Tactical Overlay System, part 2

Oskar Norberg

Robert Johansson

2008-03-09

Master's Thesis in Computing Science, 2*30 ECTS credits

Supervisor at CS-UmU: Niclas Börlin

Supervisor at BAE Systems: Johan Ohlsson och Mikael Knutsgård

Examiner: Per Lindström

Umeå University

Department of Computing Science

SE-901 87 UMEÅ

SWEDEN

Abstract

The interest for integrating visual aids in combat vehicles has been growing in recent years within BAE Systems, Hägglunds. In a previous Master's thesis "Integration av data från yttre källor i en videobild" [1] (Integration of data from external sources into a videoframe) by Markus Isaksson and Jesper Wide, a demonstrator was designed that could fuse simple computer graphics with a video stream.

The aim of this report is to develop the concept of video-overlaid computer graphics further. Important questions to address is what information is helpful and interesting for the user and how should the overlayed video stream be presented to the user. A demonstrator has been implemented. The demonstrator provides BAE Systems with better knowledge of potential obstacles in developing a commercial system. Furthermore, the demonstrator suggests what functionality would be useful in a real deployment.

Keywords: Information presentation, video overlay, 3D-graphics, data fusion, interface

Sammanfattning

De senaste åren har intresset att integrera visuellt stöd i form av datorgrafik i stridsfordon ökat inom BAE Systems, Hägglunds. I ett tidigare examensarbete ”*Integration av data från yttre källor i en videobild”* [1], av Markus Isaksson och Jesper Wide, har en enklare demonstrator utvecklats. Demonstratorn kunde huvudsakligen sammanfoga enkel datorgrafik med en videoström.

Denna rapport utgör resultatet av ett examensarbete, inom civilingenjörsprogrammet i datavetenskap, och har som syfte att vidareutveckla detta koncept med videoöverlagrad datorgrafik. Likaså att finna svar på vilken information som skulle vara intressant att grafiskt kunna åskådliggöra och vilken nytta detta skulle innebära för användaren. Vidare utreds frågan om hur information ska kunna integreras och presenteras. En ny demonstrator har implementerats, som är mer dynamisk och som totalt sett har en större praktisk tillämpbarhet. Genom beskrivet förfarande ges en bättre bild av hur ett system som detta skulle kunna se ut, bättre kunskap om vilka hinder och problem som finns vid systemutvecklingen, men även vilka funktioner som skulle kunna vara användbara och tillämpningsbara i en reell version av ett system som detta.

Nyckelord: Informationspresentation, video, 3D-grafik, datafusion, interface

Innehållsförteckning

Figurförteckning

1 Inledning

Detta kapitel beskriver bakgrunden till detta projekt och lite om anledningen till varför detta arbete ska genomföras.

Integration av data från multipla yttre informationskällor är idag en aktuell fråga, inte minst vid utvecklingen av ledningssystem för olika stridsfordon. Anledningen till detta är att personalen i allt större utsträckning tvingas hantera en växande mängd information som samtidigt blir allt mer komplex. I dagsläget presenteras denna information ofta via ett flertal olika medier, på ett flertal skärmar eller vyer. Detta kan göra informationen svårtolkad samt försvåra för användaren att ta ställning till vad som måste göras i en kritisk situation. Ifall information från dessa multipla källor, som kan vara exempelvis kameror, sensorer, kartor och information från stridsledningssystem, kunde sammanställas samt presenteras på ett överskådligt sätt i en och samma bild, vore det att föredra. Förhoppningen är således att förenkla arbetet för personalen, göra den mer handlingskraftig i kritiska situationer och även att i större utsträckning, då det gäller stridsfordon, undanröja nödvändigheten för optiska utblickar som kan vara mål för fientlig laserstrålning eller dylikt.

2 Problembeskrivning

I detta kapitel formuleras problemet som ska lösas samt relaterat arbete där det ges en liten bakgrund om tidigare arbeten som detta arbete bygger på.

2.1 Problemformulering

Idag finns det visioner hos företaget BAE Systems att integration och fusion av data från multipla källor, i deras användargränssnitt, till stor del skulle kunna öka en användares förståelse samt handlingskraft vid olika situationer. Detta har skapat intresse från företagets sida att utreda ifall det är möjligt att införa liknande funktionalitet i något av deras fordon. Därför vill de att en undersökning ska göras för att ta reda på vilken information ett sådant system ska kunna hantera och sedan realisera detta i form av en demonstrator. Huvudsakligen för att ge en bild av hur ett sådant system skulle kunna se ut och hur väl det skulle fungera. Den största delen av informationen och sannolikt den viktigaste, i realiteten, kommer från någon form av stridsledning, via ett så kallat *Battlefield-Management-System* (BMS). Idag existerar det ett flertal olika sådana ledningssystem. Dessa kan skilja, inte minst, från nation till nation, men även när det gäller vilka meddelandetyper som skickas, formatet på dessa samt underliggande kommunikationslänk. Därför måste det även undersökas hur ett generellt gränssnitt kan implementeras som gör det lätt att konfigurera och distribuera systemet oavsett till vilket land eller oavsett vilket BMS som används.

Följande frågor måste alltså besvaras:

1. Hur ska ett generellt gränssnitt se ut?
2. Hur ska det utformas?
3. Hur ska sensorinformation på bästa sätt kunna integreras och representeras i systemet?
4. Vilken information är viktig att kunna hantera, visa eller distribuera?
5. Är det möjligt att implementera ett system som detta och med vilka resultat?

2.2 Begränsningar

Det praktiska arbetet begränsas till att använda den redan existerande demonstrationsvagnen och kommer därför att bestå av att utveckla ny mjukvara. Den nya mjukvaran i kombination med demonstrationsvagnen kommer endast att användas i demonstrativt syfte.

2.3 Relaterat arbete

I ett tidigare examensarbete på BAE Systems, som utfördes av Markus Isaksson och Jesper Wide i början av 2007, har en portabel demonstrationsvagn utvecklats med syftet att sammanföra information från flera olika källor till en enda vy i ett PC-program. Demonstrationsvagnen bestod huvudsakligen av en tröghetsnavigator, en dagsljuskamera samt en IR-kamera, en tablet-PC, en Axis 250S videoserver och till sist ett utvecklingskit från Analog Devices med en Blackfin DSP processor. Med hjälp av denna utrustning var det möjligt att med deras program läsa in information om områden i terrängen från fil och sedan projicera in området i bilden från dagsljuskameran. Det var även möjligt att mäta upp avstånd till olika punkter i videobilden genom matematiska uträkningar, givet kamerans position och riktning, tillsammans med information om omgivande terräng i form av en höjddatabas.

Utritningen av områdena i denna demonstrator gällde bara ett områdes kontur och utfördes med hjälp av *Microsoft-Foundation-Classes* (MFC) Pen-objekt där olika visuella effekter såsom perspektiv och skuggning implementerades på ett minst sagt omständigt vis. Exempelvis genererades det vid initiering hundra MFC Pen-objekt, tio olika tjocklekar med tio olika färger vardera, där sedan något av dessa objekt kunde användas för utritning. Valet av dessa objekt gjordes beroende på hur långt bort linjen befann sig eller om den skulle vara skuggad. En annan begränsning med denna lösning var att det inte var kopplat till något riktigt stridsledningssystem och inläsningen av nya områden var statisk, det gick alltså endast att läsa in ett område vid programstart från fil vilket inte var särskilt praktiskt. Stöd för användning av IR-kamera implementerades till den grad att information om ett objekts position och storlek kunde skickas via ett särskilt protokoll från Analog Devices-modulen till PC. Dock så integrerades aldrig denna information på något sätt i huvudprogrammet.

3 Metod

I detta kapitel beskrivs den utrustning som användes vid implementering av systemet samt några av metoderna för de viktigaste funktionerna.

3.1 Material

3.1.1 Tröghetsnavigator

En fristående *Dynamic Reference Unit* (DRU) används för navigeringen av systemet och bestämmer demonstratorvagnens position. Tröghetsnavigatorn tilldelas en referenskoordinat vid uppstart, sedan använder den sig av accelerometrar och gyron för att hålla reda på vilken position den har i världen och även hur den är roterad. Detta är ett norrsökande system som därför försöker att räkna ut i vilken riktning den geografiska nordpolen befinner sig. Detta gör den alltså genom att känna av och mäta jordens rotation runt sin egen axel och det är alltså dessa mätningar som utgör grunden för dess beräkningar. Enheten kan ge positionsangivelser i *Rikets koordinatsystem 1990* (RT90). [8]

3.1.2 Kamera

I systemet ingår en dagsljuskamera. Eftersom de optiska egenskaperna för denna kamera var okända har de uppskattats genom att göra avståndsmätningar på ett givet avstånd och ut till videobildens periferi och sedan bestämma de viktigaste egenskaperna med hjälp av enkel trigonometri.

3.1.3 Bärbar dator

En bärbar dator finns tillgänglig för att köra demonstratorn. Datorn är utrustad med en *Intel Core* processor på två gigahertz, har två gigabyte internminne och ett *ATI Mobility Radeon X300* grafikkort. Ett någorlunda bra och modernt grafikkort krävs för att hårdvaruaccelerationen som finns implementerad i grafikbiblioteket ska fungera och en bra processor är nödvändig för att kunna hantera den bild som videoservern producerade.

