort horisont (1) medför att farkosten inledningsvis färdas i princip vinkelrätt relativt riktningen till objektet. Detta beror på att den passiva sensorn enbart kan mäta vinkeln till objektet. Med en lång horisont går farkosten mer rakt på för att vika av mot slutet. Jämför man informationen ser man att resultatet med den korta horisonten (1) har mest information till en början, men passeras av den mellanlånga horisonten (2) vid t = 3, och av den långa horisonten (3) vid t = 6. Med lång horisont (3) har man alltså råd att inhämta lite information inledningsvis om man vet att man kommer få mycket information på slutet.

6 Nätverksdistribuerad simulering med HLA

En stor del av årets verksamhet har varit att integrera scensimuleringsverktyg och signalbehandlingsmetoder till MSSLab. För detta har HLA använts, vilket är en förkortning för ”High Level Architecture” [18]. HLA är en standard för hur olika simulatorer skall kommunicera data emellan sig över närverk, för att upprätthålla en gemensam och synkroniserad lägesbild under simuleringen. HLA har utarbetats successivt sedan mitten av 1990-talet med Amerikanska försvarsmakten som huvudintressent och pådrivande part. Standardiseringsorganet som driver utvecklingen heter ”Simulation Interoperability Standards Organization” (SISO). Standarden syftar till att underlätta integrering av simulatorer, men det krävs ett antal kompletterande dokument för att definiera upp en simuleringsomgivning. De två viktigaste är ”Federation Agreements” och ”Federation Object Model” (FOM). En standard FOM har definierats upp av standardiseringskommittén för HLA. Denna FOM kallas RPR-FOM [19] och har sina rötter i den gamla standarden ”Distributed Interactive Simulation” (DIS). En FOM beskriver vilka objekt som kan finnas i simuleringen, vilka meddelanden (interaktioner) som kan skickas mellan olika federater (simulatorer eller simuleringskomponenter), hur data kodas i meddelanden och objektens data variabler (attribut), synkroniseringspunkter m.m.

Kommunikationen mellan simulatorerna hanteras av en RTI (Run-Time Infrastructure). RTI:n är ett program som fungerar som en kopplingsstation mellan olika federater i simuleringen. Denna komponent specificeras av standarden, men tillhandahålls av tredje part. FOI använder RTI:n som försvarsmakten via FMV valt vilket är pRTI utvecklad av Pitch [20].

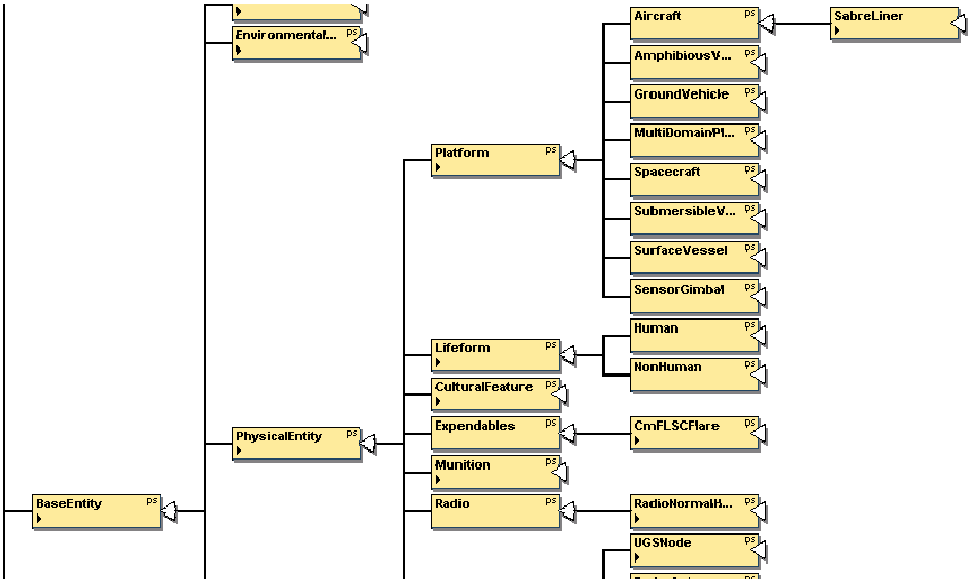
MOSART var ett försvarsmaktsprojekt som pågick på FOI för några år sedan [21]. I detta projekt implementerades en stor kodbas för HLA och RPR-FOM:en. Man definierade även upp ”MOSART Federation Agreements” [22] och gjorde en uppsättning applikationer för hantering av federationer, uppsättning och styrning av scenarier m.m. Utan detta arbete skulle MSSLab inte kunnat realiseras med en rimlig arbetsinsats. Flera projekt har använt kodbasen som togs fram av MOSART och en kontinuerlig utveckling och uppdatering sker fortfarande.

6.1 Introduktion till FOM

En FOM (Federation Object Model) är en fil i XML-format som beskriver vilka objekt, interaktioner m.m. som finns i simuleringen. Varje federat definerar sedan upp ett subset av denna FOM i någon som kallas ”Simulation Object Model” (SOM). För att man ska kunna koppla ihop olika simulatorer är det viktigt att de använder samma FOM, eller åtminstone att de objekt, interaktioner och datatyper som kommuniceras eller är gemensamma mellan federaterna i simuleringen måste vara identiska i FOM:en. RPR-FOM är en standard FOM som är mycket vanligt förekommande. RPR-FOM:en innehåller objekt för att främst simulera plattformar, såsom flygplan, markfordon m.m. som agerar tillsammans i en större distribuerad simulering.

