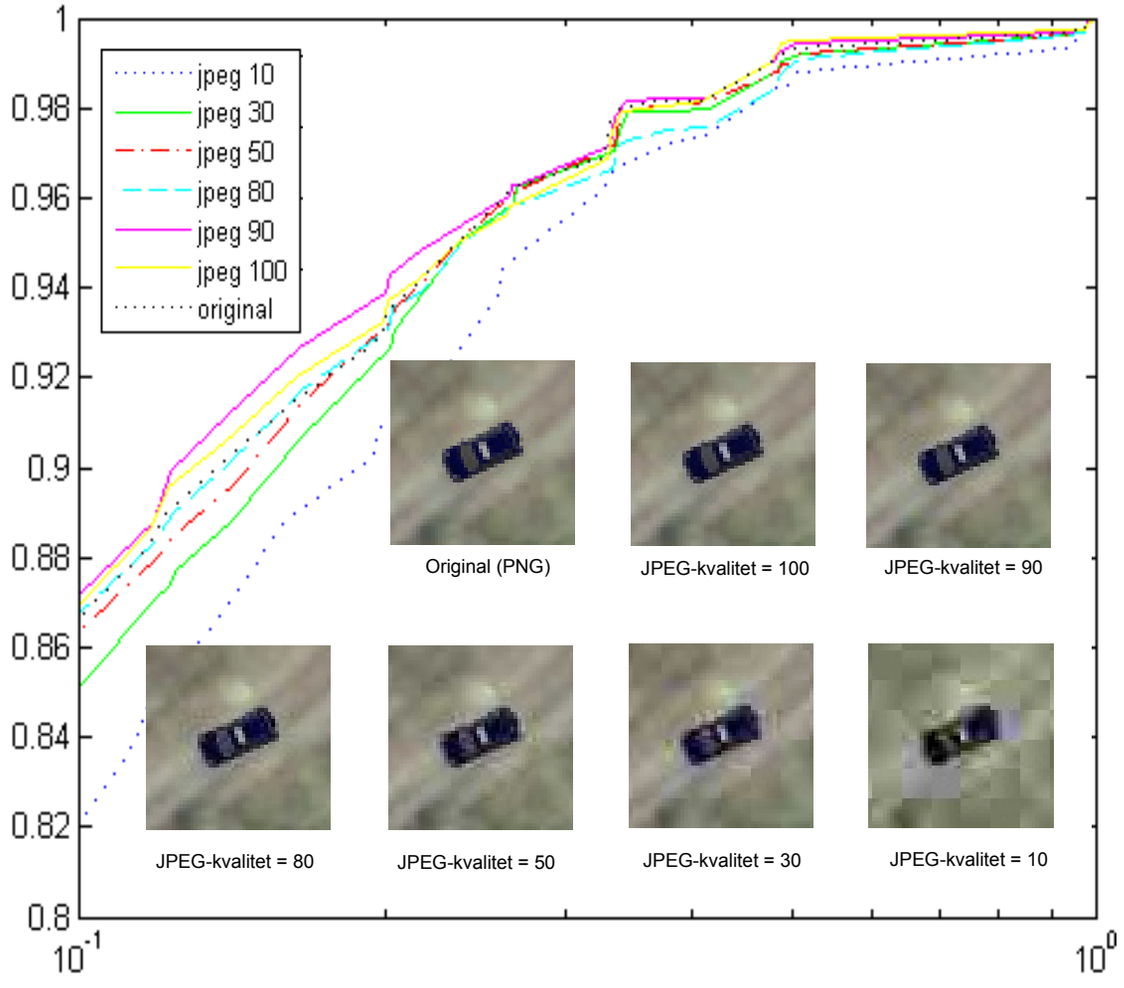
Datakomprimering i sensorsimuleringsmiljö

K-G STENBORG, JONAS ALLVAR,
GUSTAV HAAPALAHTI, FREDRIK NÄSSTRÖM



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--3113--SE
ISSN 1650-1942

Teknisk rapport
December 2010

Informationssystem

K-G Stenborg, Jonas Allvar, Gustav Haapalahti,
Fredrik Näsström

Datakomprimering i sensorsimuleringsmiljö

Titel	Datakomprimering i sensorsimuleringsmiljö
Title	Data compression in sensor simulation environment
Rapportnr/Report no	FOI-R--3113--SE
Rapporttyp Report Type	Teknisk rapport
Månad/Month	December
Utgivningsår/Year	2010
Antal sidor/Pages	36 p
ISSN	ISSN 1650-1942
Kund/Customer	FM
Projektnr/Project no	E53188
Godkänd av/Approved by	Lars Bohman

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

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

Sammanfattning

Beroende på storleken på lagringsutrymme eller begränsad överföringskapacitet behövs ibland sensordata komprimeras. Det finns icke-förstörande (lossless) komprimering, som inte är så effektiv, och förstörande komprimering (lossy) som är mer effektiv, men inför distorsion i sensordata. Ett sätt att undersöka hur sådan distorsion påverkar automatiska signalbehandlingsmetoder är att testa dem i en simuleringsmiljö.

Detta projekt har undersökt hur komprimering kan användas i simuleringsmiljön MSSLab. Dessutom har en undersökning genomförts av hur en automatisk signalbehandlingsmetod bäst skall vara inställd, för att fungera på sensordata som blivit utsatt för förstörande komprimering.

En förstörande datakomprimeringsmetod (JPEG) har tillförts till MSSLab och en undersökning har genomförts av dess inverkan på en automatisk detektionsmetod som bygger på klassificering av särdrag. Undersökningen har tittat på huruvida detektorn skall tränas på komprimerade bilder eller på icke-komprimerade bilder, för att uppnå så bra resultat som möjligt.

Utgående från bilder med sju olika grader av komprimering, har ROC-kurvor tagits fram för att undersöka detektionsalgoritmen. Ingen eller låg komprimering ger ofta bra detektionsresultat, men ibland kan hårt komprimerade bilder ge bättre resultat. Det verkar som att komprimeringen då fungerar som en förfiltrering som framhäver egenskaper som detektorn har nytta av. Därav är det svårt att entydigt säga hur graden av komprimering påverkar den valda detektionsalgoritmen.

Nyckelord: Datakomprimering, JPEG, MPEG, Simulering, Sensorer

Summary

Depending on the size of storage space or limited transmission capacity compression of sensor data is sometimes needed. There exist non-destructive (lossless) compression methods but lossy compression methods are more efficient. These lossy methods will distort the sensor data. One way to examine how this distortion affects automatic signal processing methods is to test them in a simulation environment.

This project has investigated how compression can be used in the simulation environment MSSLab. In addition, a study has been made on how an automatic signal processing method handles sensor data that has been subjected to lossy compression.

A lossy data compression methods (JPEG) has been added to MSSLab and its impact on a detector have been investigated. Whether the detector must be trained on the compressed images or on the original images in order to achieve the best possible outcome has been investigated.

Based on images with seven different degrees of compression, a number of ROC curves have been developed to investigate the detection algorithm. No or low compression often produces good detection results, but sometimes a heavily compressed image can give even better results. It seems that the compression when acts as a pre-filtering that highlights properties that the detector can benefit from. Hence, it is difficult to clearly say how the degree of compression affects the selected detection algorithm.

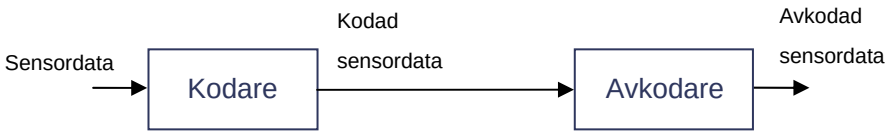
Keywords: Data Compression, JPEG, MPEG, Simulation, Sensors

Innehållsförteckning

1 Inledning

Det blir mer och mer vanligt att signal- och bildbehandlingsalgoritmer används på sensordata för att automatisk få fram information. Denna typ av information kan exempelvis vara detektioner av objekt som människor eller fordon, igenkänning av signatur för databaslagrade objekt samt återigenkänning av tidigare observerade objekt.

Oftast är det en stor datamängd som samlas in av sensorerna. För lagring eller överföring krävs då komprimering av sensordata. Komprimeringen består av en kodare som komprimerar data till ett nytt format och för att använda komprimerad data måste den sedan genomgå en avkodare som packar upp sensordata igen, figur 1.



Figur 1: Sensordata komprimeras i en kodare. För att kunna läsa den komprimerade senosrdatan måste den packas upp i en avkodare.

Det finns icke-förstörande komprimering (lossless) som inte inför någon distorsion, men sådana metoder leder inte till speciellt stor komprimering. Därför är förstörande komprimering (lossy) vanlig i de flesta tillämpningar. Fast ju högre komprimeringsfaktor som nås desto mer distorsion införs då på sensordata, av komprimeringsalgoritmen. För bilder och video är denna komprimeringsdistorsion anpassad efter det mänskliga synsinnet så att vi inte skall notera kodningsartefakterna.

De flesta kodningsalgoritmer är anpassade till oss mänskor på så sätt att de kodningsartefakter som algoritmerna ger upphov till är svåra att uppfatta för det mänskliga synsinnet. Det är inte säkert att dessa artefakter är lämpade för automatiska bildbehandlingsmetoder. Dessa kanske reagerar på andra saker i sensordata än de detaljer som vi ser.

Vid signal- och bildbehadlingsforskning tas det sällan hänsyn till komprimering. Om det går arbetar man med icke komprimerad data, men om det inte finns att tillgå (exempelvis är det sällan möjligt att få helt icke-komprimerad video från konsumentkameror), används komprimerad data utan att undersöka hur kodningsartefakterna kan tänkas påverka forskningsresultaten.

Vi har tidigare undersökt hur komprimering av olika former av sensordata påverkar olika algoritmer [1]. Då undersöktes bara verkliga sensordata vilket i sig har många fördelar. Simulerade sensordata har dock andra fördelar, som att det går skapa stora mängder data som är annoterat. Sådana sensordata kan användas vid träning av boostingbaserade algoritmer, som detektorn som vi har undersökt i denna studie.

Arbetet har bestått av två delar. Först att implementera möjlighet för att använda komprimerad simulerad senordata i MSSLab. I den andra delen har en studie av hur en automatisk bildbehandlingsalgoritm i form av en detektor på olika sätt påverkas av att sensordata innehåller komprimeringsdistorsion.

2 Implementering i MSSLab

En av projektets två delar var att bygga ut MSSLab för att det skall kunna hantera komprimerad sensordata.

2.1 MSSLab

MultiSensorSimuleringsLab (MSSLab) [2] är ett simuleringslaboratorium för simulering av avancerade sensorsystem. Med MSSLab kan olika sensorsystem värderas i olika miljöer och vid olika väderförhållanden och tidpunkter. Ramverket möjliggör simulering av dynamiska scenarier med rörliga plattformar, sensorbärare, målobjekt m.m. Högkvalitativa sensorsimuleringar kan göras med IR, visuellt, laser och radar och dessa sensorsystem kan simuleras både enskilt och som multisensorsystem. Avancerade signalbehandlingsmetoder, följealgoritmer och datafusionsmetoder är även integrerade i simuleringsverktyget.