3.1.4 Höjddatabas

Höjddatabasen är inköpt från Lantmäteriet och är angivet i RT90 format. Höjddatabasen består av 40401 punkter ordnade i en tvådimensionell matris med 201 värden på längden och höjden. Värdena har ett mellanrum på 50 meter mellan varje värde och hela databasen täcker därmed ett område på en kvadratmil. Höjddatabasen har dock en felmarginal på cirka 2,5 m på värdena vilket kan komma att påverka felmarginalen i avståndet. [4]

3.1.5 Videoserver

För att kunna ta in videoströmmen till datorn, som saknar videoingång, används en videoserver som gör om den analoga videoströmmen till *Moving Pictures Expert Group 2* (MPEG2) som är en digital videokomprimeringsstandard. Videoservern skickar den digitala videoströmmen via Ethernet till datorn, varpå videoströmmen avkodas med en ActiveX-kontroller. Den senaste bilden som har avkodats och mottagits kan sparas undan i primärminnet eller till disk. [5]

3.1.6 Radio 180 och PC-DART

Två Radio 180-system har använts för att simulera hur systemet skulle kunna fungera i verkligheten genom att skicka och ta emot information. Radio 180 är ett radio-system med stöd för frekvenshoppning samt krypto och där sändningen kan ske både analogt och digitalt. PC-Datarapporteringsterminal (PC-DART) kopplas via Radio 180 för att kunna skicka DART-meddelanden. DART-meddelanden kan liknas med *Short Message Service* (SMS) och kan max innehålla 200 tecken. DART använder sig oftast av färdiga formatmallar och protokoll för de meddelanden som ska skickas. Kommunikationen mellan systemet och PC-DART sker via Ethernet. [7]

3.2 Grafikutritning

3.2.1 MFC - Nuvarande metod

Den existerande metoden för grafikutritning har brister på grund av tre skäl. För det första introducerar den videoserver man använde en fördröjning av videoströmmen. För det andra ritades grafiken ut i ett transparent fönster ovanför videoströmmen, vilket introducerade ett flimmer. Flimret kunde reduceras genom att endast rita ut grafiken ungefär två gånger i sekunden, men var fortfarande synligt och medförde att grafiken fick en låg uppdateringsfrekvens. Den tredje och kanske den värsta bristen var, som en effekt av att använda metoder i *Microsoft-Foundation-Classes* (MFC) för att rita ut all grafik, att man hade begränsade möjligheter att få till en trovärdig grafik. Exempelvis fanns det bara tillgång till tio olika nyanser av röd och tio olika tjocklekar på de linjer som man ritade ut, vilket medförde att speciella metoder för att skugga och bestämma tjockleken på linjen behövde användas. Dessutom saknades möjlighet att göra transparenta ytor.

3.2.2 OpenGL – Ny metod

Open Graphics Library (OpenGL) är ett öppet programmeringsbibliotek för grafikutritning. Fördelen med OpenGL jämfört med konkurrerade bibliotek är att OpenGL är ett öppet bibliotek som finns implementerat för de flesta plattformar och därmed är det inte bundet till något speciellt operativsystem.

Med OpenGL får man lösningar på de problem som det förra examensarbetet brottades med i princip gratis. Förutom att det är mycket

enkelt att implementera perspektiv och färger kan man dessutom använda till exempel djupbuffer, stencilbuffer och transparenta ytor. [2]

3.2.3 Rendering av video

ActiveX-kontrollern som följde med videoservern har en metod för att spara undan den aktuella bilden. Dock kunde inte ActiveX-kontrollern skapas som ett fristående objekt, utan den var i stället tvungen att skapas i en MFC-dialogruta i vilken den avkodade videoströmmen sedan kunde visas. Istället för att visa dialogrutan gömdes denna så den inte skulle synas, vilket inte är en elegant lösning eftersom videoströmmen spelas upp och upptar processortid. Videoservern introducerar också ett par sekunders fördröjning men eftersom detta bara är tänkt att användas som en demonstrator ansågs fördröjningen vara acceptabel.

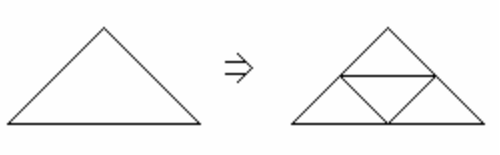
Av videobilden som ActiveX-kontrollern sparar undan från videoservern skapas sedan en textur med hjälp av färdiga metoder i OpenGL. Den texturen appliceras sedan på en fyrkant som ritas ut ortogonalt på skärmen. Efter fyrkanten har ritats ut kan i princip vad som helst ritas på den och på så sätt undviks flimmer, vilket som tidigare nämnts var ett problem i tidigare version. [5]

3.2.4 Triangulering

När ett område markeras på kartan bildas en yta i form av en polygon. Om endast dessa punkter skulle användas för att rita ut detta område skulle den verka helt platt. Därför måste polygonen delas upp i trianglar för att få den att följa terrängen. Den metod som används letar efter polygonens öron och skapar en triangel om den hittar ett öra. Ett öra utgörs av tre stycken konsekutiva punkter v_1, v_2, och v_3, tillhörande polygonen, och där ingen av polygonens andra punkter befinner sig inom den triangel som dessa tre punkter bildar. Öronpunkten får heller inte ha en vinkel in mot polygonen som är mer än 180 grader, - har den det innebär det att den triangel som den då skulle skapa ligger på utsidan av polygonen. Sammantaget fungerar metoden ganska bra, men det finns några begränsningar. För det första fungerar den bara med enkla polygoner, det vill säga polygoner som inte har någon sida som överlappar en annan sida. Polygonerna får heller inte innehålla några hål. För det andra är den relativt långsam, $O(n^3)$, det finns snabbare algoritmer men de är krångligare att implementera och eftersom man sällan använder polygoner som är över tre eller fyra punkter sparar man inte vidare mycket på att byta ut den mot en snabbare algoritm. Dock skulle den kunna bytas ut mot en mer robust algoritm som klarar av mer avancerade polygoner. [3, 14]

3.2.5 Subdivision

För att få en yta som följer terrängen måste man dela upp de trianglar man får av trianguleringen till mindre trianglar. Den algoritm som används här är en mycket enkel algoritm som kräver en stor mängd minne. Algoritmen skapar först tre nya punkter på varje sida av triangeln och av dessa punkter skapar den sedan fyra nya trianglar av de sex punkterna som redan finns, se



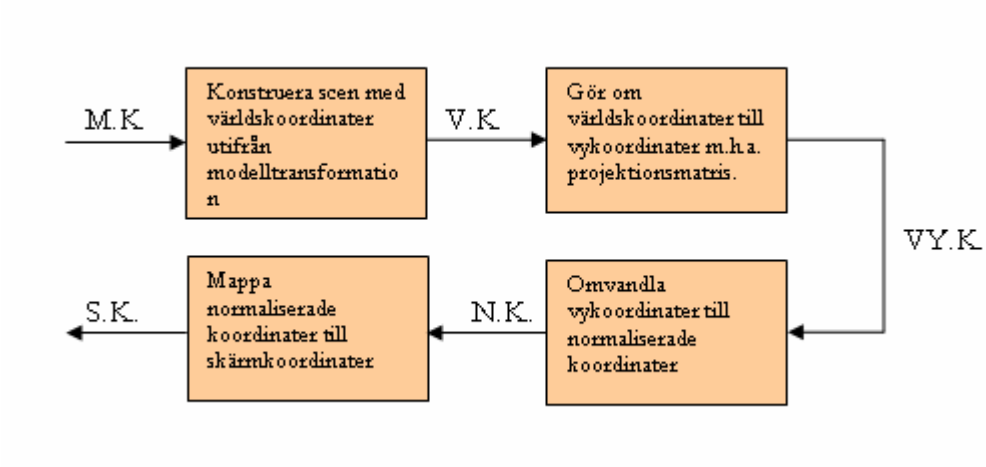
Figur 3-1 Princip för uppdelning

Figur 3-1 Princip för uppdelning. Anledningen till att den använder mycket minne är att den inte återanvänder punkter som den har skapat. Om man hade använt en mer avancerad datastruktur som till exempel *half-edged*, skulle man kunnat återanvända skapade punkter och därmed minskat minnesanvändningen. Eftersom varje triangel rekursivt skapar fyra trianglar innebär det att varje triangel skapar 4^n stycken trianglar, där n är antalet uppdelningar. Varje sida på triangeln delas upp med 2^n så för att dela upp en sida på en triangel som är 1000 meter till trianglar med sidan mindre än 25 meter behövs sex stycken uppdelningar, vilket ger 4096 trianglar som i sin tur ger en sidlängd på ungefär sexton meter. [3]

3.2.6 Avståndsmätning

Stöd för avståndsmätning fanns i tidigare version, dock inte fullständigt implementerad. Algoritmen för att göra detta kan grovt sett beskrivas som att man stegade sig fram längs med en riktning som räknades fram utifrån markerad skärmkoordinat och sedan kontrollerades det kontinuerligt om man hoppat förbi marken eller inte.

För att mäta avstånd i programmet används djupbufferten i OpenGL. När användaren klickar på skärmen hämtas ett värde för djupet ur djupbufferten för just denna skärmkoordinat. Sedan kan detta värde räknas om till objektkoordinater genom att veta aktuell modelviewmatris, projektionsmatris och viewportmatris. Detta kan göras enkelt med hjälp av GLU-funktionen *gluUnProject*. För att denna mätning ska fungera måste kamerans position hela tiden uppdateras och att det endast är terrängen som renderas till djupbufferten, ty annars kan objekt som inte ska finnas med i mätningen komma i vägen.



Figur 3-2 omvandling mellan modellkoordinater och skärmkoordinater.

Figur 3-2 illustrerar den vanliga omvandlingsprocessen från modellkoordinater till skärmkoordinater. Med *gluUnProject* går processen åt andra hållet, från skärmkoordinat till modellkordinat och använder de olika transformationsmatriserna för att åstadkomma detta. Här defineras modellkoordinater som ”M.K.”, världkoordinater som ”V.K.”, vykoordinater som ”VY.K.”, normaliserade koordinater som ”N.K.” och till sist skärmkoordinater som ”S.K.”.

3.2.7 Punkter i terrängen

För att representera intressanta punkter i terrängen används en teknik kallad *billboards*. Det finns flera typer av *billboards*. Den typ som används i systemet kallas *axial billboard* som ger intryck av en helt vanlig skylt som ritas ut i terrängen, men vars framsida roteras i riktning mot kamerans position. För att skapa en *axial billboard* måste man räkna ut en riktningsvektor mellan positionen på objektet och kamerans position sedan behövs en uträkning av en vinkel mellan den framräknade riktningsvektorn och kamerans riktningsvektor. Som uppvektor används världens uppvektor för att den inte ska roteras med kameran. [3]

3.2.8 Riktningsvisare

Den riktningsvisare som är implementerad i systemet använder sig av samma metod som i 3.2.7 förutom att det är pilen som roteras och inte objektet. Först räknas en riktningsvektor ut mellan pilen och objektet den ska peka mot. Genom att använda den inre produkten mellan riktningsvektorn och kamerans riktningsvektor kan vinkeln som pilen ska vridas räknas ut.

3.2.9 Utritning av höjddatabas

För att rita ut höjddatabasen kunde inte de värden som var angivna användas i den form de gavs eftersom de var för stora, vilket sedan resulterade i att grafiken började flimra. För att komma tillrätta med det problemet bestämdes centrum på området och därefter ritades alla värden

utifrån det referensvärdet. På så sätt användes istället bara värden mellan -5000 och 5000 vilket inte orsakade några synliga problem.