6.1.1 HLA-objekt

Objekten i FOM:en är inordnade i en hierarki där objekten går från det mest basala funktionaliteten till mer och mer specialiserade objekt. En sådan hierarki kallas för arvshierarki och är vanlig inom datorprogrammering. Funktioner läggs till objekten ju längre ner i hierarkin man kommer, men inga tas bort. Olika förgreningar används för att skapa olika typer av snarlika objekt, såsom exempelvis ”Aircraft” i figur 37, som i likhet med ”AmphibiousVehicle” och ”GroundVehicle” m.fl. är en ”Platform” som i sin tur är en ”PhysicalEntity” och så vidare. ”Aircraft” och ”GroundVehicle” har således alla funktioner och data som en ”Platform” har, men dessutom har de ett antal funktioner som är unika för flygplan respektive markfordon.



Figur 37: Del av arvshierarkin i RPR-FOM:en plus några tillägg för FOI-specifika objekt, t.ex. ”SabreLiner” och ”SensorGimbal”. Ovanför ”BaseEntity” finns bara ”HLAObjectRoot”, dvs. toppnoden.

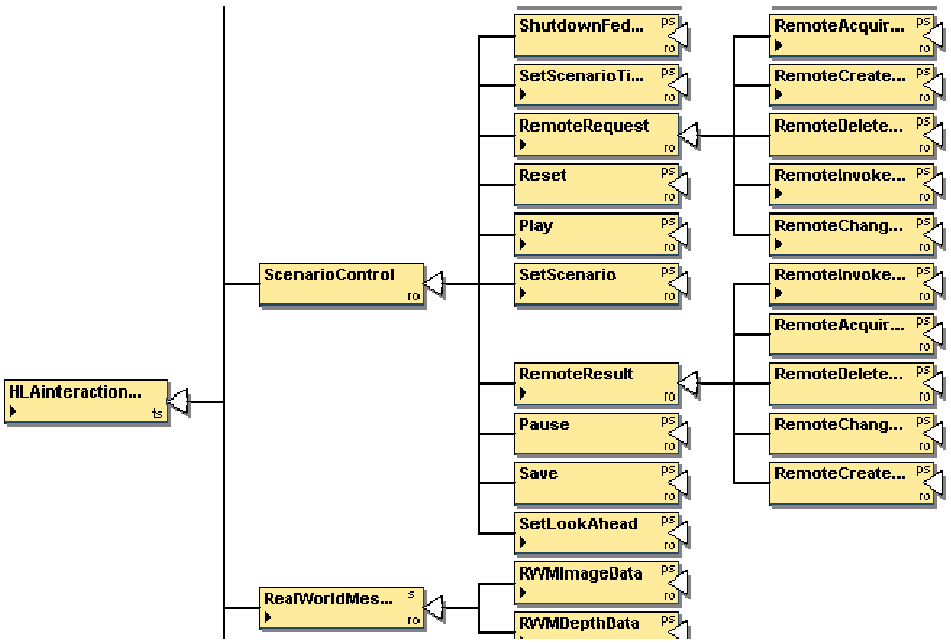
Ett objekts data (attribut) speglas i varje tidpunkt mellan alla de federater som känner till objektet (prenumererar på det). Ett objekt ägs av en federat och normalt är det federaten som uppdaterar och förändrar attributen i objektet. Man säger att en federat publicerar ett objekt när den skapar och äger det. Ägandet av ett objekt eller t.o.m. ett enskilt attribut i objektet kan överlåtas mellan federater, men i dagsläget är det en funktion som vi inte använder.

En federat kan äga och prenumerera på flera typer av objekt. Varje objekt kan finnas i flera instanser, t.ex. kan det finnas många ”GroundVehicle” och dessa kan alla hanteras av en federat eller så kan de vara spridda över flera olika federater. Olika objekt får en unik identifieringskod av RTI:n vid exekvering av simuleringen. Det är ofta praktiskt att också använda unika namn på olika instanser av samma objekttyp för att skilja dem åt systematiskt mellan körningar.

6.1.2 Interaktioner

Tidsstämplade meddelanden kan skickas mellan federater med hjälp av interaktioner. Interaktioner kan, liksom objekten, inordnas i en arvshierarki, se figur 38. I RPR-FOM:en har man skapat en ganska platt hierarki, medan de tillägg vi gjort ligger samlade i ett par grenar med större djup. På motsvarande sätt som

federater prenumererar på och publicerar objekt, publicerar och prenumererar federaterna på interaktioner. En interaktion skickas till alla som prenumererar på den typen av interaktioner, inklusive alla de interaktioner som ärvts från den. Detta är viktigt att tänka på när man skapar sin interaktionshierarki, så att man inte behöver införa andra, metoder för att skilja interaktioner åt i de olika federaterna.