Vid avdelningen för Sensorsystem har genom åren flera olika sensorsimuleringsverktyg införskaffats eller utvecklats. Avancerade algoritmer för objektdetektion, objektklassificering och följning har också tagits fram. Integreringen av sensorsimuleringsprogram med avancerade signalbehandlingsalgoritmer ger nu helt nya möjligheter till algoritmutveckling och utvärdering av hela sensorsystem i mycket realistiska signalmiljöer och scenarier.

MSSLab har hittills använts för simulering av markscenarier, men även sjö- och luftscenarier kan simuleras. MSSLab används även till att ta fram dataunderlag för signal- och bildbehandlingsalgoritmer. Verktyget kan även användas för att validera förenklade sensormodeller som används för mängd- eller interaktiva simuleringar. Sensorsimuleringsprogrammen som har integrerats är radarsimuleringsprogrammet SE-RAY-EM (tidigare kallat SPECRAY EM), IR-simuleringsprogrammet CameoSim, en egenutvecklad 3D-lasersensormodell, samt simuleringsverktyget SceneServer för måttligt realistiska IR-simulering och för visuell presentation av scenariot.

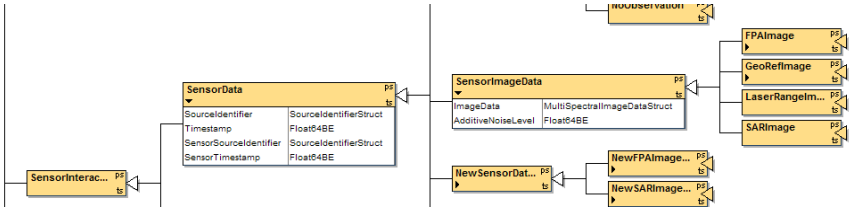
2.2 Komprimerad sensordata i HLA

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. 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 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. En standard FOM har definierats upp av standardiseringskommittén för HLA. Denna FOM kallas RPR-FOM.

MOSART var ett försvarsmaktsprojekt som pågick på FOI för några år sedan [3]. I detta projekt implementerades en stor kodbas för HLA och RPR-FOM:en. Genom Simsens projektet har MOSART integrerats i MSSLab. RPR-FOM har dessutom utökats [2] med HLA-objekt och interaktioner för att underlätta sensorsimuleringar. Figur 2 visar några av de utökningar som gjorts för sensordata och sensorbilddata. Det är i parametern ”ImageData” i interaktionen ”SensorImageData” som sensordata och bilddata lagras för en sensors sampel.

För att stödja komprimering av bilddata så har fältet ”CompressionMethod” lagts till. Innebörden av fältet är att indikera vilken komprimeringsmetod som använts för det data som skickas och tas emot. Den kan sättas till något av följande värden: ”None”, ”PNG”, och ”JPEG”. Dessutom har fältet ”CompressionParamList” lagts till där det finns en möjlighet att lägga till valfria nyckel/värde. Det kan användas här till att skicka med komprimeringskvalitet på JPEG-komprimeringen.



Figur 2: Arvshierarkin för sensorinteraktion och sensorbilddata.

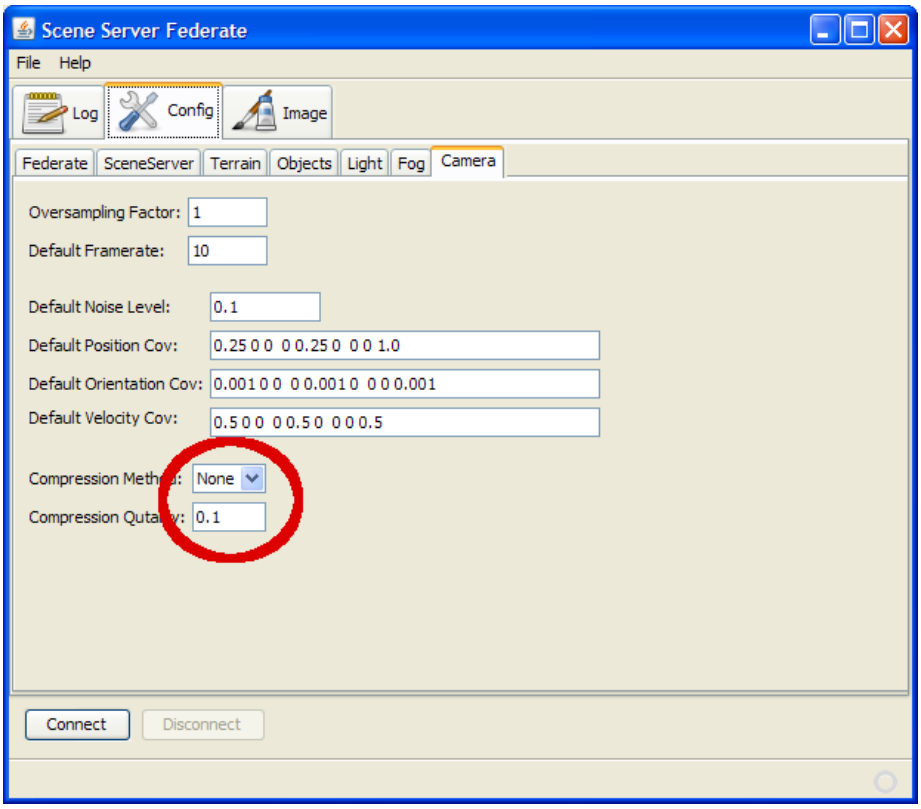
För att förenkla och automatisera hanteringen av komprimerad sensordata så har MOSARTs javaimplementation utökats. Detta med hjälp av javas inbyggda stöd för att spara och ladda komprimerade rasterbilder (ImageIO). I funktionen som hämtar ut bilddata (”getImageData”) så avkodas automatiskt bilden om ”CompressionMethod” är satt till något annat än ”None”. För att komprimera en bild har hjälpfunktionen ”compressData” lagts till.

2.3 Implementering SceneServerFederate

Förenklade simuleringar av visuella och IR-sensorer kan göras med SceneServer. SceneServer [4] är ett program som utvecklats av FOI för realtidsvisualisering av syntetiska omgivningar. Integrationen av SceneServer i MSSLab är gjort via javaprogrammet SceneServerFederate. Den använder MOSART för att

distribuera visuella och IR-sensorbilder till bland annat de detektionsfederater som finns implementerade i MSSLab.

För att enkelt kunna studera effekterna av komprimerade sensorbilder så har nya stödet i MOSART integrerats i SceneServerFederaten. Under fliken ”Camera” finns inställningar för att välja vilken komprimeringsmetod och eventuell kvalitet som ska användas för de bilder som SceneServer genererar, figur 3. I dagsläget är det är bara för JPEG som kvaliteten används och då är det ett värde mellan 0 och 1 där 0 är max komprimering och 1 minst komprimering.



Figur 3: Fliken Camera i SceneServerFederate används för att ställa in komprimering. Möjliga val på metod är ”None”, ”PNG” och ”JPEG” där det för den sistnämnda även går fylla i kvalité.

En fördel som man kan se med komprimerade sensorbilder i MSSLab, förutom att simulera effekterna, är att mängden data som ska distribueras minskar drastiskt. Om simuleringen är distribuerad mellan flera datorer och nätverket är långsamt så kommer det minska på körningstiden för simuleringarna.

3 Detektion på komprimerade bilder

Vi har undersökt hur en detektionsalgoritm påverkas av att arbeta på komprimerade sensordata. Detektorn finns installerad i MSSLab och resultaten från denna studie kan därför enkelt användas vid simuleringar mellan detektorfederaten och andra federater.

3.1 Detektor

Den algoritm som vi använt är skapad för att detektera fordon från simulerad visuell (RGB) data tagna av en UAV [5]. Genom SceneServer [4] placeras ett antal modeller av fordon ut i en modell av Norrköping [6]. I denna simulerade miljö finns sedan en virtuell kamera tänkt att sitta på en UAV som tar stillbilder av marken och markfordonen. Detektorn avgör automatiskt om det finns markfordon i bilden och var dessa i så fall är placerade. Algoritmen känner alltså inte igen olika fordon utan avgör bara huruvida det finns ett fordon där eller inte.

Denna visuella detektor bygger på boosting av olika särdrag [5]. Genom att träna på små bildpatchar (figur 4) med fordon respektive utan fordon, lär sig detektorn vilka särdrag som tillsammans ger kunskap om huruvida bildpatchen innehåller ett fordon eller inte. När sedan särdragsparametrarna har ställts in i träningssteget genom boostingen, genomförs en evaluering av resultatet. Denna test genomförs på liknande bildpatchar men det är viktigt att inte exakt samma patchar används för både träning och test. Testet resulterar i en ROC-kurva [7] som ger en uppfattning om hur stor andel fel som den givna träningen ger. Det finns då två typer av fel:

- Bildpatchar med fordon som inte detekteras.
- Bildpatchar utan fordon som ändå ger utslag i detektorn (falsklarm).

Genom ROC-kurvan går det utläsa hur fördelningen mellan dessa båda typer av fel förhåller sig för detektorn.

Fordonen är alltid placerade i mitten av bildpatcharna. Däremot bör fordonen placeras ut med olika rotationer för att algoritmen skall vara robust oberoende av fordonets rotation. På samma sätt bör kamerans vinkel ner mot marken förändras för att klara av olika betraktningsvinklar. Kamerans avstånd till marken kan dessutom förändras.

I våra försök har vi valt att inte använda alla dessa inställningar eftersom det skulle göra både träning och evaluering långsam. Därför befinner sig alltid kameran på samma avstånd till marken. Kameran tittar alltid i det närmaste rakt ner, med en vinkel på 1, 5 respektive 10 grader mot markens normal. Lika många bildpatchar har använts från dessa tre betraktningsvinklar. Fordonens rotation varierar bara mellan 25 och 65 grader (i intervall om 5 grader) istället för mellan

0 och 360 grader. Fordonen är alltså roterade 25, 30, 35, 40, 45, 50, 55, 60 och 65 grader i förhållande till betraktningsriktningen. Både träningen och testningen använder likvärdig fördelningen mellan dessa olika rotationer. En faktor som däremot är helt slumpmässig är styrkan på belysningen i den simulerade scenen. Ljusstyrkan variera därför mellan olika bildpatchar så att detektorn skall bli robust mot ljusförändringar.

Antalet olika fordonsmodeller är 23. Fast då är några av dessa modeller samma typ av fordon (samma form) men med olika färg eller textur.

Figur 4: Åtta olika bildpatchar. (a)-(d) är tagna med samma vinkel på fordonen, 25 grader. (e)-(f) är samma fordon i 65 grader men med infallsvinkel 1 respektive 10. Här syns dessutom den slumpmässiga beslysningen som i (g) är väldigt ljus. (h) visar slutgiltligen att fordonen inte alltid är placerad på väg eller gräs utan ibland kan hamna på byggnader.