Utritningen av höjddatabasen utförs med hjälp av så kallade displaylists. Eftersom höjddatat är statiskt och inte ändrar sig under programmets exekvering kan *displaylists* användas. Fördelen med *displaylists* är att belastningen på processorn blir mindre, då de punkter som ska renderas redan har skickats till och sparats i minnet på grafikkortet vid programmets start.

3.2.10 Radiokommunikation

Radiokommunikationen är implementerad för att stödja kommunikation via DART. Metoder från ett färdigt bibliotek användes för att implementera de metoder som behövdes för att skapa, skicka och ta emot ett meddelande. För att skicka ett meddelande skickas först en förfrågan till PC-DART som säger vilket sorts meddelande som ska skickas. Som svar returneras en mall som motsvarar rätt meddelandetyp. Denna mall fylls med passande information som sedan är redo att skickas. [15]

3.2.11 Utritning av områden

Områden ritas ut som en polygon där man kan sätta ut ett hörn i taget mellan vilka det sedan ritas linjer. Hörnen på polygonen kan antingen klickas ut med enskilda musklick eller genom att efter ett klick hålla ned musknappen, för att på så vis kontinuerligt ha möjlighet att se polygonens utseende fram tills att man åter släpper upp knappen. Eftersom områden ska ritas ut med en transparent yta och följa marken måste det, som nämnts tidigare, delas upp i trianglar. Detta beskrivs i kapitel 3.2.4 Triangulering samt kapitel 3.2.5 Subdivision.

3.2.12 Skymda linjer

Runt hela området finns även en linje för att markera kanten bättre. Linjer som är befinner sig bakom annan terräng ritas ut som streckade samt i en annan färg. För att bestämma om linjer är dolda används en buffert i OpenGL som kallas för *stencil buffer*. Bufferten används för att jämföra värden för enskilda pixlar med ett heltalsvärde och på så sätt bestämma om den ska synas eller inte. Baksidan på ett område ritas ut så att värdena i stencilbufferten blir lika med ett. Sedan ritas kantlinjen ut där stencilbufferten har ett värde som är lika med ett, detta görs med en röd färg. Till sist kan kantlinjen ritas ut där värdet i stencilbufferten är skilt från ett med en blå färg.

4 Gränssnitt

I detta kapitel diskuteras hur ett generellt gränssnitt mellan Tactical Overlay System och godtyckligt ledningssystem skulle kunna tas fram med hjälp av olika sorters verktyg. Det diskuteras även vilka krav som ställs på ett sådant gränssnitt och presentationsansvar mot identifikationsansvar.

4.1 Inledning

Här kommer en jämförelse mellan två olika dataformat som kan användas för kommunikation mellan olika system. De två dataformat som diskuteras är *Exstensible Markup Language* (XML) och *Extensible Binary Meta-Language* (EBML). EBML är inte lika accepterad som XML men är med i jämförelsen på grund av att det är ett binärt format till skillnad från XML som är textbaserat. Både XML och EMBL är uppbyggda på samma sätt då syntaxen är skilt från semantiken, det vill säga att ett bibliotek för något av formaten kan läsa vilket format som helst baserat på det. Det är upp till programmet som använder formatet att tolka innehållet. Eftersom att båda formaten är lika, delar de många av sina fördelar samt nackdelar.

Här kommer även en diskussion finnas om vilken del av det färdiga systemet som ska stå för presentationsansvaret. Man måste försöka göra en plan om vilka tjänster som systemet ska tillhandahålla eftersom det bestämmer i stor grad var presentationsansvaret ska ligga. De olika alternativ som diskuteras är om presentationsansvaret ska ligga i servern, klienten eller om det helt enkelt ska vara en kombination av båda.

4.2 XML

XML är ett textbaserat format som bygger på *Standard Generalized Markup Language* (SGML). SGML används primärt för publicering av text och databaser. SGML, som mest används av erfarna programmerare, kan uppfattas som svårt att lära sig, på grund av ett fleratal olika orsaker. Den största orsaken är en väldigt generell och flexibel syntax som även medför att det blir svårt att utveckla en parser för formatet. XML är mycket enklare i den meningen att det har en hårdare definierad syntax. Som en följd av detta blir även parsern enklare, vilket har lett till att XML nästan helt har ersatt SGML. I XML-standarden definieras också någonting som kallas *XML*

Schema Definition (XSD). XSD beskriver strukturen på ett XML-dokument och kan användas för att verifiera att ett XML-dokument är korrekt.

4.2.1 Användning av XML

XML är ett textbaserat format som tillåter att användaren definierar egna taggar. Här följer ett exempel på hur man kan skriva ett meddelande i XML. Exemplet är inte tänkt att vara en guide på hur man skriver ett XML dokument. För en komplett definition av XML syntaxen se [19].

```
<?xml version=”1.0” encoding="UTF-8"?>
<message type=”123”>
          <id>område1</id>
          <coordinates>
                    <coordinate x=”7022100” y=”1640000”
/>
                    <coordinate x=”7022100” y=”1640100”
/>
                    <coordinate x=”7022000” y=”1640100”
/>
          </coordinates>
          <description>
                    Det här är ett område.
          </description>
</message>
```

Den första raden säger att det är ett XML dokument som följer standarden för XML-version ett och att den har UTF-8 som textkodning. Den andra raden säger att det data som XML dokumentet innehåller är en rottagg som alla XML dokument måste innehålla, *message*, den noden har ett attribut som säger vilken typ av meddelande det är av, i det här fallet *123*. Direkt efter rottaggen kommer en tagg som innehåller en identifierare till detta meddelande. Sedan kommer ett godtyckligt antal koordinater i det här fallet tre stycken. Sist kommer en beskrivning av meddelandet.

4.3 EBML

EMBL är ett relativt nytt format som är utvecklat till en container för video och ljud kallad *Matroska,* men det är inte begränsat till video och ljud. EMBL är designat att vara ett utbyggbart binärt format. En fördel gentemot XML är att EBML är ett binärt format som tar mindre plats än ett textbaserat. Men en nackdel med EBML är att det inte är lika utbyggbart som XML eftersom det kräver att ett *Document Type Definition* (DTD) är känt i förväg. DTD beskriver hur ett korrekt dokument ska vara uppbyggt. En annan nackdel som är gemensam för både EBML och XML är att de innehåller överflödig data. Däremot har EBML ett användbart tillägg som tillåter att standardvärden kan sättas för värden som inte ändras så ofta.

W3C, skaparna av XML, har utsett en arbetsgrupp som ska undersöka hur stora vinster man kan få om man utvecklar ett binärt format utöver det textbaserade. [20]

4.3.1 Användning av EBML

EBML kan vara besvärligare att använda eftersom att det är ett binärt format och inte är lika självförklarande som ett textbaserat format. EBML använder sig av taggar precis som XML, men i EBML kallas de för id. Ett id består av element id, data storlek och data. EBML använder sig av variabel heltalslängd, det används för att man ska kunna definiera storleken på data i en id och även för att definiera värden. Exemplet nedan visar ett meddelande som innehåller exakt samma data som XML exemplet. Även detta exempel är inte tänkt att vara en komplett guide för hur man använder EBML korrekt. För en komplett specifikation se [21]. I exemplet är två tecken en byte, förutom där det står i text där istället ett tecken representerar en byte. Mellanslag och radbryt är bara med för att göra exemplet läsligt.

```
1a45dfa3 86

4282 83 TOS

1b4569a5 c7
41a1 fb
4115 87 område1

4121 9e
82 406b2614
87 40190640
82 406b2614
87 401906a4
82 406b25b0
87 401906a4
4135 96 Det här är ett område.
```

Den första raden är ett EBML start id som säger att dokumentet startar där, ungefär som rotnod i XML, sedan följer en storlek på dokumentets huvud. Den andra raden säger vad det är för system dokumentet tillhör. På den tredje raden börjar själva meddelandet med ett id och resterande längden av meddelandet. Nästa rad definierar meddelandenumret, 123, som är skrivet som ett variabelt heltal. Sedan kommer ett id som definierar områdets id följt av storleken på strängen och till sist strängen. 82 och 87 är id för x-koordinat respektive y-koordinat med värden i variabel heltalslängd, de har en kortare id eftersom de kan förekomma så många gånger i dokumentet och därmed så sparas lite plats.

4.4 Ansvar för presentationen

Oberoende av om man väljer att använda XML eller EBML återstår frågan om presentationsansvar, ska servern eller klienten sköta presentationen av

grafiken? Det finns tre möjliga alternativ; servern ansvarar för presentationen, klienten ansvarar för presentationen eller en kombination av de två första alternativen. En kombination är i sin tur möjlig att sätta samman på olika sätt. Enklast vore en kombination där klienten definierar ett antal basfunktioner som servern kan välja bland men där den också har möjlighet att specificera vissa attribut ytterligare.

4.4.1 Server

När presentationsansvaret ligger hos servern måste klienten definiera ett grundutbud av funktioner för till exempel utritning av linjer, kvadrater, cirklar eller text. Funktionerna har inget definierat sätt som de måste användas på utan servern bestämmer själv hur de ska användas. Komplexiteten för ett sådant system skulle då hamna på servern och i gränsytan mot klienten. Eftersom det är serverns ansvar att tillhandahålla nödvändig data måste den konfigureras på ett sådant sätt att varje klient har tillgång till den information de behöver, vilket leder till att gränsytan blir mer komplex. En annan svårighet är att bestämma vilka basfunktioner som ska tillhandahållas, till exempel om ett navigeringshjälpmedel ska vara en av basfunktionerna eller om servern måste definiera sådant.

4.4.2 Klienten

Om hela ansvaret för prestationen ligger i klienten överförs komplexiteten till densamma. Istället får servern använda en definierad gränsyta mot klienten. Klienten kan definiera objekt såsom återsamlingsplats där servern inte har något och göra med hur det objektet ser ut, bara var det är lokaliserat i världen och eventuellt om det ska ha någon speciell text. Om det tillkommer ett nytt objekt går det inte bara att ändra färg eller form på ett liknade, redan existerande, objekt utan då måste det skapas ett nytt i klienten.

4.4.3 Kombinerad

Den tredje varianten är en kombination av server och klient där klienten definierar ett antal grundobjekt som servern kan använda. De har en fördefinierad betydelse och ett på förhand bestämt utseende men som servern ändå har möjlighet att ändra på. På detta sätt kan presentationen av objekt specialiseras och utformas tills viss del av klienterna givet ett grundutbud av funktioner i servern.