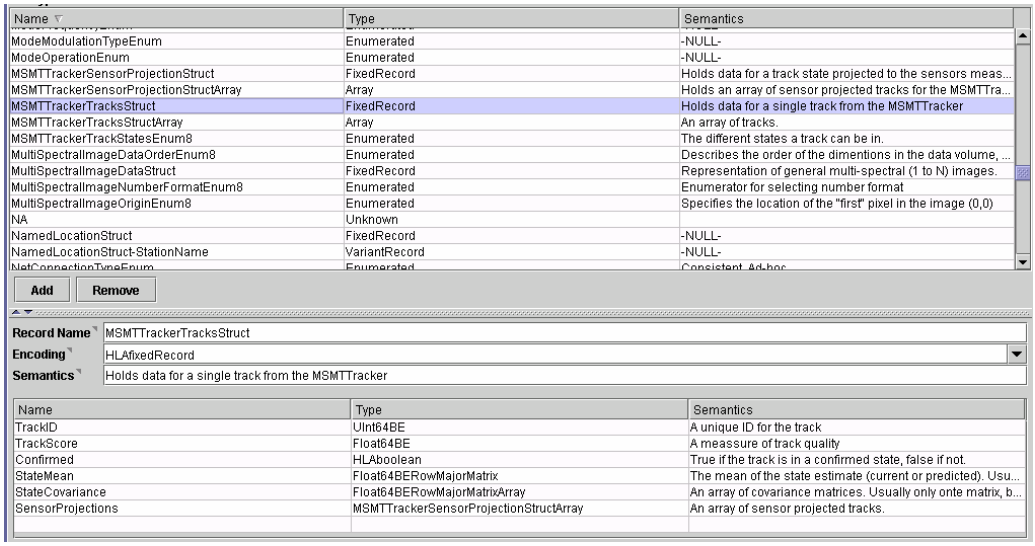
Figur 38: Exempel på en del av interaktionernas arvshierarki. Dessa interaktioner används av MOSARTs federations- och scenariohanterare för att styra och sätta upp en simulering.

Interaktionerna är kortlivade och existerar egentligen endast i simuleringen tills den tidsstämpel de tilldelats passerats. Federater kan dock ha olika ”samplingsintervall”, varför alla interaktioner som levererats till en federat under hela dess sampelintervall kommer att sparas och vara tillgängliga för federaten i nästa tidpunkt när den aktiveras.

En interaktion kan vara sammansatt av flera olika attribut och dessa kan i sig vara sammansatta datatyper. Det är därför möjligt att skicka ganska stora och komplexa meddelanden mellan federater, till priset av ökad nätverksbelastning.

6.1.3 Datatyper

Datatyper är de grundläggande informationsbärarna i objekt och interaktioner. De kan i sig vara sammansatta i s.k. strukturer av andra datatyper, eller bara representera enkla heltal eller flyttal med olika antal bitar, precision och kodning, se figur 39.



Figur 39: Ett exempel på hur en sammansatt datatyp kan se ut. ”MSMTTrackerTracksStruct” innehåller en uppsättning variabler som kodar ett målspår i en MSMTTracks-interaktion.

Datatyper representeras inte av arv, vilket gör hanteringen av sammansatta datatyper lite omständligt. HLA-standarden föreskriver en något omständlig strategi rörande införandet av datatyper, vilket resulterar i mängder av ekvivalenta typer i FOM:en. Den totala listan över använda datatyper är därför ganska omfattande.

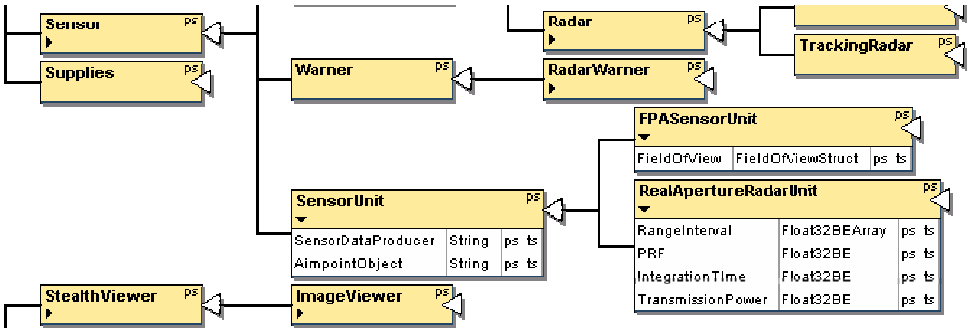
6.2 Tillägg till RPR-FOM:en för distribuerad sensorsimulering

När man studerar RPR-FOM:en ser man mycket tydligt att den är gjord för interaktiva spelliknande simuleringar där människor skall agera tillsammans i distribuerade simuleringar i olika miljöer. Användningen av HLA i MSSLab är lite annorlunda. Här vill vi använda funktionaliteten i HLA för att synkronisera och kommunicera data mellan olika delfunktioner i detaljerade sensorsystemsimuleringar för att studera autonoma funktioner och sensorfunktion i komplexa miljöer och scenarier. Sensorspecifika delar saknas helt i RPR-FOM:en och behöver därför läggas till. De delar som finns i RPR-FOM:en kan dock användas för att driva scenarier och de miljöer som sensorerna skall agera i. Detta medför att sensorerna kan agera tillsammans med t.ex. plattformssimulatorer i större scenarier för att höja realismen. En begränsning i dagsläget är att sensorsimuleringarna bara i undantagsfall kan köras i realtid, vilket utesluter man-in-the-loop simuleringar i realtid.