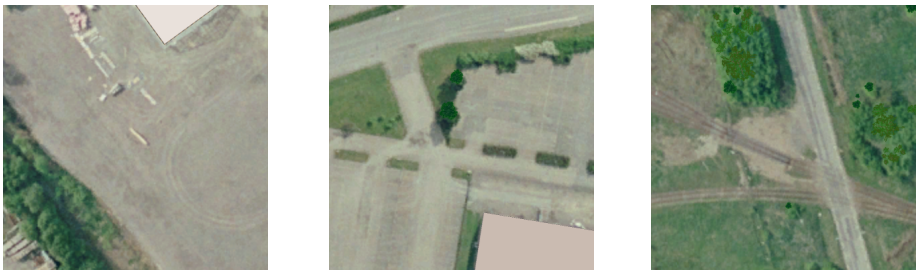
Vi har tagit fram fem bildpatchar på varje fordon för varje vinkel (men med olika mark under fordonet). Av dessa används tre till träningen och två till testen. Med andra ord har vi tränat på alla tillgängliga typer av fordon och på samma sätt ingår alla fordonstyper i testen. Sammanlagt blir det då 1863 bildpatchar vid träning och 1242 bildpatchar vid test. Eftersom vi varierat de olika vinklarna så lite behövs inte fler bildpatchar. Skulle vi däremot skapa en detektor som är robust mot fler rotationer på fordonen och flera betraktningsvinklar behöver antalet bildpatchar i träningen utökas.

Bildpatcharna är av storlek 50 x 50 pixlar och fordonen är omkring 25 pixlar långa (beroende på om det är stora eller små fordon). I figur 4 visas några sådana bildpatchar.

De negativa bildpatchar som används i träningen och testningen innehåller inga fordon. Dessa bildpatchar är också av storlek 50 x 50 pixlar och skapas utgående från större bilder efter filtrering av en mask, figur 5.

Som särdrag har vi valt att använda Viola & Jones [8]. Egentligen borde fler typer av särdrag användas och utvärderas för hur de påverkas av kompression, men i denna lilla studie finns det inte utrymme att undersöka fler.

Vi är väl medvetna om att detektorn brukar fungera bättre på IR-bilder än vanliga visuella bilder. Om fordonen är varma är de nämligen lättare att urskilja mot bakgrunden i IR, än för visuella fallet. Valet att använda vanliga visuella färgbilder beror på att vi tror att den påverkan som kompressionen medför kan noteras tydligare här. Så för IR kommer detektorn troligtvis vara mer robust för komprimering.



rav4i0_r25-65_i1_r65_p5.png

rav4i0_r25-65_i10_r65_p2.png

v70_defaulti0_r25-65_i1_r35_p4.png

Figur 5: Tre negativa bilder. Genom att ta fram slumpmässiga delmängder från dem på 50 x 50 pixlar skapas negativa bildpatchar som används i träning och testning. För att försäkra sig om att inga riktiga fordon skall finnas på dessa bilder används kontrolleras de negativa bildpatcharna mot en mask som innehåller information om huruvida det är en lämplig negativ bild eller ej.

3.2 Komprimering

Det finns många typer av datakomprimering. Vi intresserar oss här för förstörande komprimering (lossy compression) som inför distorsion i bilderna när de komprimeras [9][10]. Eftersom vi arbetat med bilder har vi använt JPEG-komprimering [11], som är den absolut vanligaste typen av kompression för

stillbilder. De artefakter som JPEG ger upphov till är av samma typ som den distorsion som videokomprimering skapar. Därför skulle resultaten inte bli så mycket annorlunda om dessa bilder istället hade tagits från en ström av komprimerade videobilder.

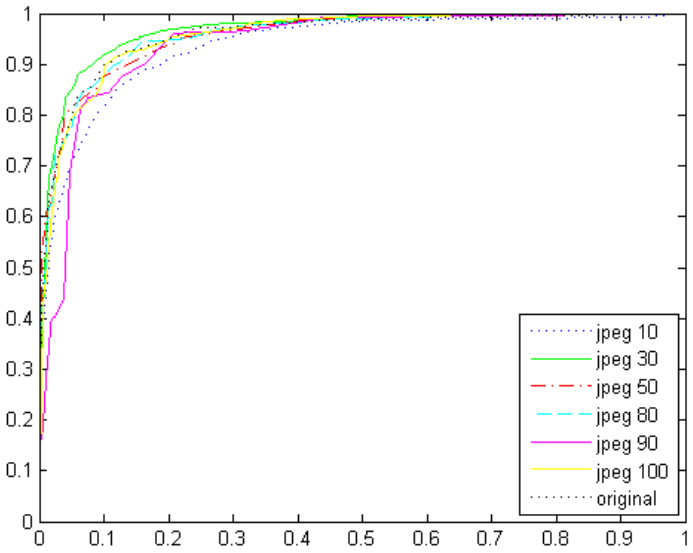
Distorsion från JPEG är allra mest tydlig i de block som skapas. Se exempelvis blocken för sista bildpatchen i figur 6. Dessa block om 8 x 8 pixlar blir tydliga eftersom JPEG bygger på transformkodning [12] där just sådana block transformeras var för sig. Efter transformeringen kvantiseras de olika frekvenskomponeterna olika beroende på om de anses innehålla värdefull information eller ej. För vissa ointressanta frekvenser sparas ingen information alls. Denna kvantisering av frekvenserna gör att exempelvis kanter (skarpa skillnader i färg eller ljusstyrka) i bilden kan bli utsmetade. Vågmönster kan då uppstå omkring kanter, se återigen figur 6. Sammanlagt kan alltså distorsion från JPEG-komprimering ge upphåll till kanter mellan olika block där det ursprungligen inte fanns några kanter, utsmetade färger där det tidigare var skarpa kanter och texturer från vågmönstren där det tidigare bara fanns svaga förändringar.

Figur 6: Samma bildpatch med olika grad av komprimering. (a) är på 100 x 100 pixlar och används för att skapa bildpatcharna (b)-(h) som är på 50 x 50 pixlar.

Det finns flera sätt att ställa in vilket typ av kvalité man vill ha på sin JPEG-komprimering. Vi har valt att använda den skala mellan 0 och 100 som finns, där 100 är så bra kvalité som går (fast ändå med distorsion) men med liten kompression, och 0 är så hårt komprimerat som går med mycket distorsion. Vi har använt sex olika nivåer på jpeg-komprimeringen: Kvalité 100, 90, 80, 50, 30

och 10. Dessutom har vi undersökt att inte använda någon komprimering alls. Totalt används alltså sju olika nivåer av bildkvalité på bilderna.

När vi skapat bildpatcharna har vi utgått från större bildpatchar på 100 x 100 pixlar, se figur 6 (a). För att inte blocken skall hamna på samma ställe i alla bilder har vi sedan pseudoslumpmässigt minskat patcharna med mellan 0 och 7 pixlar i höjd respektive bredd. Sedan har sex olika versioner av bildpatchen skapats med de olika komprimeringsgraderna. Slutgiltligen har de komprimerade bildpatcharna beskurits till 50 x 50 pixlar med fordonet i mitten. Dessa bildpatchar sparas därefter i det icke-förstörande formatet PNG för att inte införa mer komprimeringsdistorsion än den vi är intresserad av. Även de okomprimerande bildpatcharna, som figur 6 (b), har beskurits till 50 x 50 pixlar.



Figur 7: ROC-kurva för de sju tillfällena då de är tränade på en komprimeringsgrad och sedan testad på samma komprimeringsgrad.

I detektorns träningsfas och under evalueringen används negativa bildpatchar som inte innehåller något fordon. Vi har valt att även använda samma nivå av komprimering på dessa. Så om vi har tränat detektorn på bildpatchar med fordon inställd på jpeg-kvalité 50 har även bildpatcharna utan fordon haft jpeg-kvalité 50. Samma sak vid evaluering. Testar vi detektorn på bildpatchar med fordon inställd på jpeg-kvalité 10 har även negativa bildpatcharna utan fordon med jpeg-kvalité 10 använts.

3.3 Resultat

Nu har vi alltså skapat sju uppsättningar med samma bildpatchar fast med olika grad av komprimering och därmed komprimeringsdistorsion. För varje komprimeringsgrad finns det 1863 bildpatchar med fordon som kan användas vid träning. På samma sätt finns det för varje komprimeringsgrad 1242 bildpatchar med fordon som kan användas vid evaluering.

Det går alltså träna detektorn på någon av de sju uppsättningarna bildpatchar av olika bildkvalité. På liknande sätt kan detektorn evalueras av någon av de sju uppsättningarna med bildpatchar av olika bildkvalité. Allt som allt finns det då 49 olika kombinationer. Vi har testat dem alla.

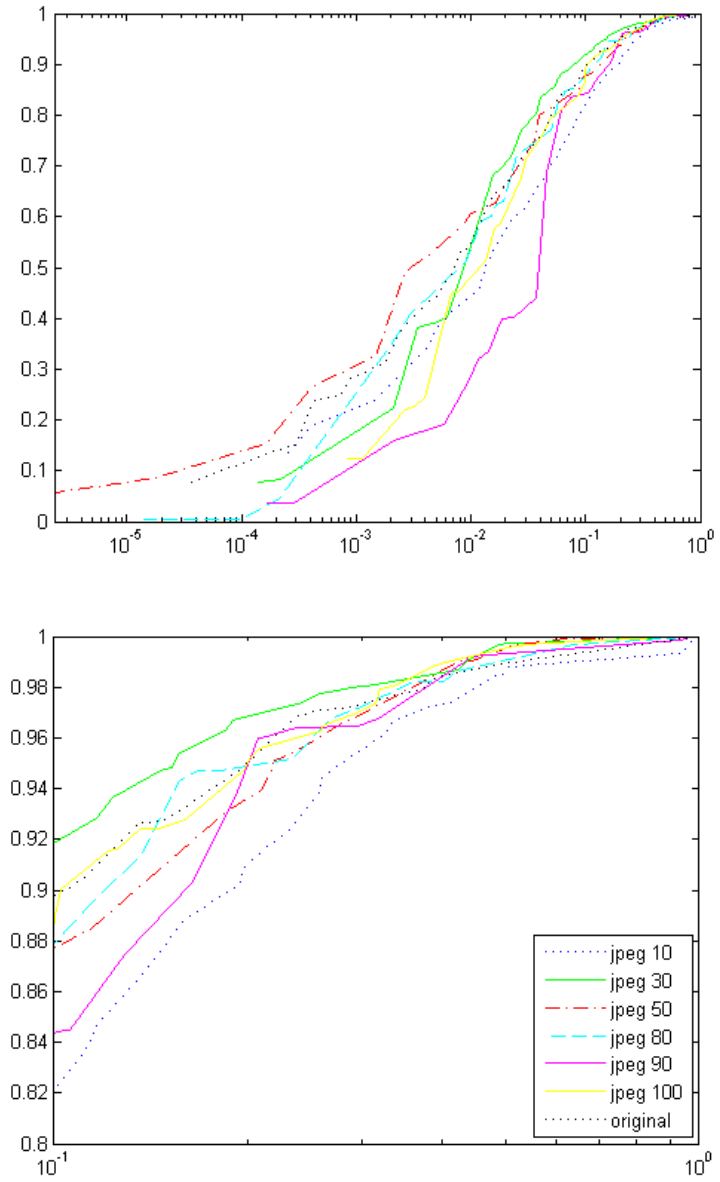
3.3.1 Samma bildkvalité på träning och test

Vid undersökning av hur detektorn påverkas av att arbeta med komprimerad sensordata har vi använt ROC-kurvor, figur 7. Dessa illustrerar relationen mellan antal fordon som har detekterats och antalet falskdetektioner. På den nedre axeln visas andelen falsklarm, alltså hur sannolikt det är att en bildpatch utan fordon ändå anses innehålla ett fordon enligt detekton. Vi vill hålla denna sannolikhet så låg som möjligt. Den övre axeln anger hur stor andel av fordonen som detekteras, alltså hur många av bildpatcharna med fordon som detektorn anser innehålla fordon. Denna sannolikhet vill vi hålla så hög som möjligt.