4.5 Ansvar för identifikationen

Även ansvaret för identifiering av olika objekt måste diskuteras eftersom det har en inverkan på vilken data som måste skickas till och från ett system som detta. Liksom presentationsansvaret är indentifikationsansvaret helt oberoende på vilket underliggande format som används mellan systemen. Med identifikationsansvar menas vilket system som ska identifiera och

hantera till exempel ett stridsfordons signatur från en IR-kamera. Var än identifikationsansvaret ska ligga så är det något som gränsytan måste ta hänsyn till. Detta för att kunna skicka information till andra system, exempelvis ett stridsledningssystem, om ett fientligt stridsfordons position borde det kanske skickas till alla andra stridsfordon inom gruppen.

4.6 Slutsats

Eftersom EBML är ett så pass nytt format och saknar stöd inom industrin så är valet mellan XML och EBML inte svårt att göra. Dessutom kan många utvecklare redan XML-syntaxen och därför är det onödigt att behöva lära sig något nytt. XML är också mycket lättare att felsöka eftersom att det är lätt att läsa. En fördel som EBML har är att det tar mindre plats än XML, men det är inte en avgörande faktor i detta fall eftersom systemet ska användas med Ethernet och inte har så väldigt stränga krav på storleken dessutom så finns det effektiva komprimeringsmetoder som lämpar sig för att komprimera XML. Man kan även argumentera att ett binärt format är snabbare att läsa in, men i det här fallet kommer inte detta att vara en stor sak eftersom varje meddelande inte kommer vara så stort och den mesta tiden i systemet kommer att gå åt till andra processer. Dock så ska man nog inte avfärda EMBL som ett format som aldrig kommer att användas till något annat än det var utvecklat för, men för just detta ändamålet passar XML bättre.

Vilken del som ska ta hand om presentationsansvaret är lite svårare att ge ett exakt svar på, men i ett sådant här system så är det viktigt att gränsytan är flexibel eftersom man kanske vill ändra betydelse eller utseende för att anpassa systemet till olika länders stridsledningssystem och eventuella nya objekt. Det skulle fungera med att bara tillhandahålla några grundfunktioner för grafikutritning i klienten, gränsytan skulle inte bli överdrivet komplicerad, men istället blir nog servern mer komplicerad än den behöver vara. För detta system är dock en kombinerad gränsyta den bästa lösningen, dels för att kunna hålla meddelandena relativt små men samtidigt för att kunna anpassa systemet till olika ändamål.

Ansvaret för identifikationen måste delas upp mellan olika system eftersom det inte är en bra ide att låta ett system som *Tactical Overlay System* ansvara för all identifikation, exempelvis så kan och analys av videobilden mycket väl överlåtas till ett sådant system eftersom det nog har tillgång till ett grafikkort. Men annan analys som inte har med grafik att göra bör göras av något annat system. I vilket fall som helst måste systemet åtminstone ge ett grundläggande stöd för att presentera de objekt som tas in från andra system.

5 Sensorfusion

Här redogörs allmänt för vad sensorfusion är och beskriver de olika kategorierna av sensorfusion. Sedan presenteras några av de vanligaste algoritmer relaterade till området, såsom Kalman filter, Bayesianska nätverk och Dempster-Shafer samt diskuteras huruvida de kan appliceras i ett system som Tactical Overlay System.

5.1 Introduktion

Sensorfusion [12, 23] innebär kortfattat att data från en eller ett flertal olika sensorer, vid olika tidpunkter, kombineras på ett sätt som gör att det slutgiltiga resultatet eller användandet av denna datakombination blir mer exakt, pålitlig eller på något sätt mer fullständig än om informationen från varje enskild sensor skulle ha använts var för sig. Sensorfusion kan indelas i tre undergrupper, direkt sensorfusion, indirekt sensorfusion samt fusion av dessa båda gruppers resultat.

Direkt sensorfusion innebär att information från ett antal olika homogena eller heterogena sensorer, virtuella sensorer eller tidigare registrerad information kombineras. Indirekt sensorfusion använder istället information som baseras på tidigare vetskap om miljön eller på exempelvis förväntad input. Exempel på sensorer som kan ingå i ett system som använder sig av sensorfusion kan vara radar, sonar, *Electronic-Support-Measures*-sensorer, olika sorters kameror, magnetiska sensorer och så vidare.

När sensorfusion ska appliceras i verkligheten (eller även så kallad datafusion) brukar den ofta falla inom ramarna för en eller flera av följande tre kategorier; *Low-level fusion*, *Feature-level fusion* eller *Decision fusion*. *Low-level fusion* är då rådata kombineras på något sätt, för att kunna generera en bättre och mer exakt information. Ett exempel på *Low-level fusion* är *Pixel-level fusion* som innebär att relevant information från en eller ett antal olika bildkällor förenas genom någon algoritmisk kombination av dessa. Den andra kategorin, *Feature-level fusion*, befinner sig ett abstraktionslager ovan *Low-level fusion*. Då extraheras istället så kallade features ur informationen som sedan kombineras på olika sätt för att förbättra resultatet. En *Feature* kan beskrivas som ett egenartat drag i informationsmängden, exempelvis kanter, linjer eller hörn i bilder. På den högsta nivån av kategorierna inom sensorfusion befinner sig *Decision Fusion* som kombinerar ett antal olika beslut utifrån så kallade expertkunskaper eller *a priori-kunskap*. *A priori-kunskap* är kunskap som innehas redan från början och som är baserad på slutsatser som gjorts

genom en deduktiv resonering utifrån ett antal premisser. Inom matematiken är det detsamma som teorem som bevisats utifrån en eller flera axiom eller andra teorem. Denna kunskap används sedan för att bestämma den optimala fusionen av beslut och data.

5.2 JDL - Datafusion

År 1991 släppte *Joint Directors of Laboratories* (JDL) en definition av en funktionell modell [13] som beskrev vilka de viktigaste funktionerna för datafusion är, vilken information som anses relevant, samt hur alla delarna i ett sådant system relaterar till varandra. Tre år senare omarbetades och utökades denna definition av *Australian Department of Defense* (DSTO) så att definitionen även omfattade fler-nivå-processer för hantering av problem som mönsterigenkänning, association, korrelation, uppskattning och kombination av information ifrån enskilda eller multipla källor. Denna definition har undergått ytterligare bearbetningar sedan 1994, men modellen nedan baseras alltså på 1994 års version.

Modellen beskriver datafusionsprocessen samt dess organisation och består av fyra stycken olika nivåer *Nivå 1 – Object refinement*, *Nivå 2 - Situation refinement*, *Nivå 3 – Threat refinement* samt *Nivå 4 – Process refinement.*

5.2.1 Nivå 1 – Object refinement

Detta är den första nivån av funktionsmodellen och delas ofta upp i fyra funktioner. Dessa är *data-alignment*, *association*, *tracking* och *identification.* Genom dessa funktioner kan informationen kombineras och ett objekts identitet, typ och beteende bestämmas. Resultatet av nivå 1 kallas *Situation picture.*

5.2.2 Nivå 2 - Situation refinement

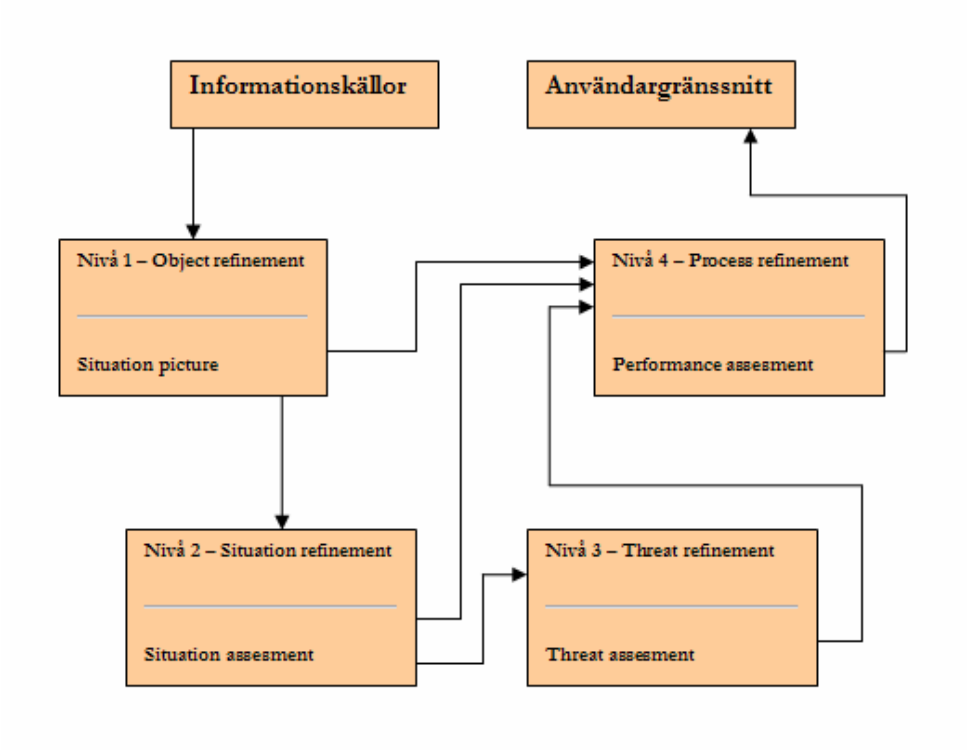
Här grupperas olika objekt efter deras temporala och spatiala relationer till varandra för att kunna göra en abstrakt tolkning av olika mönster i informationen. Resultatet av denna processnivå kallas *Situation assesment.*

5.2.3 Nivå 3 – Threat refinement

Tanken med nivå 3 är att iterativt koppla samman data om en fiendes aktiviteter och slagkraft för att kunna dra slutsatser om dess intentioner och att avgöra vilket hot de utgör, därav kallas resultatet från denna nivå för *Threat assesment.*

5.2.4 Nivå 4 – Process refinement

Den fjärde och sista nivån i funktionsmodellen används för att övervaka och utvärdera hela fusionsprocessen och för att sedan kunna använda denna information till att förfina och förbättra denna process. Detta nivåsteg får



Figur 5-1 JDL – processflöde

information från alla de andra stegen och resultatet som produceras kallas Process assesment.

Figur 5-1 illustrerar informationsflödet och output från varje nivå i funktionsmodellen framtagen av JDL och DSTO.

Det finns en antal olika hjälpmedel, metoder och processer som kan användas för att implementera fusion av data som exempelvis kan baseras på statistik eller numeriska metoder. Följande fyra sektioner beskriver några vanliga exempel på sådana.