6.2.1 Tillagda sensorobjekt

Förutom ett särskilt ”SensorGimbal”-objekt under grenen ”Plattform”, har en särskilld gren, ”SensorUnit”, lagts till under ”Sensor” objektet i ”PhysicalEntity”-grenen av RPR-FOM:en, se figur 37 och figur 40. Tanken med ”SensorUnit” och dess subklasser är att de ska utgöra ett kontroll- och positioneringsinterface till andra federater som vill placera ut sensorn eller styra olika inställningar hos den under simuleringen. En ”SensorUnit” är en ”BaseEntity” och således fysiskt utplaceringsbar i den virtuella miljön, även om den inte nöd-

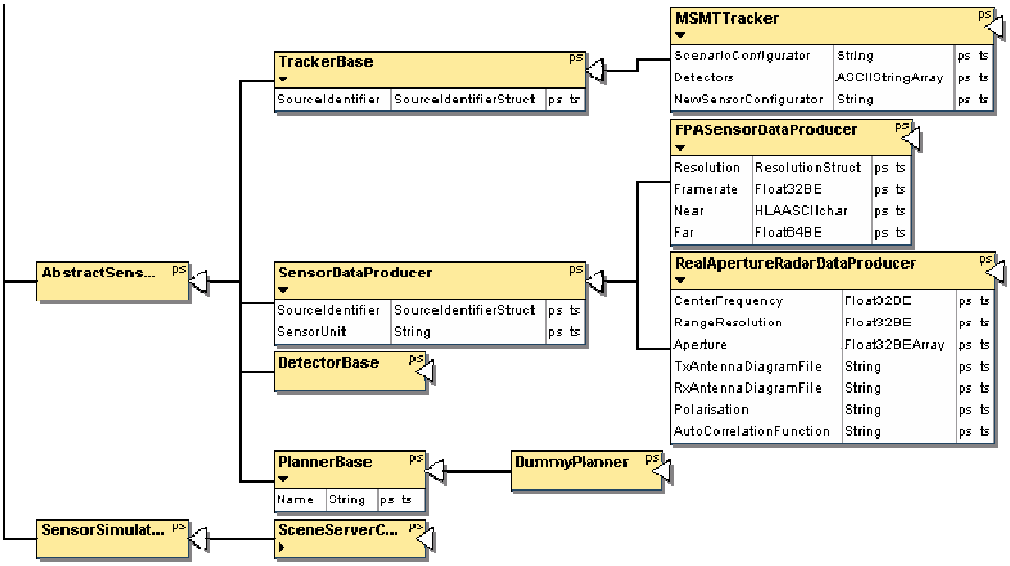
vändigtvis är modellerad med en 3D-modell. Ett vanligt förfarande är att koppla en ”SensorUnit” till antingen ett ”SensorGimbal”-objekt eller ett annat ”Plattform”-objekt och låta sensorn följa objektets förflyttningar. Subobjekt till ”SensorUnit” tillför sensorspecifika inställningsmöjligheter till olika sensorer som vill simuleras.



Figur 40: Tillagda objekt under ”Sensor” i ”PhysicalEntity”-grenen i RPR-FOM:en. ”Sensor” ärver direkt från ”PhysicalEntity”. De tillagda objekten har expanderats så att deras attribut visas.

En ”SensorUnit” används i par med abstrakta sensorobjekt av typen ”SensorDataProducer”, se figur 41, vilket utgör ett interface till den mer statiska konfigurationen av en sensor. Anledningen till att man vill separera sensorkonfigurationen i två objekt, är att underlätta att en annan federat än sensorsimuleringsfederaten publicerar och kontrollerar de justerbara sensorparametrarna. Detta hade teknisk varit möjligt att göra genom att överlåta ägandet av enskilda parametrar i ett enda objekt, men det hade varit svårare att hantera i scenarioverktyg m.m.

Subklasser till ”SensorDataProducer”-objekt används för att instruera olika sensorsimuleringsfederater att generera data och för att sätta upp statiska parametrar i sensorn. Genom att ha flera instanser av t.ex. ”FPASensorDataProducer”-objekt, kan multipla kameror simuleras av en sensorfederat.



Figur 41: Denna figur visar de grenar med sensorsimuleringsobjekt som lagts till RPR-FOM:en direkt under ”HLAObjectRoot”. Arbete pågår kontinuerligt med att utöka dessa grenar vartefter ny funktionalitet läggs till i MSSLab.

Målföljningsfederater kan konfigureras med objekt under ”TrackerBase”. I nuläget finns stöd för en implementering av en MSMT-målföljare som beskrivits i avsnitt 5.2.

Stöd för objektdetektorfederater och sensorplaneringsfederater finns också implementerade.