Att vi får en kurva beror på att ett tröskelvärde måste ställas mellan vad som anses vara fordon och icke-fordon. Genom att ändra detta tröskelvärde ändras förhållandet mellan antalet detekterade fordon och falsklarm. ROC-kurvan kan alltså användas för att bestämma var på kurvan vi vill lägga oss, dvs vilket tröskelvärde som skall användas. Genom att jämföra flera olika ROC-kurvor går det avgöra vilken detektor som är bäst.

Egentligen skulle även alternativ till ROC-kurvor användas för att se om de ger liknande resultat, eftersom det skulle göra resultaten mer tillförlitliga. Inom detta lilla projekt har det tyvärr inte funnits utrymme för det. Andra sätt att värdera resultatet från detektorn beskrivs i [7][13].

I figur 7 kan vi se resultatet för när detektorn har tränats på en bildkvaliténivå och samma kvalité sedan har använts vid evalueringen. Eftersom det finns sju olika grader av kvalité är det sju ROC-kurvor utritade i olika färg. Vi noterar att det är svårt att avgöra skillnaden mellan de sju linjerna. En helt ideal detektor ligger högst uppe i det vänstra hörnet när alla fordon detekteras och inga falska detektoner förekommer.



Figur 8: (a) ROC-kurva för de sju tillfällena då de är tränade på en komressionsgrad och sedan testad på samma kompressionsgrad. (b) För att tydligare se skillnaden i övre delen visas kurvan för området med andelen falsklarm mellan 0.1 och 1.

Svart linje är för originalbild, gul linje jpeg-kvalite 100, magentafärgad linje är jpeg-kvalité 90, cyan jpeg-kvalité 80, röd jpeg-kvalité 50, grön jpeg-kvalité 30 och slutligen blå linje jpeg-kvalité 10.

Genom att använda en logaritmisk skala på x-axeln blir det enklare att uttyda skillnaden mellan de olika linjerna, se figur 8 (a). Vi ser också att det inte är bildpatcharna med sämst bildkvalité (blå linje) som är sämst utan oväntat nog är det jpeg-kvalité 90 (magenta) som ligger längst ner för falsklarmssannolikhet < 0.05. I denna undersökning utgår vi från att vi är intresserade av att detektera många fordon. Därför är vi extra intresserad av den vänstra delen av kurvan. I figur 8 (b) tittar vi på området då falsklarmssannolikheten > 0.1. Inom detta område är det mycket riktigt jpeg-kvalité 10 (blå linje) som är ger sämst resultat. Mer oväntat är det inte icke-komprimerad data (svart linje) utan jpeg-kvalité 30 (grön linje) som ger bäst resultat!

3.3.2 Fast bildkvalité på träning och varierad bildkvalité på test

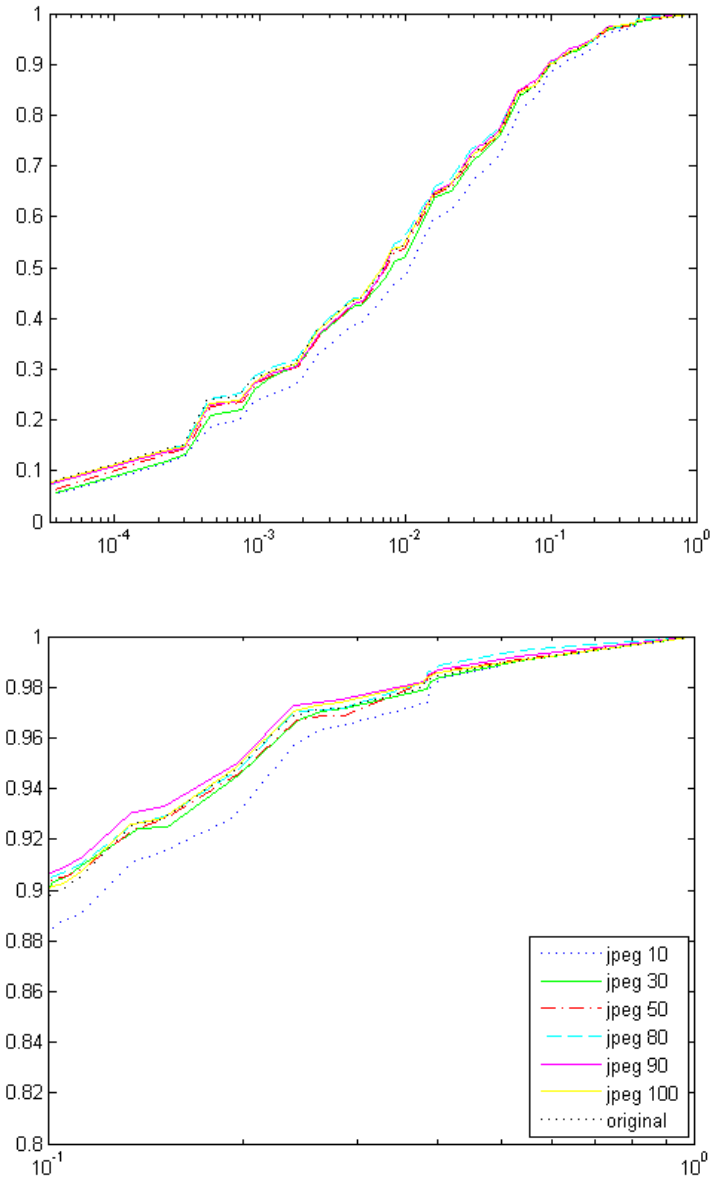
I detta stycke redovisar vi resultaten från när träningen av detektorn sker på en fast kvalité och evalueringen görs på vart och en av de sju kvalitétsnivåerna. Färgerna på linjerna motsvarar de samma som tidigare och anger den bildkvalité som använts vid testningen.

En inledande gissning från vår sida är att om detektorn tränas på bildpatchar från en kvalitétsnivå så kommer resultatet bli bäst vid evalueringen av bildpatchar med samma kvalitétsnivå. Det visar sig dock inte vara så enkelt.

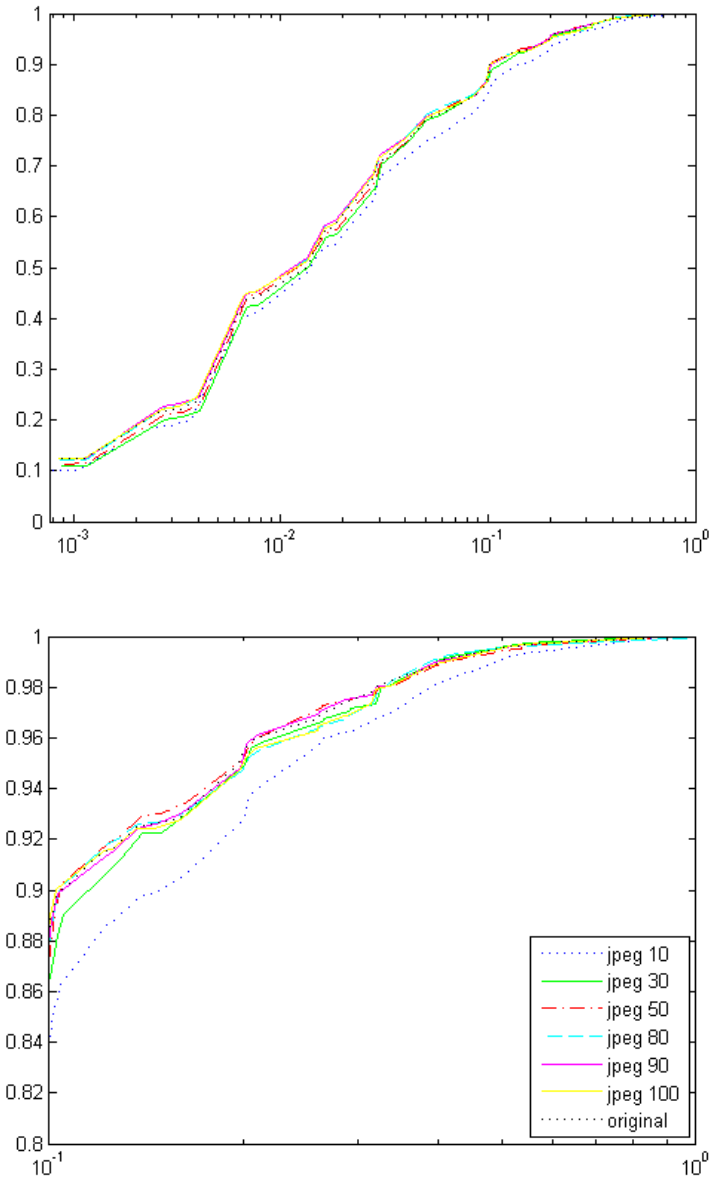
När träningen har skett på okomprimerade bildpatchar blir resultatet rätt väntat, figur 9. Evaluering av de mest komprimerade bilderna (blå) ger sämst resultat. Fast vi kan notera att jpeg-kvalité 90 (magenta) och 80 (cyan) i evalueringen ger bättre resultat än att använda icke-komprimerade bilder (svart).

För övriga figurer 10-15 får vi rätt liknande resultat. Intressant är att skillnaden vid olika bildkvalité ändå kan bli så stor. Det mest överraskande är att bäst resultat av evalueringarna fås efter träning på jpeg-kvalité 30 (figur 14) för en falsklarms-sannolikhet mellan 0.1 och 0.5. En gissning är att jpeg-komprimeringen för fram artefakter som detektorn har användning av. Lite som en förfiltrering. Fast det är ändå oväntat att resultatet blir så bra även när evalueringen sker på bilder som inte innehåller den formen av kodningsartefakter.