5.3 Transducar-Markup-Language

Transducar-Markup-Language (TML) [9] är ett språk som kan användas för att beskriva ett system uppbyggt av en mängd olika komponenter såsom sensorer eller sändare/mottagare samt hur dessa relaterar till varandra, så väl logiskt som fysiskt. TML kan lagra övergripande information om ett systems struktur eller metadata i form av XML och även detaljerad information om datatyper, datastorlekar, mätnoggrannheter, tidsstämplar eller exempelvis specifika fysiska undantag för någon särskild sensor. Språket har en rad olika fördelar gentemot tidigare metoder då det gäller att implementera system med multipla informationskällor som ska kombineras. Exempelvis möjliggörs korrelation av information från ett antal olika sensorer som är registrerad vid olika tidpunkter, eftersom tidstämplar lagras specifikt för

varje enskild datakälla och både fysiska och logiska relationer i systemet kan modelleras. All denna sensorspecifika information sammanställs sedan i en realtidsdataström och kan, eftersom informationen redan finns representerad lokalt, kraftigt reducera tiden som krävs för att ta hand om och representera aktuell data. En annan fördel är att information från sensorer är i samma format vilket gör informationssökningar och analyser lättare att utföra. I och med att varje dataelement, samt systemets metadata, är tidsstämplade blir det lättare att kontrollera systemets tillstånd och ger möjlighet till noggrannare sökningsresultat.

5.4 Bayesianska nätverk

Bayesianska nätverk [10, 18, 23] är ett verktyg som används inom området datafusion och med hjälp av sådana nätverk kan man bygga upp en probabilistisk grafisk modell. Modellen beskriver sannolikheten för en mängd olika variabler samt hur dessa beror av varandra. Bayesianska nätverk använder sig i grunden av Bayes teorem [16] för sannolikhet och beskrivs med hjälp av en DAG (*Directed-Acyclic-Graph*), där noderna i grafen beskriver variablerna och kanterna de relativa beroendena mellan noderna.

Bayes teorem definierar förhållandet, när det gäller obetingad så väl som betingad sannolikhet, för två stokastiska variabler A och B så som:

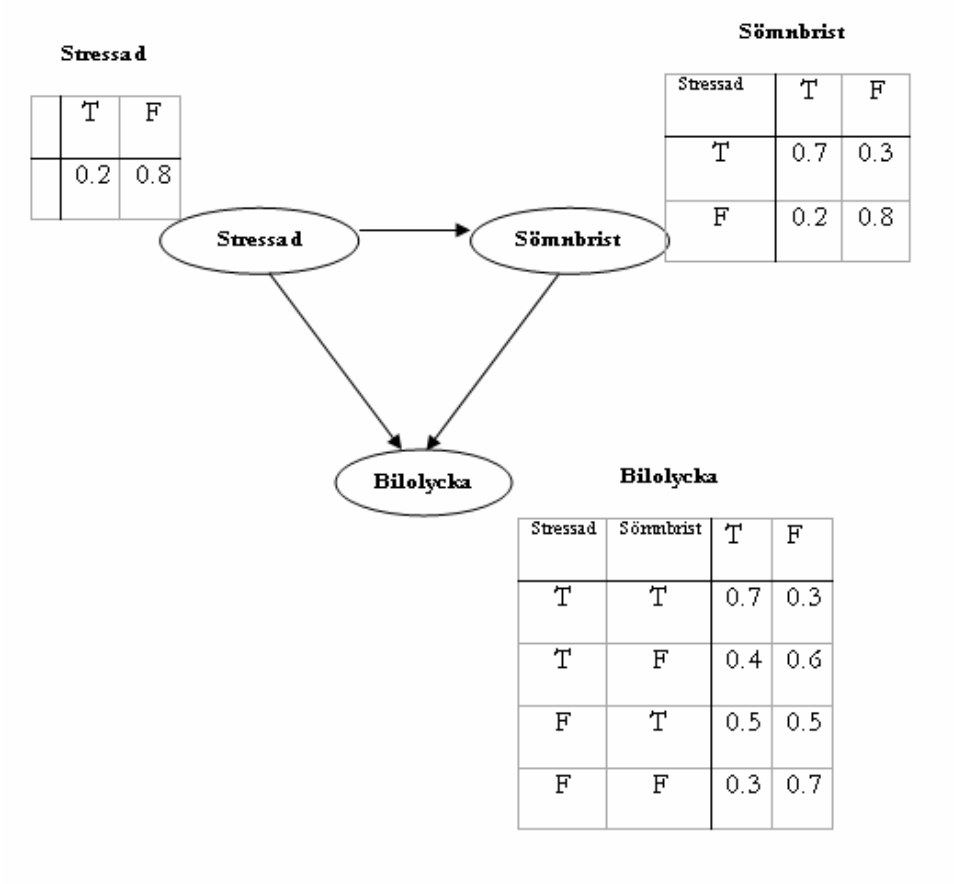
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

P(A) står för den obetingade sannolikheten för A och P(B) den obetingade sannolikheten för B. P(A|B) är den betingade sannolikheten för A givet B och P(B|A) är omvänt den betingade sannolikheten för B givet A.

Ett Bayesianskt nätverk definieras som en DAG där den totala distributionen är densamma som produkterna av varje nods, samt dess förälders, lokala distribution.

$$\mathrm{P}(\mathrm{X}_1,\ldots,\mathrm{X}_n) = \prod_{i=1}^{n} \mathrm{P}(\mathrm{X}_1 | parents(\mathrm{X}_i))$$

Om det går en kant från en nod A till en annan nod B, betyder det att A är en förälder till B. På samma sätt definieras omvänt B att vara ett barn till A. Vidare sägs en nod som saknar förälder vara *unconditional* och i annat fall *conditional*.



Figur 5-2 exempel på Bayesianskt nätverk

Figur 5-2 visas ett Bayesianskt nätverk och hur två olika händelsers probabilitet att inträffa skulle kunna påverka en tredje. Om man antingen är stressad eller har sömnbrist bidrar det i sig, enligt exemplet, inte till en större risk att råka ut för en bilolycka. Men om man är stressad samtidigt som man lider av sömnbrist ökar risken för en bilolycka markant. Notera att värdena i figur 5-2 endast är fiktiva.

Problemet med att använda sig av Bayes teorem är svårigheten att förse modellen med lämpliga initiala sannolikhetsvärden för varje möjligt fall. Oftast brukar dessa värden sättas till att vara lika och att sedan använda teoremet, tillsammans med en iterativ process, för att i varje iteration räkna ut nya värden. Dessa värden används sedan i nästa steg som nya begynnelsevillkor vilket leder till att resultaten för varje iteration blir mer meningsfulla.

5.5 Kalman-filter

Kalman-filter [6, 11, 23] är en rekursiv metod som används för att göra en effektiv och precis uppskattning av ett dynamiskt systems nuvarande tillstånd. Filtret använder sig av en serie uppmätta värden som kan vara ofullständiga eller missvisande. Filtret togs fram i slutet av 1950-talet och början på 1960 av personer som Thorvald Nicolai Thiele, Peter Swerling och inte minst Rudolf E. Kalman, som filtret även har låtits bli uppkallat efter.

Idag finns det en rad olika typer av Kalman-filter där den vanligaste typen är den så kallade *Phase-Locked-Loop* (PLL). En PLL-krets används i en sluten, återkopplad krets för att fixera och låsa en genererad signal till en viss frekvens, som motsvarar en yttre referensfrekvens. Genom att kontinuerligt justera och jämföra den genererade signalen mot referenssignalen kommer den så småningom, när det gäller både signalens frekvens liksom fas, att matcha referensens. PLL-kretsar förekommer främst i radioapparater, datorer och inom telekommunikation.

Kalman-filtret kan utifrån en uppskattning av systemets tillstånd i föregående tidssteg och uppmätta värden i det aktuella tidssteget, användas för att räkna ut det uppskattade tillståndet kommande tidssteg. Denna process brukar delas upp i två faser; *Predict*, *Update*.

Predict

Systemets förutsedda tillstånd, där $\hat{x}_{n|m}$ står för systemets uppskattade tillstånd vid tiden *n*, givet alla tidigare observerade värden fram till och med tiden *m*:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} B_k u_k \ .$$

Den förutsedda uppskattningen av systemets noggrannhet där $P_{n|m}$ är systemets kovariansmatris vid tiden *n*, givet alla tidigare observerade värden fram till och med tiden *m*:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \ .$$

Update

Mätvärdesresidual:

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} .$$

Kovariansresidual:

$$S_k = H_k P_{k|k-1} H_k^T + R_k .$$

Kalman gain:

$$K_k = P_{k|k-1} H_k^T S_k^{-1} .$$

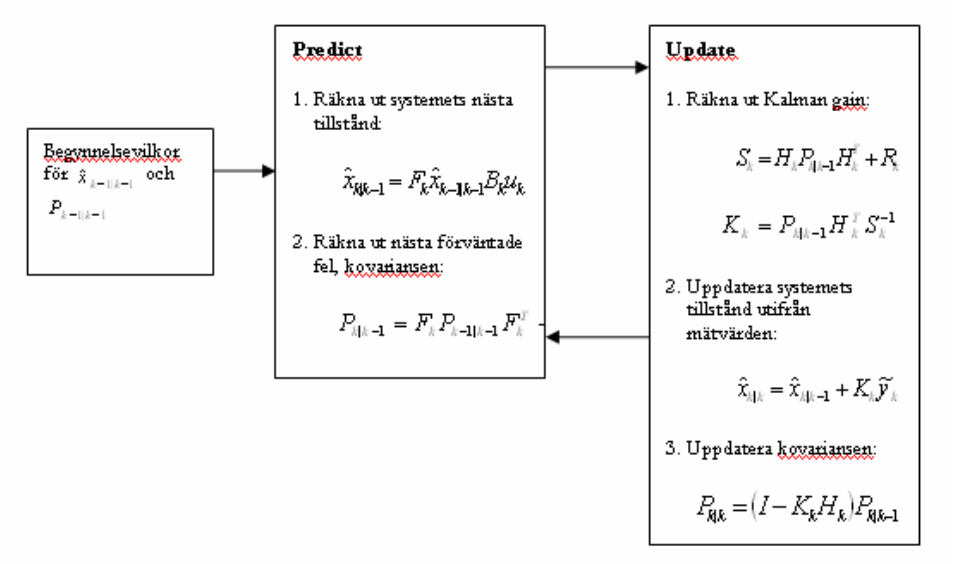
Uppdaterad tillståndsuppskattning:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k .$$

Uppdaterad kovariansuppskattning:

$$P_{k|k} = \left(I - K_k H_k\right) P_{k|k-1} .$$

Matrisen H innehåller information som relaterar det aktuella tillståndet till de uppmätta värdena och R utgör dessa värdens kovarians.