6.2.2 Tillagda sensorinteraktioner

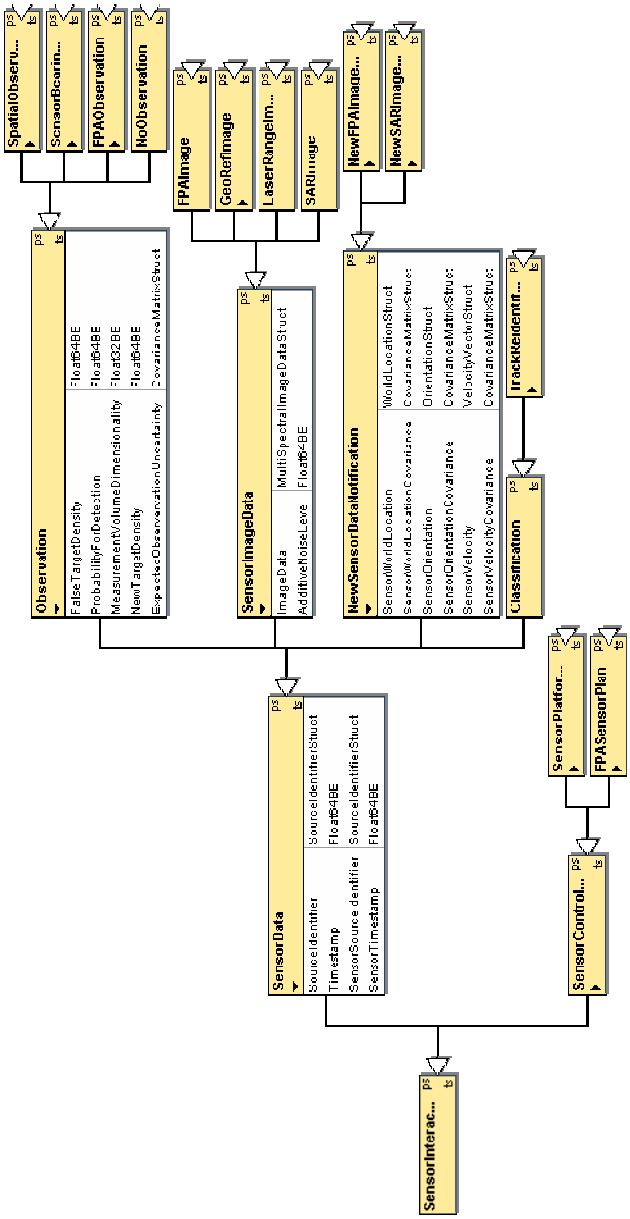
För att det ska vara möjligt att distribuera sensorsimuleringens olika delar på flera datorer krävs en flexibel och lätthanterad kommunikation mellan de olika federaterna. För detta används i princip uteslutande HLAs interaktionsmekanism. Denna medger kommunikation av typen ”från en till många” genom ett prenumereringsförfarande, vilket underlättar implementeringen av federaterna och ökar flexibiliteten vid olika sensorkonfigurationer. Att skicka större datamängder över HLA är inte optimalt ur prestandahänseende, men med tanke på att de avancerade sensorsimuleringar och signalbehandlings- och planeringsmetoder som är tänkt att används i denna simuleringsmiljö, så är realtidsprestanda ändå nedprioriterat.

”SensorInteraction”-grenen i FOM:en innehåller interaktioner för att kommunicera data till och från sensorerna och mellan olika steg i signalbehandlingskedjan, se figur 42. Exempelvis så används interaktionen ”FPAImage” för att skicka IR- eller visuella bilder från SceneServer, avsnitt 4.3, till detektionsalgoritmen, avsnitt 5.1 och återigenkänningsalgoritmen, avsnitt 5.3.

Ett lite speciellt designbeslut är att använda en extra interaktion för att publicera sensorns parameteruppsättning för en viss sensordatainteraktion, som en separat interaktion, t.ex. ”NewFPAImageNotification” som finns under ”NewSensorDataNotification”. Detta görs för att federater som ligger en bit bak i signalbehandlingskedjan ska få tillgång till sensorparametrar utan att behöva ta emot rådata, vilket skulle innebära mycket onödig nätverkstrafik. En annan anledning är att federater kan få reda på att en ny bild genererats och därmed förbereda för processning av nytt data. Ett bra exempel på detta är målföljaren, som kan beräkna och publicera målspårprediktioner till exempelvis en detektionsalgoritm eller presentationssystem utan att dessa behöver skicka en förfrågan om detta. Målföljaren behöver dessutom informationen om sensorns parametrar när den ska ta hand om detektioner från detektorn och uppdatera målspåren.

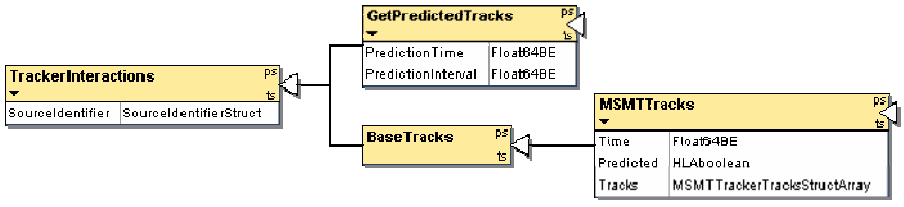
Sensorobservationsgrenen, ”Observation”, har även en speciell interaktion för att signalera avsaknad av observationer i sensordata, ”NoObservation”. Denna interaktion är nödvändig att inkludera eftersom informationen om tillgängligheten av ny sensordata publiceras till alla federater som är intresserade av detta. Dessa måste då också få information från t.ex. en detektionsalgoritm att det inte fanns några detektioner i data, för att slippa besvärliga specialfall i federaterna.