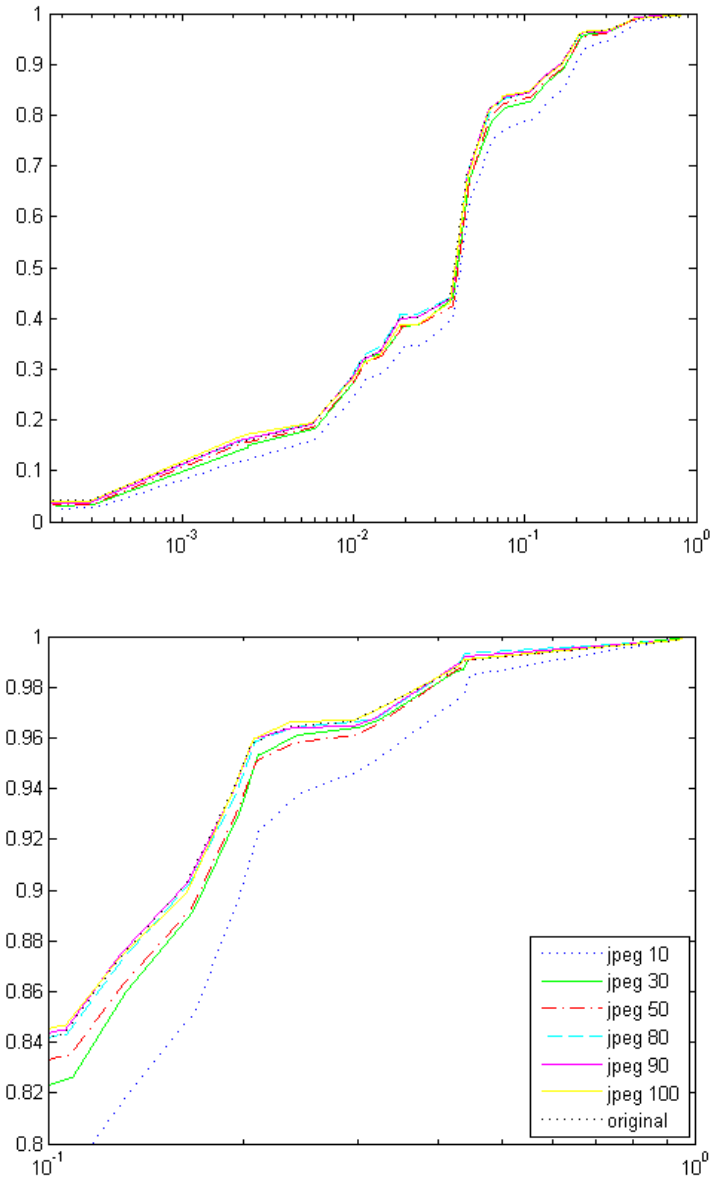
Notera att skalan på x-axeln skiljer sig mellan de olika övre bilderna (a) nedan. Det kan därför vara lite svårt att jämföra dem åt för låga falsklarmssannolikheter. I nästa sektion finns andra figurer där det är enklare att jämföra resultaten från olika kvalité på träningen.



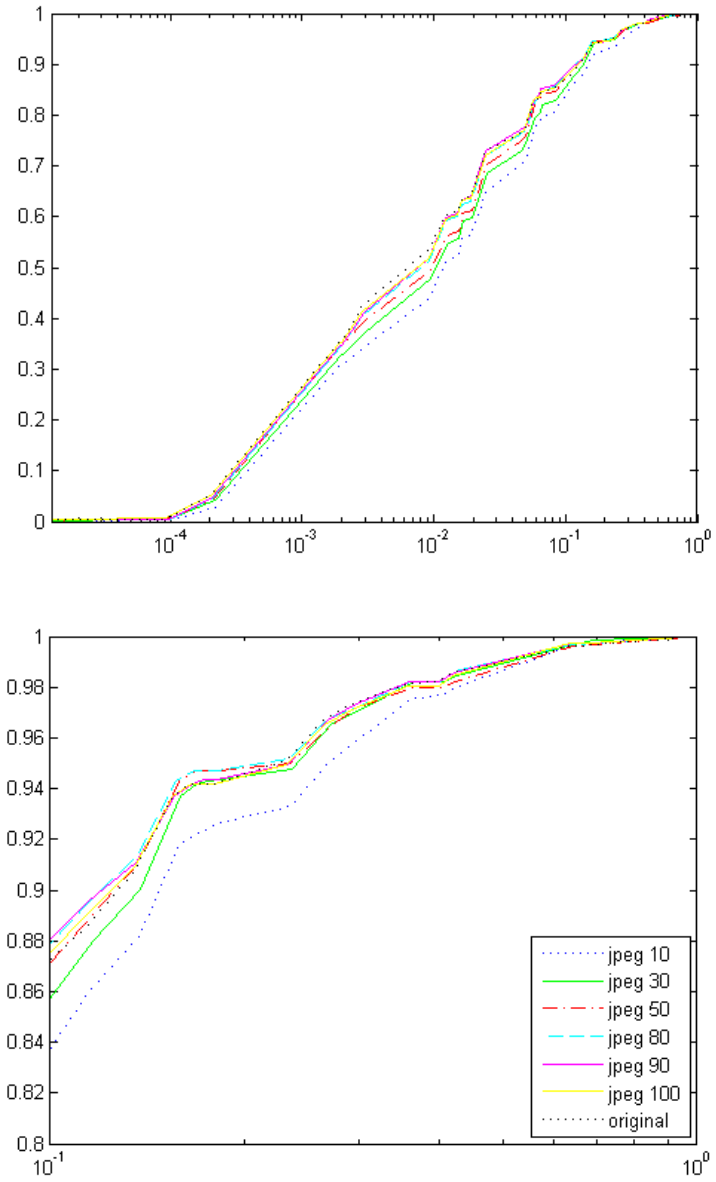
Figur 9: ROC-kurva för när träningen har skett på originalbilder och evaluering skett på bildpatchar för de sju olika kvalitetsnivåerna. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



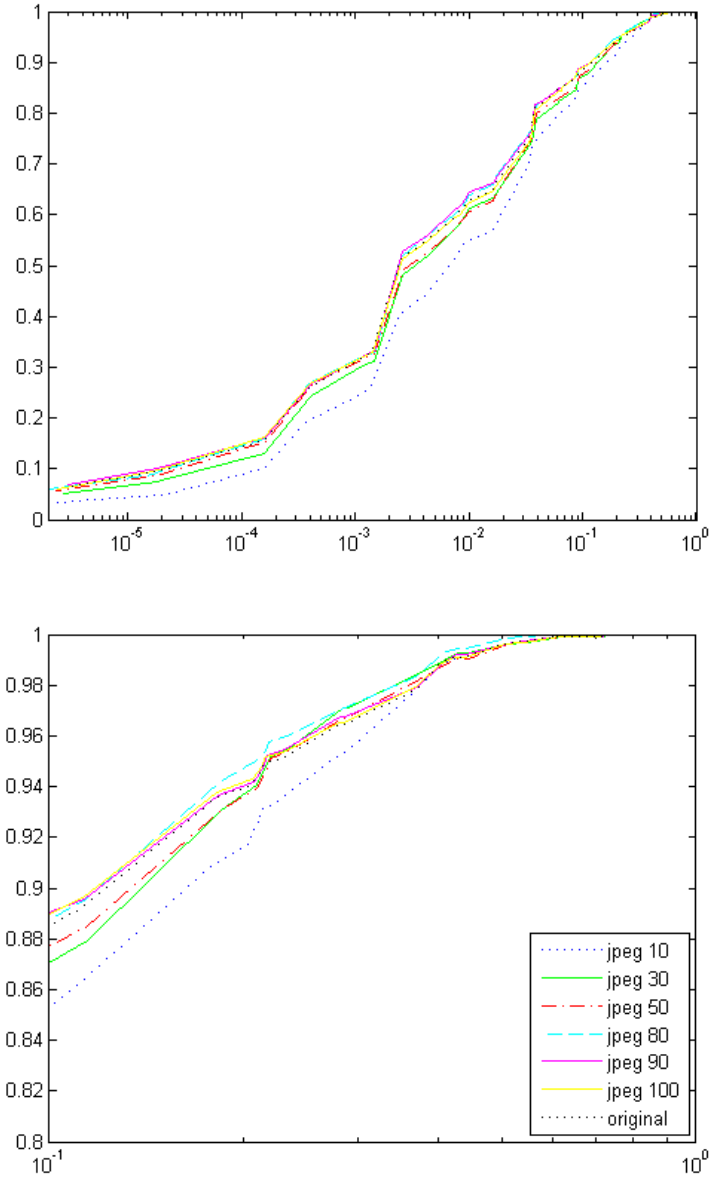
Figur 10: ROC-kurva för när träningen har skett på bilder med jpegkvalité 100 och evaluering skett på bildpatchar för de sju olika kvalitetsnivåerna. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



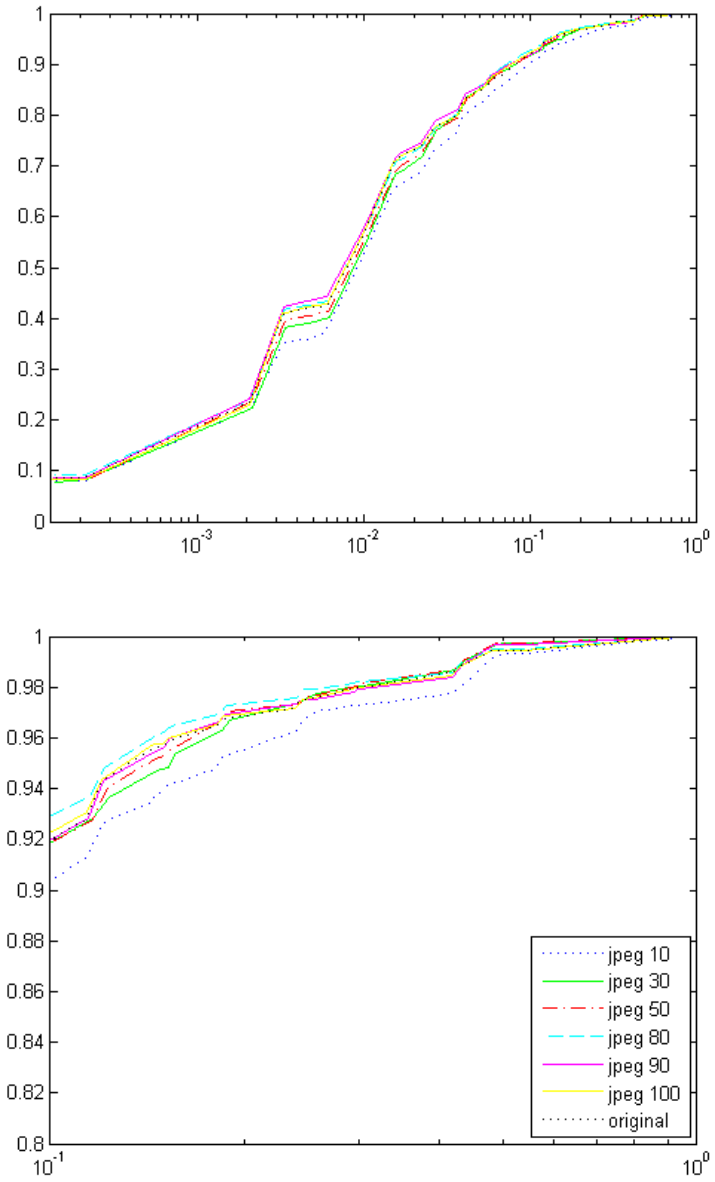
Figur 11: ROC-kurva för när träningen har skett på bilder med jpegkvalité 90 och evaluering skett på bildpatchar för de sju olika kvalitetsnivåerna. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