Figur 5-3 Kalmanfiltrets process

Figur 5-3 illustrerar hur processen i ett Kalman-filter fungerar och i vilken ordning formlerna ovan appliceras.

5.6 Dempster-Shafer

Demster-Shafer [17, 18, 23] är en generalisering av den vanliga bayesianska sannolikhetsteorin och används för att matematiskt bestämma bevis utifrån termerna *stöd* och *förtroende*.

5.6.1 Definitioner

Om M är den totala mängden tillstånd som systemet omfattas av sägs P(M) vara ett *power set* om P(M) innehåller alla möjliga delmängder av M, där den tomma mängden är inkluderad. Exempelvis är P(M) = {{ø}, {1} ,{3},{1, 3}} om M = {1, 3}.

Vidare tilldelas varje delmängd av P(M) en massa för dess stöd, så kallad *Belief mass*, via en överföringsfunktion som i sin tur kallas *Basic-Belief-Assignment*

$$m : P(M) \rightarrow [0,1]$$

då den verifierar följande två axiom:

1. $m(\text{ø}) = 0$

2. $$\sum_{A \varepsilon \mathrm{P}(X)} m(A) = 1$$

Det axiom 1 säger att massan för den tomma mängden ska vara lika med 0 och det axiom 2 att summan av ”stöd-massan” för ett påstående A, som tillhör ett *super set* P(M), är lika med 1.

Stöd (*Support* eller *Belief*) definieras för en mängd A som:

$$spt(A) = \sum_{B|B \subseteq A} m(B) .$$

Det vill säga att stödet för A är summan av massan för alla möjliga tillstånd som utgör en delmängd av A.

På ett liknande sätt formuleras definitionen för förtroende (*Plausibility*):

$$pl(A) = \sum_{B|B \cap A \neq \emptyset} m(B) .$$

Definitionen betyder att *förtroendet* för A är summan av massan för mängden av alla tillstånd som utgör snittet till A.

De båda uttrycken förhåller sig till varandra som:

$$pl(A) = 1 - spt(\overline{A})$$

Dempster-Shafer skiljer sig från det bayesianska sättet att räkna på sannolikhet, vilket framgår av definitionerna för stöd och förtroende. Detta eftersom att Dempster-Shafer, istället för att representera olika grader av sannolikhet som en bayesiansk sannolikhetsfördelning, uttrycker detta genom dess överföringsfunktion. Vidare tilldelas värden för sannolikhet inte för ett enskilt fall utan för en mängd av möjliga utfall. Detta leder till problemet att de olika utfallen på något sätt måste kombineras till ett gemensamt resultat.

5.6.2 Dempsters kombinationsregel

För att kombinera en mängd olika utfalls tilldelade massor används Dempsters kombinationsregel (*Dempster's rule of combination*).

Definitionen av denna kombinationsregel för massorna m_1 och m_2 lyder:

$$m_{1,2}(\text{ø}) = 0$$

$$m_{1,2}(A) = \frac{\sum_{B \cap C = A \neq \text{ø}} m_1(B) m_2(C)}{1 - \sum_{B \cap C = \text{ø}} m_1(B) m_2(C)}.$$

Här utgör nämnaren i formeln en normaliseringsfaktor och är även ett mått på hur stor konflikt som råder mellan de olika mängderna m1 och m2. Som ett resultat av att denna normaliseringsfaktor helt och hållet eliminerar all eventuell påverkan av konflikter mellan de mängder som ska kombineras, kommer uttrycket att kunna ge motsägelsefulla resultat. Detta leder till att det blir svårt att dra slutsatser av resultaten då konflikten mellan mängderna är stor.

5.7 Diskussion

De metoder som presenterats är endast exempel på metoder som används för att genomföra fusion av data. Att applicera dessa på ett projekt som Tactical Overlay Systems skulle dramatiskt kunna utöka möjligheterna till användbar och intressant funktionalitet. Många av metoderna har utvecklats inom försvarsindustrin där målsökning eller måligenkänning är vanliga exempel på användningsområden som skulle kunna tillämpas på denna form av projekt. Andra områden där datafusion används ofta är robotteknik, medicinteknik och rymdteknik. Det man vill åstadkomma med fusionen kan ha olika ändamål men det kan röra sig om tracking, detektering av olika slag, identifiering eller olika former av beslutstagande. Valet av metod beror mest på ändamålet och styrs likaså av kvaliteten på tillgängliga begynnelsevillkor.

6 Förstudie

Kapitel 6 är en sammanställning av en intervju med potentiella slutanvändare där intressanta funktioner för ett system som detta diskuterades. Två större indelningar görs bland viktiga funktioner.

Innan implementationen av demonstratorn påbörjades intervjuades ett antal personer. Personerna i fråga har haft stor erfarenhet av att framföra stridsfordon i skarpa situationer och anses vara en lämplig publik. Tanken är att de som möjliga slutanvändare ska kunna besvara frågor som exempelvis vilka funktioner som är önskvärda av ett system av denna sort och på samma gång ge oss en bild av den problematik som kan uppstå i dessa skarpa situationer. Efter detta möte kunde två indelningar av vad de tillfrågade personerna tyckte var viktigt göras; *Force Protection*, *navigationsstöd.*

6.1 Force Protection

Force Protection är den första av indelningarna som gjordes vid mötet och kanske även den viktigaste. Termen syftar på att den egna styrkans säkerhet ska vara så god som möjligt och att den inte ska behöva utsättas för onödiga risker, denna del ansågs vara väldigt viktig och stöd för detta i ett system som Tactical Overlay System en självklarhet.

6.1.1 Markering av vänliga styrkor

Vid mötet var samtliga intervjuade personer överens om att det ibland kunde upplevas som svårt att skilja vänner från fiender i vissa situationer. För att undanröja detta problem ville de att systemet hela tiden skulle kunna visa var vänliga styrkor befinner sig för att på så vis undvika att till exempel beskjuta någon av dessa. Identifikation för styrkorna skulle exempelvis kunna vara fordonsnummer, anropssignal (call-signs) eller liknande.

6.1.2 Indirekta eldområden och skjutgränser

Att kunna markera ut indirekta eldområden var också en önskvärd funktion. De intervjuade menade även att det vore smidigt i fall diverse information kunde visas och presenteras i bilden tillsammans med det aktuella området. Information om själva området eller skjutgränsen, som vilken tidpunkt området ska intas eller områdets giltighetsperiod, vilken typ av pjäs som ska användas, agenda och så vidare.

6.1.3 Mineringar

Personerna tyckte även att information om potentiella minfält borde kunna utnyttjas och presenteras. Av samma orsak som i föregående sektion var de av den åsikten att det skulle underlätta för personalen om ett minfält kunde presenteras på något sätt eller om personalen istället automatiskt kunde bli varnade då de kommer i närheten av ett.

6.1.4 Oexploderad ammunition

Oexploderad ammunition (OXA) kan vara odetonerade sprängladdningar, granater eller vilken ammunition som helst som inte har exploderat men som kan utgöra en risk för militären i dess närhet. För att kunna skydda vänliga styrkor från denna fara var även möjligheten att kunna markera ut OXA en önskvärd funktion.

6.2 Navigationsstöd

Annat som det visades stort intresse för under intervjun var funktioner som kan underlätta navigeringen. Till dessa hör till exempel markering av olika sorters poster eller platser som kan vara av intresse. Det kan vara sjukvårdsplatser, återsamlingsplatser, reparationsplatser, fältarbeten, utgångsplatser i terräng, farbara vägar eller farbara broar och övergångar.

Andra typer av stöd för navigering diskuterades också. Exempel på sådana orienteringshjälpmedel kan vara en pil som pekar mot ett valfritt mål, en kartbild uppifrån som visar vagnen i dess omgivande terräng, kamerans möjliga synfält och närliggande objekt, avståndsmätning utan att använda laser och hjälp med triangulering av fientlig aktivitet.

6.3 Sammanfattning

Sammanfattningsvis ska ett sådant system kunna ge möjlighet att skydda vänliga styrkor och göra det lättare att urskilja samt samverka med dessa. Vidare är olika former av stöd vid navigering en viktig funktion där olika sorters markeringar eller annan viktig information ska kunna presenteras för att hjälpa personalen att framföra fordonet bättre.

7 Demonstrator

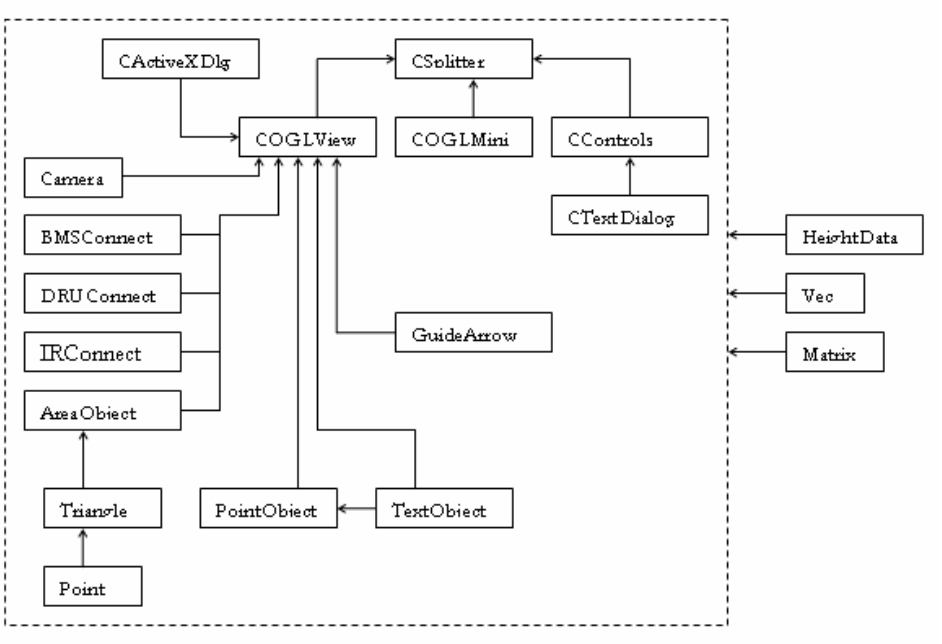
I detta kapitel beskrivs den demonstrator som implementerades och hur dess underliggande systemdesign ser ut. Varje klass beskrivs kortfattat, vilka funktioner som implementerades och anledningen till varför dessa valdes.