”SensorControlPlan”-grenen innehåller olika nivåer av sensorstyrnings- och sensorplattformstyrningsinteraktioner. Dessa skickas från en planeringsfederat, avsnitt 5.4, till en särskild entitethanteringsfederat som utför styrplanen koordinerat över tiden.



Figur 42: ”SensorInteraction”-grenen är tillagd direkt under ”HLAinteractionRoot” och innehåller interaktioner för att kommunicera data mellan olika federater som implementerar olika delar av en signalbehandlingskedja i ett sensorssystem.

Figur 43 visar den särskilda gren för målföljningsinteraktioner som implementerats. Den har i nuläget endast utförligt implementerad funktionalitet för MSMT-målföljaren som beskrivits i avsnitt 5.2.

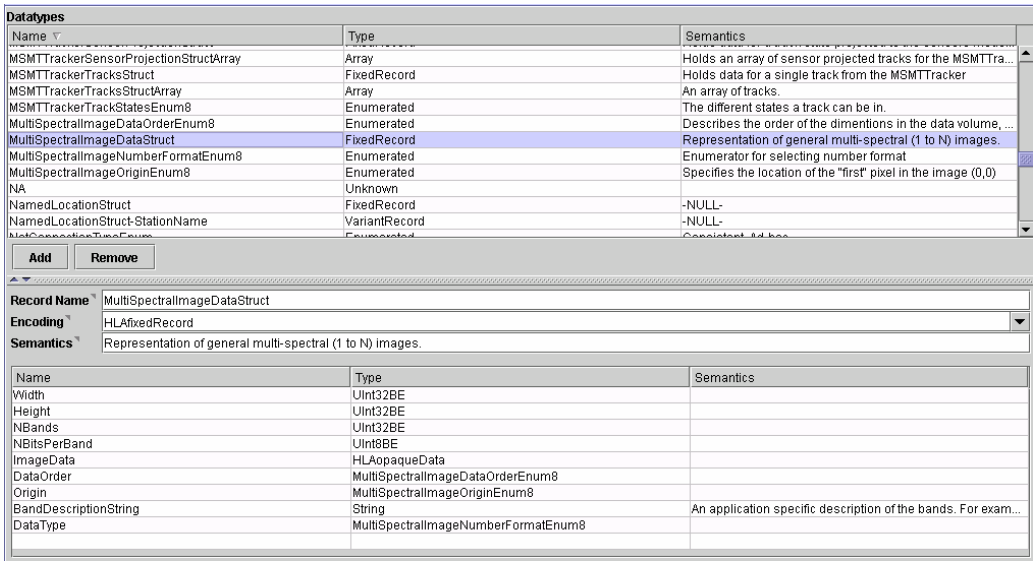


Figur 43: En särskild gren direkt under ”HLAinteractionRoot” har skapats för målföljningsinteraktioner.

6.2.3 Datatyper

Till varje objekt och interaktion finns en uppsättning attribut med mer eller mindre komplexa datastrukturer. De enklaste utgörs av primitiva datatyper såsom heltal eller flyttal med varierande kapacitet och precision. Arrayer av godtyckliga datatyper finns definierade i HLA-standarden.

Till de mer komplexa datatyperna hör t.ex. ”MultiSpectralImageDataStruct”, som kodar en multispektral bild med information om datatyp för enskilda pixlar, bildstorlek, bildorigo m.m., se figur 44. I figuren syns också några ”MSMTTracker”-datatyper.



Figur 44: Ett exempel på hur en datatyp är definierad i FOM:en.

6.2.4 JAVA- och C++-implementering av FOM-tillägg

Definieringen av FOM:en är oberoende av vilket programmeringsspråk som federaterna sedan är implementerade med. Alla objekt, interaktioner och datatyper måste även implementeras i de programmeringsspråk man avser använda. De språk som MSSLab använder är JAVA och C++, JAVA integrerar sedan även Matlab på ett rättframt sätt.

Allt arbete med implementering av RPR-FOM:en har redan gjorts i MOSART, vilket underlättat enormt mycket vid utvecklingen av MSSLab. Alla tillägg i FOM:en som gjorts har i första hand implementerats i JAVA för att säkerställa kopplingen till Matlab. C++ implementationen har till stora delar också färdigställts.

Federater för alla delar som beskrivs i avsnitt 4.1-4.4 och 5.1-5.4 har implementerats i Matlab via Matlabs inbyggda JAVA-koppling. Dessutom har federater för fordon och andra entiteter, samt en del enklare federater för utvecklings- och testsyfte implementerats.

6.3 Fortsatt arbete

Arbetet med att formulera tillägg till FOM:en och att implementera dessa är till de största delarna klara. Delar som kräver fortsatt arbete är t.ex. konfigurering och dynamisk förändring av väder, funktionalitet för konfigurering av federater

från t.ex. NetScene vid uppstart, automatisk distribuering av federater över flera datorer m.m. CameoSim har varit integrerad i MSSLab via en fedrat, men efter den stora uppgradering av FOM:en som gjorts i år, har CameoSim-federaten inte hunnits med att uppgradera. CameoSim är dessutom något besvärlig att integrera på grund av dess stängda filformat m.m.