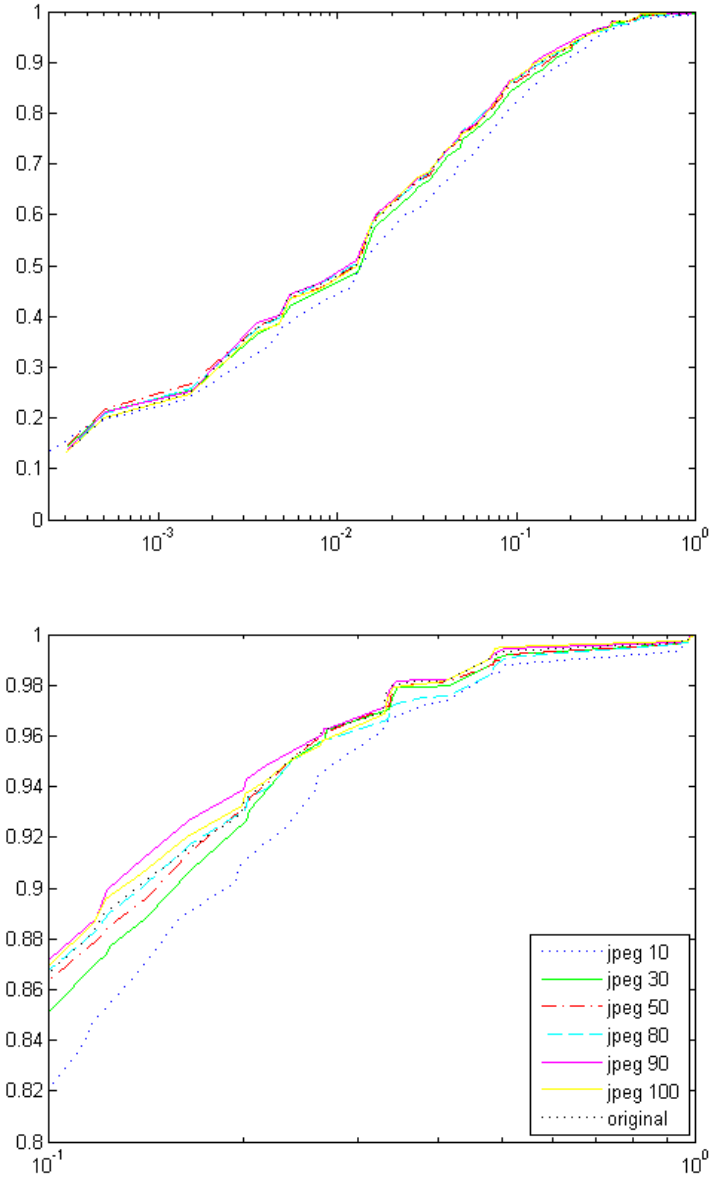
Figur 12: ROC-kurva för när träningen har skett på bilder med jpegkvalité 80 och evaluering skett på bildpatchar för de sju olika kvalitetsnivåerna. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



Figur 13: ROC-kurva för när träningen har skett på bilder med jpegkvalité 50 och evaluering skett på bildpatchar för de sju olika kvalitetsnivåerna. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



Figur 14: ROC-kurva för när träningen har skett på bilder med jpegkvalité 30 och evaluering skett på bildpatchar för de sju olika kvalitetsnivåerna. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



Figur 15: ROC-kurva för när träningen har skett på bilder med jpegkvalité 10 och evaluering skett på bildpatchar för de sju olika kvalitetsnivåerna. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.

3.3.3 Varierad bildkvalité på träning och fast bildkvalité på test

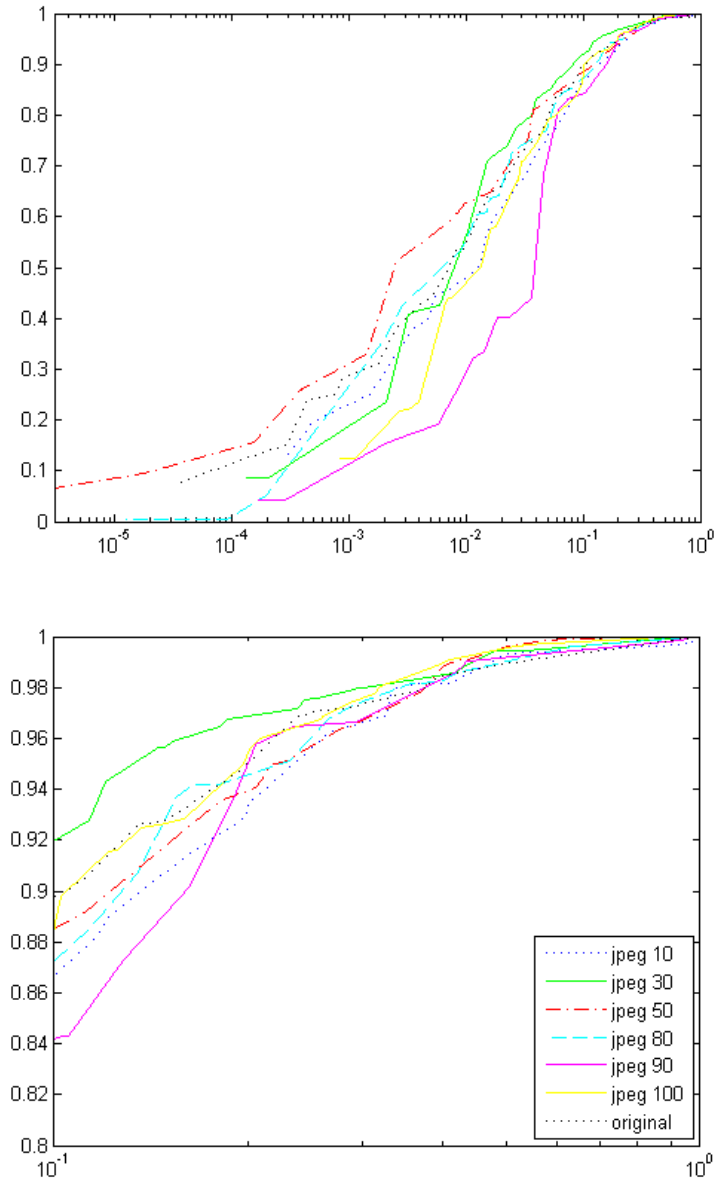
Vi har även skapat ROC-kurvor med omvända förutsättningar mot föregående sektion. Alltså en fast komprimeringssgrad på evalueringen och med varierad komprimeringsgrad vid träningen av detektorn. På liknande sätt som tidigare får vi då sju figurer med ROC-kurvor. Färgerna på linjerna motsvarar återigen samma saker som tidigare och anger nu den bildkvalité som använts vid träningen.

Om vi tittar på figur 16-22 så ser vi att resultatet är ganska liknande mellan dem. Träning på jpeg-kvalité 90 (magentafärgade linjer) ger dock alltid sämre resultat, eftersom den dyker snabbt vid falsklarmssannolikhet < 0.1 vilket vi även kunde se på figur 11. Även träning på jpeg-kvalité 100 (gula linjer) ger rätt dåligt resultat. Dessa resultat är förvånande. En liten kvalitétsförsämring som för det mänskliga synsinnet innehåller icke noterbara förändringar är alltså sämre vid träning av detektorn än stor distorsion med tydliga kodningsartefakter! Detta gäller dock bara för falsklarmssannolikhet < 0.2. Det skulle kunna vara så att svag komprimering är en form av icke-gynsam förfiltrering som förstör vid träning av detektorn.

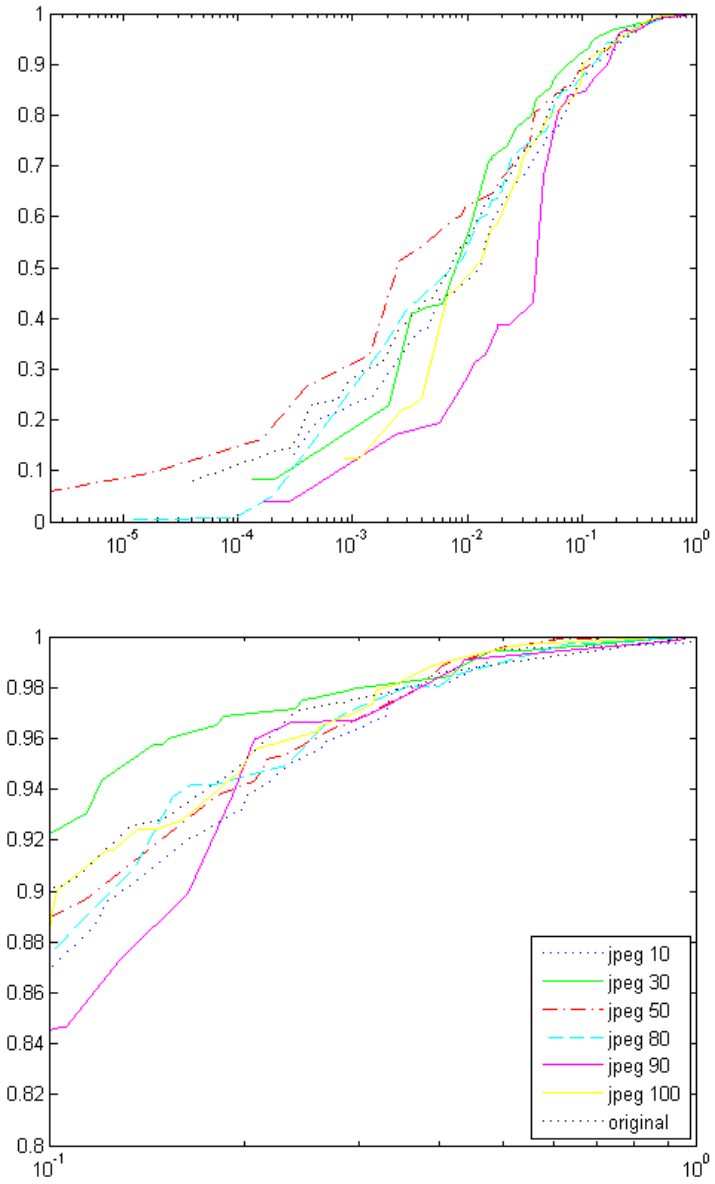
Återigen ser vi att jpeg-kvalité 30 (gröna linjer) vid detektionsträningen ger bäst resultat vid falsklarmssannolikhet melllan 0.1 och 0.5. För lägre falsklarmssannolikheter är det istället jpeg-kvalité 50 (röda linjer) som ger bäst resultat.