Demonstratorn som är implementerad i examensarbetet är en prototyp som kan visa intressanta områden och platser i terrängen, skicka markeringar, ta emot markeringar och slutligen att mäta avstånd i videobilden. Den nya demonstratorn skiljer sig väsentligt från den ursprungliga eftersom den har ett flertal nya användbara funktioner. Dessa är mer praktiskt tillämpbara och använder sig av ett färdigt bibliotek för två- och tre-dimensionell grafik, OpenGL.

7.1 Systemdesign

Eftersom det fanns en klar bild över vilka funktioner denna demonstrator skulle kunna tillhandahålla och vetskapen att systemet varken skulle komma att användas i något annat syfte än som demonstrator eller vidareutvecklas, så valdes en rudimentär systemdesign.

Det främsta hindret då det gäller utbyggbarhet är att ingen generell datarepresentation för objekt finns. Istället lagras dessa objekt, som i denna demonstrator endast kan vara områden eller punkter i terrängen, i två separata listor. Ett alternativ till denna lösning vore en såkallad scengraf, vilket skulle möjliggöra godtyckliga objektstyper, göra det enkelt att utföra eventuella transformationer på dessa och en smidigare traversering och uppdatering av befintliga objekt i systemet.



Figur 7-1 systemets design.

Figur 7-1 beskriver systemets underliggande design och struktur.

Systemet är uppbyggt av 19 klasser som beskrivs i följande sektioner.

7.1.1 CActiveXDlg

Denna klass används som wrapper-klass för ActiveX kontrollen som följer med videoservern och ärver av *CDialog*. På grund av detta får den alla egenskaper som ett vanligt dialogfönster har i MFC och kan därför utnyttja ActiveX. Dialogen är dold men körs i bakgrunden för att hämta enskilda bilder från videoservern, se sektion 3.2.1 och 3.2.2. När en ny bild har kommit in till videoservern och den är färdigbehandlad generas en händelse som hanteras av metoden *OnNewImageAxismediacontrol1*. Det enda denna metod gör är att sätta en flagga som visar att det finns en ny bild att hämta.

7.1.2 AreaObject

AreaObject representerar ett område som exempelvis kan vara ett eldområde, minfält eller något annat som bör uppmärksammas. Ett område är uppbyggt av ett antal punkter som bildar områdets kontur, ett antal Triangle-objekt som utgör själva ytan och ett TextObject-objekt som används för att presentera information om området såsom centrumpunkt, avstånd etc. För vidare information om hur området delas upp i trianglar utifrån given kontur, se sektion 3.2.4.

7.1.3 BMSConnect

BMSConnect ärver av *CSocket* och kopplar upp mot en server när den skapas. Den överlagrar också metoden som körs då något tas emot från servern. När ett meddelande tas emot skickas det till en metod som tolkar det och skapar ett områdesobjekt. För att skicka ett meddelande används metoder i klassen *CSockets*. För mer information om hur meddelanden skickas se sektion 3.2.10.

7.1.4 Camera

Denna klass används för att simulera hur en kamera kan roteras och positioneras. Objekt av denna klass kan representera både rörelse för DRU samt en virtuell förstapersonskamera. Genom att lagra information specifikt för de båda kamerorna kan man enkelt hoppa mellan dessa, beroende på om bilden ska visas sett från positionen för DRU eller från positionen för den virtuella kameran. Funktionalitet för att rita ut kamerornas frustum finns implementerad i funktionen *drawFrustum*. Frustumet räknas ut utifrån aktuell kameras position, up-vektor och riktningsvektor. Det utgörs av den volym som begränsas av *near*-, *far*-, *top*- och *bottom*-clipping plane och är alltså det möjliga synfältet.

7.1.5 CControls

I programmet finns en enkel meny med ett antal knappar för att exempelvis välja utritning av områden, registrering av mål och markering av sjukvårdsplatser samt välja kameraläge eller avståndsberäkning. All denna funktionalitet hanteras i klassen *CControls*. Klassen ärver av *CFormView* och fungerar som en enkel tillståndsmaskin som håller reda på vilket tillstånd den är i och vilken funktion som används.

7.1.6 DRUConnect

Denna klass används för att hämta ut information om kamerans position, rotation och riktning. Genom att ärva av MFCs klass *CWinThread* kan en ny tråd startas som kontinuerligt uppdaterar denna information utan att inverka på andra delar av programmet.

7.1.7 GuideArrow

En önskvärd funktion i detta demo var att det skulle finnas någon sorts mekanism för att hjälpa användaren att hitta områden. En instans av *GuideArrow*-klassen gör just detta genom att utifrån kamerans position och det valda objektets centrumposition räkna fram en riktningsvektor. Denna riktningspil har även en färg som sätts beroende på vilken typ av objekt den pekar mot. Exempelvis är pilen grön om den pekar mot ett vanligt område, röd om den pekar mot en sjukvårdsplats och blå när det gäller en målregistrering. Denna koppling mellan färg på pilen och typer på områden är ett naturligt sätt att göra det tydligt för användaren vilken objektstyp pilen pekar mot, speciellt då objektet är utanför vyn.

7.1.8 HeightData

Denna klass har metoder för att kunna läsa in och lagra höjddata från en höjddatabas med RT-90-koordinater samt hämta ut höjddata genom två olika metoder. En metod som hämtar ut ett värde från databasen som det är och en där värdet är linjärt interpolerat med närliggande värden, detta för att kunna få en höjd för en godtycklig koordinat.

7.1.9 IRConnect

Liksom klassen *DRUConnect* ärver *IRConnect* av *CWinThread*. Information som position och storlek för objekt som registrerats av IR-kameran skickas fortlöpande från Blackfin-processorn och måste därför likaså kunna hanteras utan att exekveringen av de andra delarna i programmet störs. Data kommer in på COM-porten, parsas och relevant information lagras därefter kontinuerligt. För att förhindra att felaktig information ska hämtas av någon annan tråd är dessa variabler oåtkomliga under tiden det sker en uppdatering. Annars finns en risk att endast några av variablerna har blivit uppdaterade då data hämtas och kan därför innehålla felaktig information.

7.1.10 Matrix

Klassen *Matrix* är en matematikklass som ger stöd för de mest basala matrisoperationerna. Några av dessa är matris-matrismultiplikation, skalär-matrismultiplikation, skapande av identitetsmatris samt rotationsmatriser.

7.1.11 COGLView

COGLView utgör huvudfönstret för utritning av video och grafik. Utritningen sker med hjälp av OpenGL, vilket det står mer om i sektion 3.2.2. Här hanteras även musklick då användaren ska rita upp områden eller mäta upp avstånd. Genom att använda metoder för att omvandla skärmkoordinater till objektkoordinater kan ett värde i terrängen hittas som motsvarar den specifika skärmkoordinaten. Klassen ärver av *CView* vilket är ett måste för att kunna använda OpenGL i MFC. Metoden *RenderSub* har hand om utritningen av grafiken i det mindre fönstret som visar toppvyn.

7.1.12 COGLMini

Eftersom det var önskvärt att kunna ha två olika sorters vyer på samma gång, en vanlig vy från kamerans perspektiv samt en toppvy, behövdes det två fönster att presentera detta med i. *COGLMini* ärver av *CView* och sätter precis som *COGLView* upp en *render context* med stöd för OpenGL. Till skillnad från *COGLView* är kameran positionerad högt upp, riktad rakt ned mot marken i stället för i samma position och med samma riktning som DRU, metoder för att beräkna positioner och avstånd utifrån användarens mustryckningar är lik de motsvarande i *COGLView* men behövde anpassas.

7.1.13 Point

Denna klass representerar en koordinat med ett x-, y- och z-värde, ett *vec3* objekt. *Point* har två referenser till andra *Point*-objekt och bildar på så vis en

länkad lista av koordinater. Denna lista utnyttjas sedan i trianguleringsfunktionen för att göra algoritmen lättare.

7.1.14 PointObject

PointObject används för att representera sjukvårdsplatser och registreringar av mål. Den har en position, en typ som bestämmer om den är en sjukvårdsplats eller en målregistrering och sedan två texturer för respektive typ. Beroende på vilken typ objektet ska representera så väljs motsvarande textur. När avståndet mellan kameran och objektets position är mindre än 1000 meter presenteras en informationstext för detta *PointObjekt* automatiskt. Avståndet uppdateras kontinuerligt genom att TextObject-objektet uppdateras i varje bildsekvens.

7.1.15 CSplitter

CSplitter ärver av klassen *CMDIChildWnd* och används för att kunna hantera multipla dokument i ett och samma fönster. Genom att använda ett *CSplitterWnd*-objekt kan ett fönster delas upp i flera olika vyer. Användargränssnittet i detta system är indelat i tre olika vyer där den första utgör huvudfönstret, den andra knappmenyn och den tredje toppvyn. Det är även här alla dessa vyer initieras via metoden *CreateView* i *CSplitterWnd*.

7.1.16 CTextDialog

Denna klass ärver av *CDialog* och används för att kunna lägga till information om ett område eller sjukvårdsplats innan det placeras ut. Denna text visas i samband med ett *TextObject* som finns tillagt i exempelvis ett *AreaObject* eller *PointObject*.

7.1.17 TextObject

Objekt som är instanser av klassen *TextObject* används för att kunna visa upp information i form av text. Texten som ritas ut är i själva verket bitmaps och skapas genom att göra en mängd olika display-lists, en för varje tecken, som sedan anropas då texten ska renderas. Vid utritning av denna text finns det även stöd för enkel formatering såsom radbrytning eller maximal bredd eller höjd. För att texten ska vara positionerad på rätt koordinat, skärmkoordinat, måste hela tiden den verkliga koordinaten, objektkoordinaten, kontinuerligt räknas om till motsvarande skärmkoordinat. Detta görs i metoden *updateScreenPos* på ett sätt som liknar det i metoden för att omvandla en skärmkoordinat till en objektkoordinat i *COGLView* eller *COGLMini*. Se sektion 3.2.6.

7.1.18 Triangle

För att kunna beskriva ett område används ett flertal *Triangle*-objekt och dessa objekt består i sin tur av tre koordinater samt referenser till fyra möjliga trianglar, i fall den har blivit uppdelad i ytterligare undertrianglar. Hur denna triangulering fungerar beskrivs närmare i kapitel 3.2.5 Subdivision.