Möjlighet att köra flera snarlika simuleringar med olika variationer av parametrar (batchkörningar) etc. behöver införas. I nuläget körs alla scenarier för hand, vilket inte möjliggör effektiv generering av dataunderlag för större sensor- eller systemvärderingsuppdrag.

Högt på listan över saker att förbättra i hanteringen av simuleringarna är även att möjliggöra för federater att ansluta till scenarier efter det att federationshanteraren startat simuleringen.

SceneServer federaten börjar uppnå en sådan mognad att det är dags att implementera den i JAVA för att erhålla bättre prestanda. Den är i dagsläget onödigt långsam och sinkar simuleringar av enklare optiska system.

En genomgripande översyn av dataloggningsförfarandet i federaterna behöver också ske. Förmåga att spela upp loggat data i federater på olika ställen i signalbehandlingskedjan behövs för att underlätta utveckling av federater med många beroenden till andra federater.

7 Resultat

7.1 Status på utvecklingen av simuleringsbaserade metoder för sensorvärdering

Projektet SIMSENS (Simuleringsbaserade metoder för sensorvärdering) har i uppgift att utveckla en gemensam simuleringsmiljö för värdering av sensorsystem i olika miljöer, väderförhållanden och tidpunkter. Simuleringsverktyget som vidareutvecklas för detta är MSSLab (MultiSensorSimuleringsLab). Denna rapport har beskrivit utvecklingen och aktuell status för de olika delarna i simuleringsverktyget.

För att en korrekt sensorvärdering skall erhållas med simuleringsbaserade metoder krävs noggranna simuleringsmodeller. Under året har SIMSENS-projektet förbättrat olika simuleringsmodeller för sensor, terräng, objekt och väder. En stor del av årets verksamhet har varit att ta fram atmosfärsmodeller för de fyra väderfallen: vackert väder, regn, moln och snöfall. Atmosfärsmodellerna som tagits fram kan numera ses som noggranna och är inte gränssättande för noggrannheten vid vädersimuleringar av ett specifikt väderfall.

För IR och visuella simuleringar har beräkningsprogrammet CameoSim använts. För att beräkna de atmosfärsfiler som CameoSim använder vid simuleringen har programmet MODTRAN använts. Inledande simuleringar visar att framförallt moln kraftigt degraderar IR-prestanda.

För radarsimuleringar används beräkningsprogrammet SE-RAY-EM. Detta program har nu kompletterats med funktioner för att ta hänsyn till både dämpning och spridning för de olika väderfallen. En grov tumregel för storleksförhållandena hos utbredningsförlusterna i atmosfären för radarvågen är i minskande storleksordning: smältande snö, regn, moln och dimma, samt slutligen torr snö.

Arbete har genomförts under året för att utveckla metoder att automatiskt extrahera och texturera byggnader utgående från data om byggnadernas byggytor och ortorektifierade flygfoton. Dessa metoder har använts för att automatiskt rekonstruera byggnader i Norrköpingsmodellen.

För att göra det möjligt att simulera sensordata med animerade människor har HLAS, High Level Animation System, tagits fram. Med HLAS underlättas integrationen av animerade sekvenser för olika typer av simuleringsramverk. Med HLAS är det nu möjligt att styra animeringssekvenser på en högre abstraktionsnivå. Ett exempel på ett sådant fall kan vara att en användare vill förflytta en karaktär från en punkt till en annan utan att behöva ta hänsyn till hur terrängen ser ut.

Simuleringsverktyget har under 2008 använts av flera projekt för sensorsimuleringar. Några exempel är ARCUS, Autonoma funktioner, Ledningssystem - demonstration och värdering av teknik och metodik (LedDoV), SE-MARK och Taktikanpassning SAT.

Under året har projektet bidragit med underlag till ett antal Försvarsmaktsstudier. Projektet har delvis stöttat studierna ekonomiskt inom ramen för vad FoT-

projekt är ålagda att göra, men framförallt genom att bidra med den kompetens som byggts upp inom detta och andra pågående projekt. Tekniker och kompetenser som utnyttjats omfattar sensorkunskap och därmed tillhörande signalbehandling inom IR, laser och radarområdet.

7.2 Fortsatt arbete

MSSLab har redan idag mycket bra funktionalitet, men mycket arbete återstår för att en generell simuleringsmiljö skall erhållas. Den snabba teknikutvecklingen inom simuleringsområdet möjliggör ny funktionalitet och förbättrad sensorsignalkvalité.

Tillgängliga omvärldsmodeller har mycket god kvalité, men de bör förbättras och kompletteras. Idag är det bara Kvarnmodellen som har olika texturer såsom IR, visuell och materialklassad textur. Norrköpingsmodellen har endast visuell textur och bör kompletteras med en materialklassad modell för att scensimuleringsverktygen SE-RAY-EM och CameoSim ska kunna användas. För att göra det möjligt att värdera t.ex. ett spaningssystem i internationella miljöer bör de nuvarande modellerna kompletteras med sådana terrängmodeller.