Relationen mellan kvalité på träningen och testet av detektorn är altså inte så enkel som vi förutspådde. För figur 18 är evalueringen med bildpatchar av jpeg-kvalité 90 sämst när detekton har tränats just med jpeg-kvalité 90. Det är svårt att hitta ett entydigt mönster för de olika resultaten. En sak som står klar är i alla fall att komprimeringsartefakter påverkar både träningen av detektorn och evalueringesresultatet. Fast ibland påverkar artefakterna på positivt sätt och ibland på negativt sätt.

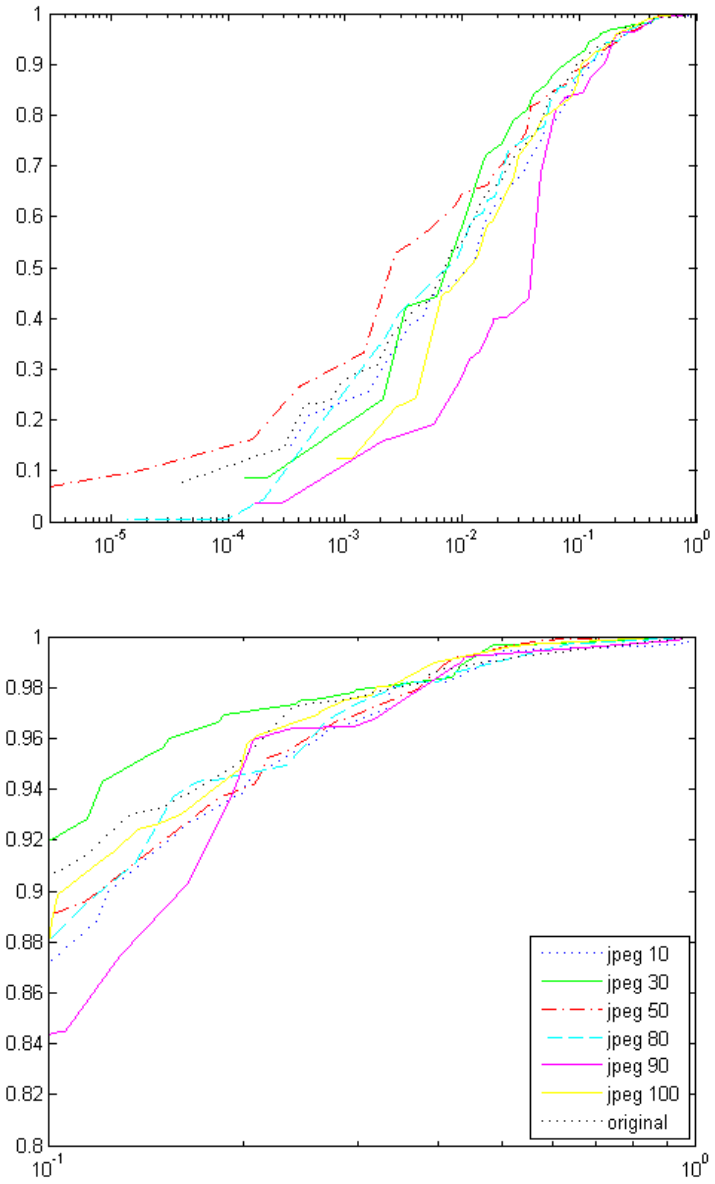
Eftersom det förekommer en form av slump under träningen så kan vissa av skillnaderna mellan de olika träningarna härstamma från det. Tyvärr finns det inte möjlighet inom denna studie att genomföra flera träningar för att undersöka detta närmare. Viola & Jones-särdragen är till skillnad från vissa andra särdrag inte lika beroende av kanter, vilket gör att de genom sin medelvärdesbildande förmåga troligtvis är mer robust mot de kantbildande kodningsartefakterna. Dessutom påverkar valet av antal särdrag, där större antal särdrag gör detektorn känsligare mot förändringar mellan tränings- och evalueringsdata. Även andra särdrag, än de från Viola & Jones, och antalet använda särdrag, bör därför undersökas närmare om en större studie skulle genomföras.



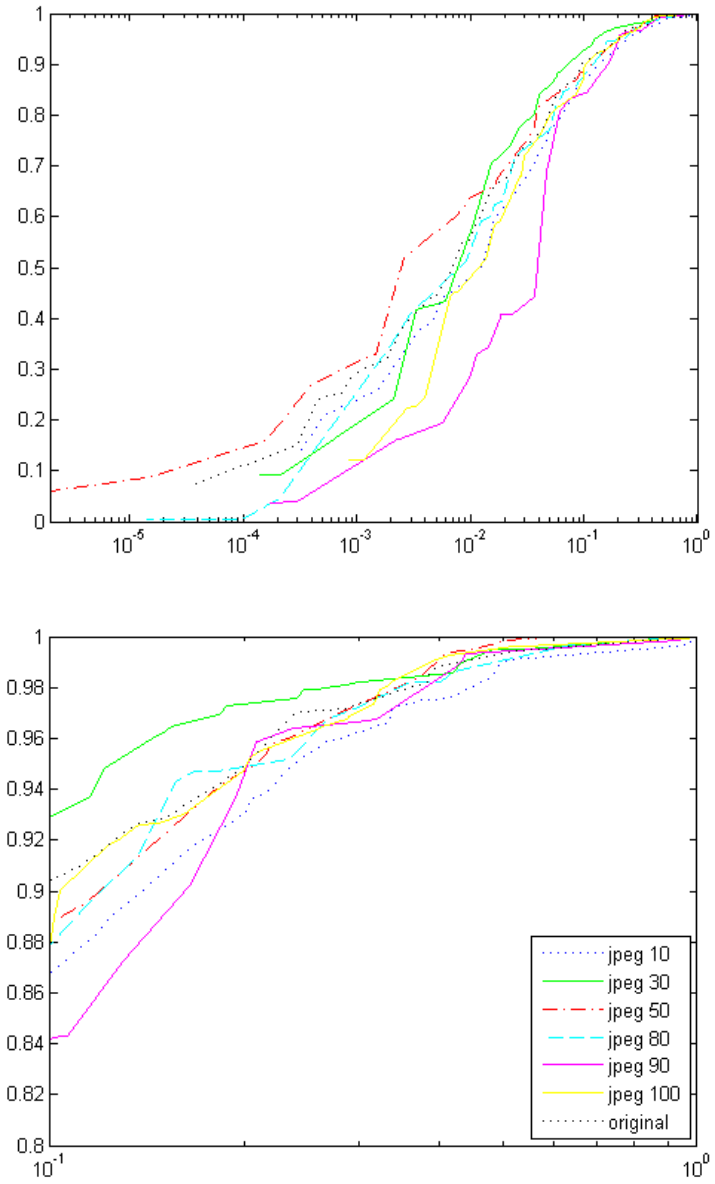
Figur 16: ROC-kurva för när träningen har skett på bildpatchar med sju olika kvalitétsnivåer och evaluering skett på okomprimerade bildpatchar. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



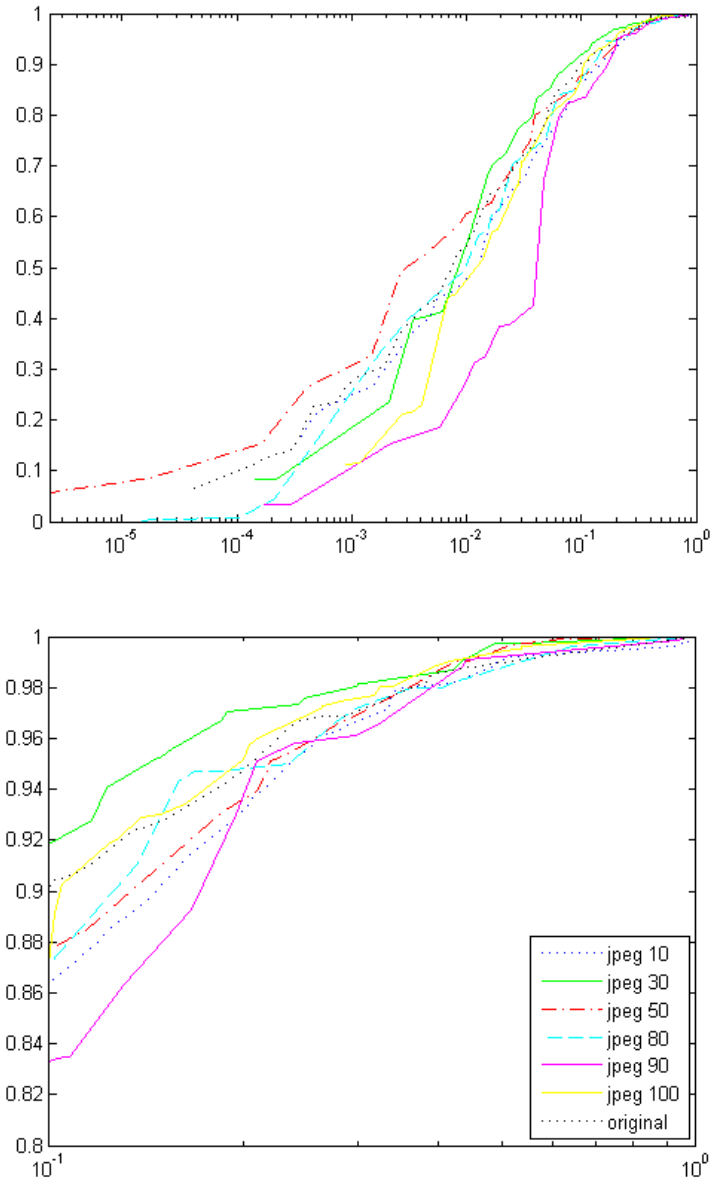
Figur 17: ROC-kurva för när träningen har skett på bildpatchar med sju olika kvalitétsnivåer och evaluering skett på bildpatchar med jpegkvalité 100. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



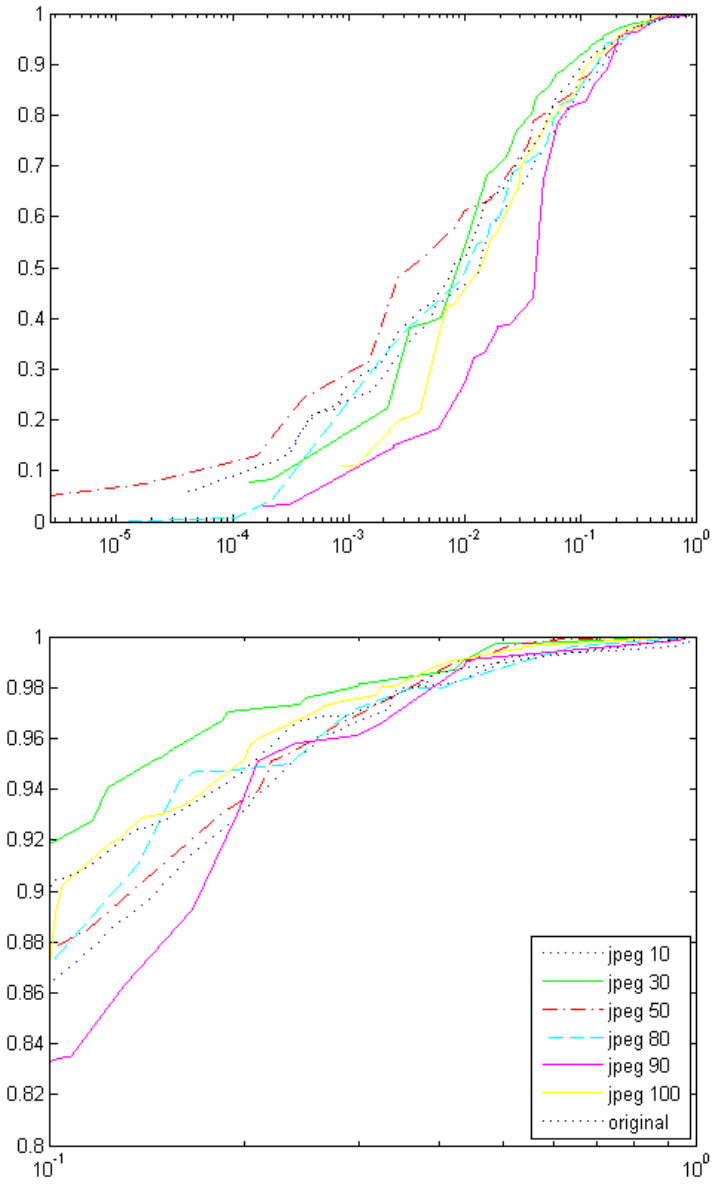
Figur 18: ROC-kurva för när träningen har skett på bildpatchar med sju olika kvalitétsnivåer och evaluering skett på bildpatchar med jpegkvalité 90. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



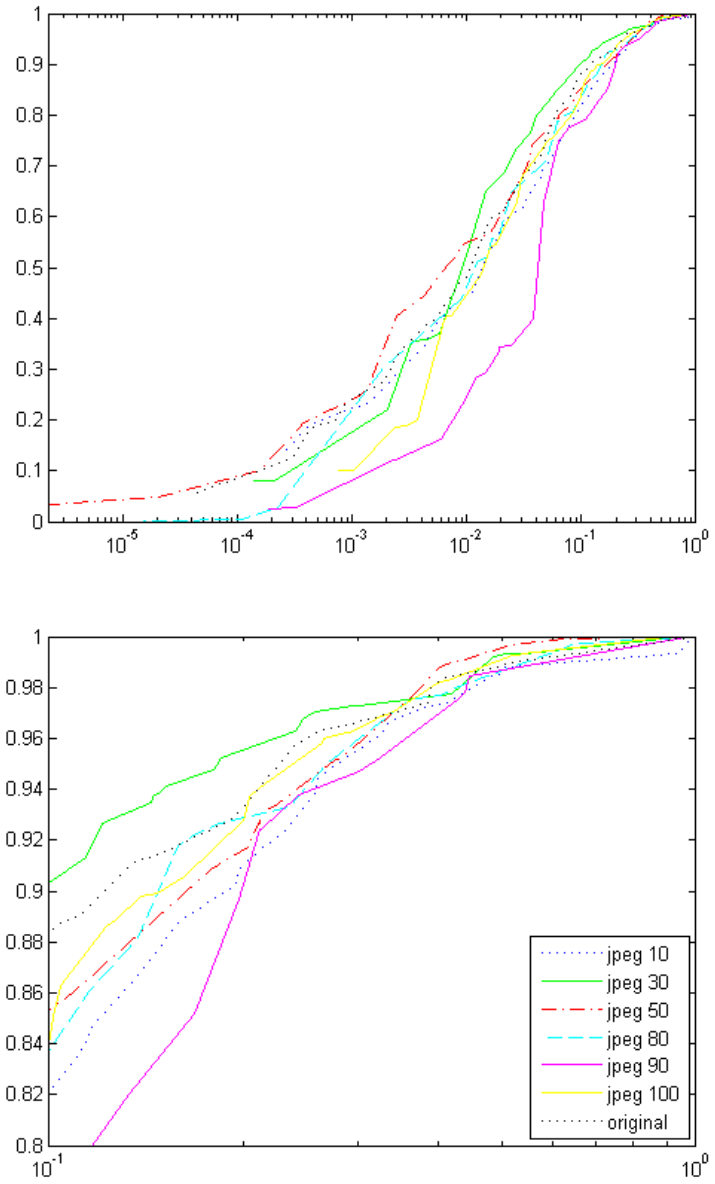
Figur 19: ROC-kurva för när träningen har skett på bildpatchar med sju olika kvalitétsnivåer och evaluering skett på bildpatchar med jpegkvalité 80. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



Figur 20: ROC-kurva för när träningen har skett på bildpatchar med sju olika kvalitétsnivåer och evaluering skett på bildpatchar med jpegkvalité 50. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



Figur 21: ROC-kurva för när träningen har skett på bildpatchar med sju olika kvalitétsnivåer och evaluering skett på bildpatchar med jpegkvalité 30. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.



Figur 22: ROC-kurva för när träningen har skett på bildpatchar med sju olika kvalitétsnivåer och evaluering skett på bildpatchar med jpegkvalité 10. (a) är hela ROC-kurvan och (b) visar kurvan för området med andelen falsklarm mellan 0.1 och 1.

4 Slutsatser

Vi har utökat simuleringslaboratoriumet MSSLab med möjlighet att köra komprimerade bilder i HLA genom utbyggnad av FOM:en. Nu går det skapa och skicka jpeg-komprimerad sensordata mellan olika federater, som exempelvis de som används för detektion och återigenkänning.

Tester har utförts för att utröna hur komprimering, och den distorsion som den orsakar, kan påverka en detektionsalgoritm. Den valda detektorn lär sig vad som skall detekteras genom en träningsfas, så kallad boosting. Sedan evalueras detektorn för att se hur bra resultat den får från den genomförda träningen. Vi använde bilder som inte var komprimerade, samt komprimerade bilder med jpeg-kvalité 100, 90, 80, 50, 30 och 10 i vår studie. Både träningen och testen utfördes på de sju olika sorternas bilder vilket resulterade i 49 olika fall.

Komprimeringen påverkar resultaten, både i tränings- och evalueringsfasen. Även när det mänskliga synsinnet har svårt att se några artefakter orsakad av komprimeringen, så blir det skillnader i resultatet från den automatiska detektorn. Däremot går det inte att säga att komprimeringsartefakterna enbart är av ondo. I vissa fall visar det sig istället att komprimerade bilder ger bättre resultat när de används vid träningen och lika så vid evalueringen. I andra fall påverkar istället komprimeringen resultatet från träning eller test negativt. Det visar sig dessutom att det inte alls är säkert att det är bäst att både träna och evaluera med samma bildkvalité.

Vi har använt ROC-kurvor när vi jämfört resultaten, eftersom det är det är det mest använda redskapet. Även andra sätt att värdera resultatet från detektorn bör egentligen användas för att se om de ger liknande resultat.

Bara en specifik algoritm har undersökts med vissa inställda fasta parametrar. Därför går det inte enkelt avgöra om dessa resultat enkelt kan appliceras på andra detektionsmetoder och andra automatiska signalbehandlingsalgoritmer.

Vi har haft en begränsad mängd bildpatchar som använts vid träning av detektorn och vid evalueringen. Det är möjligt att mer tydliga resultat skulle uppnås om en större mängd bilder hade använts. Fast de hade mycket mer tid krävts för att genomföra dessa operationer för de 49 olika fallen.

Ett sätt att undvika oväntade egenskaper från komprimerade bilder bör vara att se till att träna detektorn på bilder med olika komprimering, på liknande sätt som den tränar på bilder med olika ljussättning. Då bör detektorn bli mer robust mot olika grader av komprimering, som den kan komma i kontakt med. Inga sådana träningar har genomförts inom denna studie, varför vi inte heller har några resultat över huruvida det verkligen ger önskat resultat.

5 Referenser

[1] K-G Stenborg, Anna Linderhed, Niclas Wadstömer, Peter Follo, Mikael Lundberg, ”Sensornära datakomprimering – Slutrapport”, FOI-R--2942--SE, December 2009.

[2] Fredrik Näsström et al., ”Simuleringsbaserade metoder för sensorvärdering – Årsrapport 2008”, FOI-R--2629--SE, December 2008.

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

[4] Fredrik Bennet, Stafan Fenelius, "SceneServer - a 3D software assisting developers of computer vision algorithms", FOI-R--0831--SE, Mars 2003.

[5] Fredrik Näsström, Mirsad Cirkic, Joakim Rydell, Per Skoglar, K-G Stenborg, Morgan Ulvklo, Autonoma funktioner för intelligenta operatörsstöd, FOI-R--2752--SE, Mars 2009.

[6] Fredrik Näsström et al, ”Simuleringsbaserade metoder för sensorvärdering – Årsrapport 2009”, FOI-R--2890--SE, December 2009.

[7] Jörgen Ahlberg, Mikael Lundberg, Joakim Rydell, ”Analys av multi/hyperspektrala bilder – framsteg och resultat”, FOI-R--2895--SE, December 2009.

[8] Paul Viola, Michael Jones, “Robust real-time object detection”, Technical Report CRL 2001/01, Cambridge Research Laboratory, Compaq Computer Corporation, 2001.

[9] Anna Linderhed, Jörgen Ahlberg, Martin Blom, Björn Flood, Peter Follo, Johan Rasmusson, ”Inledande studie Sensornära datakomprimering”, FOI Memo 1818, November 2006.

[10] Anna Linderhed, K-G Stenborg, Peter Follo, Mikael Lundberg, ”Sensornära Datakomprimering - Årsrapport 2007”, FOI-R--2431--SE, December 2007.

[11] JPEG Standard, “ISO/IEC 10918-1 ITU-T Recommendation T.81”, 1992-1998.

[12] Khalid Sayoood, “Introduction to data compression - Third edition”, San Francisco, CA: Elsevier/Morgan Kaufmann, 2006.

[13] Anna Linderhed, Niclas Wadströmer, K-G Stenborg, Harald Nautsch, “Compression of hyperspectral data for automatic analysis”, SPIE Conference on Image and Signal Processing for Remote Sensing, September 2009.