För att göra det möjligt att simulera sensordata från animerade människor kommer HLAS att vidareutvecklas. Renderingsimplementationen för HLAS bygger i dagsläget på OpenGL och denna bör integreras mot OpenSceneGraph för att göra det möjligt att simulera människor i SceneServer och 3D-lasermodellen. Arbete bör även inriktas mot att simulera sensordata med grupper av animerade människor.

De genomförda IR-simuleringarna i olika väderfall behöver valideras ytterligare. En metod för detta är att jämföra de parametriska atmosfärerna som räknats fram i CameoSim via MODTRAN med körningar direkt i MODTRAN, samt jämföra med teoretiska beräkningar direkt från storheternas definitioner. Kontraster i CameoSim-bilder kan även jämföras med mätningar.

En begränsning med MODTRAN är att molnen i modellen täcker hela himlen. Det går alltså inte att ändra molnens täckningsgrad på himlen. Ett sätt att lösa detta kan därför vara att generera egna modeller av moln och placera ut dessa som objekt i scenen som skall simuleras. En annan begränsning i MODTRAN är att det inte finns modeller för snöfall och hagel.

Atmosfärsmodellerna för radar är mycket bra, men en noggrannare modellering av snöfall krävs för att få med inverkan vid högre frekvenser och för delvis smälta flingor, där det är känt att dämpningen kan överstiga den i regn, för samma ekvivalenta vattenintensitet.

Väderpåverkan på simuleringen av sensordata har begränsats till atmosfärseffekter. Förändringar av objektens och bakgrundens signatur för de olika väderfallen har därför inte tagits med. Detta måste åtgärdas i det fortsatta arbetet.

8 Referenser

[1] Ahlberg S., Elmqvist M., Persson Å., Söderman U., “Metoder för framställning av högupplösta syntetiska omgivningar / omvärldsmodeller för sensorsimulering”, Metodrapport, FOI-R--1110--SE, Linköping, 2003.

[2] Söderman U., Ahlberg S., Persson Å., Tolt G, ”Omvärldsmodellering/Dynamisk terräng - slutrapport”, FOI-R--2170--SE, 2006.

[3] Internet: http://www.got3d.com, besökt 2008-12-01.

[4] Persson A., Perzon S., Halling C., "Ground Vehicle Modelling Using RadThermIR – Progress report 2006", FOI-R-2200—SE, Technical Report, December 2006.

[5] Internet: http://home.gna.org/cal3d

[6] Internet: http://www.opengl.org

[7] Internet: http://www.openscenegraph.org/projects/osg, besökt 2008-12-09.

[8] Internet: http://www.oktal-se.fr/ENGLISH/site/sommaire.htm, besökt 2004-11-29.

[9] Näsström F. Carlsson L., Forsgren R., Chevalier T., Gustafsson O., Karlsson N., Sume A., ”Integration av atmosfärsmodeller i MSSLab”, FOI-R--2521--SE, Juni 2008.

[10] Kneizy F. X., Abreu L. W., Anderson G. P., Chetwynd J. H., Shettle E. P., Berk A. Bernstein L.S., Robertson D. C., Acharya P., Rothman L. S., Selby J. E. A., Gallery W. O., Clough S. A.: “The MODTRAN 2/3 Report and LOWTRAN 7 Model”. Phillips Laboratory, Geophysics Directorate, PL/GPOS, Hanscom AFB, MA 01731-3010, USA . 1996 (Rapporten finns i PDF format på CD-ROM distributionen av PcModWin4.0.

[11] Bennet F. och Fenelius S., "SceneServer - a 3D software assisting developers of computer vision algorithms", FOI. Linköping, Technical report, FOI-R--0831--SE, March 2003.

[12] Internet: http://www.openscenegraph.org, besökt 2007-12-16.

[13] Chevalier T., ”Statusrapport Modellering av 3D-avbildande laser”, FOI MEMO 2417, Maj 2008.

[14] Karlholm J., ”Implementation and evaluation of a method for detection of ground targets in aerial EO/IR imagery”, Linköping, FOI, Scientific report, FOI-R--1267--SE, ISSN 1650-1942, Juni 2004.

[15] Karlholm J., Rydell J., Stenborg K-G., Ulvklo M., Nygårds J., Hemström F., ”Reidentification of ground targets in EO/IR-imagery”, FOI-R--2586--SE, ISSN 1650-1942, October 2008.

[16] Chang P. och Krum J., ”Object recognition with color cooccurrence histograms”, IEEE Conference on Computer Vision and Pattern Recognition, 1999.

[17] Skoglar P., Nygårds J., Björström R., Ögren P., Hamberg J., Hermansson P., Ulvklo M., ”Path and Sensor Framework Applicable to UAV Surveillance with EO/IR sensors”, FOI-R--1741--SE, september 2005.

[18] IEEE Std 1516-2000. ”IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)”, Institute of Electrical and Electronics Engineers, Inc (IEEE), 2000.

[19] RPR-FOM Version 1.0 SISO-STD-001.1-1999, Simulation Interoperability Standards Organization Inc., 1999.

[20] Internet: http://www.pitch.se, besökt 2008-11-25.

[21] Horney T., Brännström M., Bergman J., Andersson D., Forsgren R., Ahlberg S., Lindahl B., Törne A., ”Slutrapport MOSART”, FOI-R--1814--SE, December 2005.

[22] MOSART:s hemsida (http://www.foi.se/MOSART/)