7.1.19 Vec

Vec är en klass som representerar en vektor med två, tre eller fyra element. Likt *Matrix*-klassen finns de viktigaste funktionerna implementerade för att kunna multiplicera, normalisera, addera, subtrahera och räkna ut längd på dessa vektorer på ett smidigt sätt.

7.2 Funktioner

De funktioner som är implementerade i denna demonstrator är baserade på den förstudie som återfinns i kapitel 6 och är endast ett urval av de som ansågs mest intressanta samt praktiskt tillämpbara på detta projekt.

7.2.1 Två vyer

Demonstratorn är uppdelad i två huvuddelar, en vy som innehåller videobilden samt en toppvy. Detta implementerades eftersom det uttrycktes ett intresse att kunna överblicka omgivningen samt att snabbt kunna orientera sig efter någon form av karta.

7.2.2 Laserfri avståndsmätning

En funktion som ingår i detta examensarbete, samt i det föregående arbetet, är laserfri avståndsmätning. Istället för att använda sig av laser, då det finns en risk att avslöja sin position, används höjddatabasen för att mäta upp avstånd till punkter i terrängen som markerats i videobilden. En ny metod har utvecklats istället för den som togs fram i det tidigare examensarbetet.

7.2.3 Dynamisk markering i videobild/terräng

Det finns möjlighet att markera ett område i videobilden. Denna funktion fanns även med i det första examensarbetet men den var då statisk, det vill säga att det bara gick att läsa in områden från fil vid programmets start. Vidare kan olika sorters poster placeras ut i terrängen, bland annat sjukvårdsplatser med tillhörande beskrivande text.

7.2.4 Vägvisare

Det finns även en funktion i videobilden som gör att man kan visa en pil som hjälper till med navigeringen genom att den pekar mot ett visst av användaren angivet objekt. Bakgrunden till implementationen av denna funktion var att de intervjuade ibland kunde uppleva det som besvärligt att orientera sig mot en viss punkt i världen då de var tvungna att både hålla koll på hanteringen av fordonet samt vart de var på väg samtidigt. Presentationen av denna information sker ofta på någon form av karta, en vanlig karta eller en digital karta på en annan skärm.

7.2.5 Automatisk information

Den sista funktionen som är implementerad i videobilden är en informationsruta där det står viss information om ett objekt som användaren

kan välja. Den visas endast om man är närmare än 1000 meter. Demonstratorn har även ett begränsat stöd för kommunikation via DART. Den kan skicka registrering av mål samt ta emot markering av skjutgränser.

I toppvyn kan man se åt vilket håll kameran är riktad. Även om denna funktion inte efterfrågades i början var det en funktion som valdes till eftersom det blir mycket enklare att orientera sig. I toppvyn kan man likaså markera områden, sätta ut sjukvårdsplatser och mäta avstånd till en punkt. Man får även upp information om det objekt som är valt, men till skillnad från informationsrutan i videobilden så visas den hela tiden i toppvyn, eftersom den inte skymmer sikten.

8 Resultat

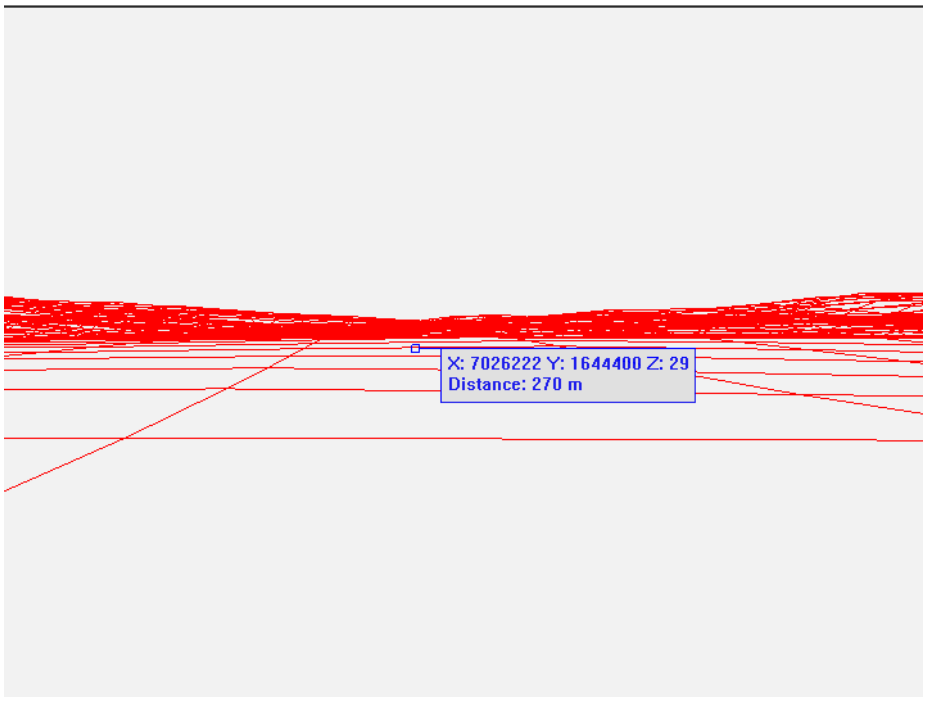
Här diskuteras hur det slutgiltiga resultatet blev och hur bra funktionerna som valdes att implementeras till slut fungerade.

8.1 Uppdateringsfrekvenser

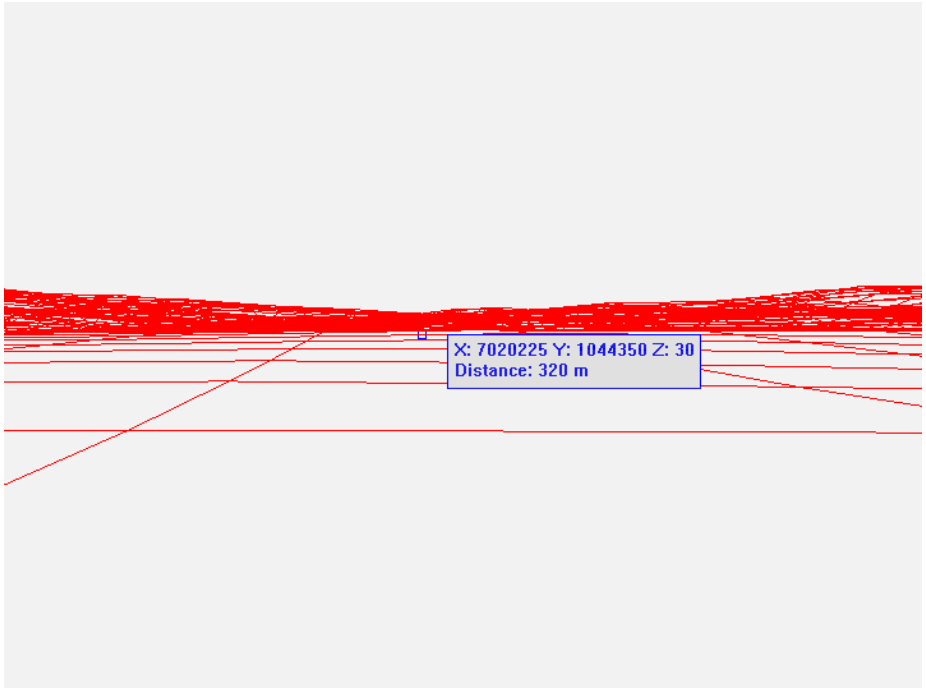
Uppdateringsfrekvensen av grafiken är acceptabel för de tester som är gjorda. Eftersom demonstratorn är implementerad i ett färdigt grafikbibliotek med hårdvarustöd innebär det att prestandan för grafiken är hög. Uppdateringsfrekvensen för den tredimensionella grafiken ligger på ungefär 30 bilder per sekund vilket gör att den inte känns långsam eller efter på något sätt. Kamerans uppdateringsfrekvens är 25 bilder per sekund vilket inte bör leda till att den känns långsam. Kameran är kopplad via en videoserver till datorn och introducerar en fördröjning på ungefär en sekund, vilket är oacceptabelt. Det finns även ett problem huruvida bilden kopieras från videoserverns ActiveX-kontroller. Detta problem leder till att en onödigt stor del av processorns tid går åt. Uppdateringen av tröghetsnavigatorn är också långsam. Exempelvis syns detta problem när demonstratorn utsätts för hastiga rörelser. Eftersom grafiken ritas direkt på videobilden uppstår inget flimmer och därmed är den slutgiltiga bilden behaglig att se på, vilket är en klar förbättring från tidigare version.

8.2 Avståndmätning

Precisionen av avståndsmätningen beror av ett flertal faktorer. Med den nuvarande metoden för att mäta längd kommer aldrig tillförlitliga resultat att kunna fås på flack terräng. Om den position man befinner sig på ligger nära marken uppstår stora problem med perspektivet. En längd kan skilja på flera hundra meter om man bara klickar några få pixlar högre upp i videobilden, problemet illustreras i och . Dagljuskamerans optik har också stor inverkan på längdmätningen eftersom inte några exakta värden för dess egenskaper finns. Därför används uppskattade värden för perspektivmatrisen i OpenGL för att få grafiken att överensstämma med det dagsljuskameran visar. Därför kommer grafiken inte visas helt korrekt. Höjddatabasen har också begränsningar i noggrannheten, vilket medför att precisionen på platt terräng blir ännu mer lidande. Det existerar även ett problem med vagnens altitud i världen. Om man åker nedför en backe händer det att tröghetsnavigatorns position uppfattas vara under markytan.



Figur 8-1 perspektivproblem bild ett.



Figur 8-2 perspektivproblem bild två.

Hårdvaran och de metoder som används i OpenGL för avståndsmätningen är inte heller helt perfekt. Detta fel är, vilket diskuteras vidare nedan, i jämförelse med de andra felfaktorerna i detta projekt mer eller mindre försumbart.

Felet som introduceras genom djupbuffertens begränsade precision vid avståndsmätningen kan uppskattas enligt följande formel. [22]

$$\Delta = z * z \Big/ \left(zNear * \left(1 << N\right) - z \right)$$

```
∆ = Minsta möjliga djupvärdet som kan urskiljas vid ett
visst avstånd.

z = Distans till objekt.

zNear = Avståndet till near-clipping-plane.

N = Djupbuffertens upplösning.
```

I vårt fall, då vi använder oss av ett *near-clipping-plane* på 1 meter och en djupbuffert med en upplösning av 32 bitar, skulle värdena

Near-clipping-plane: zNear = 1

Djupbuffertens upplösning: N = 32

användas i formeln ovan.

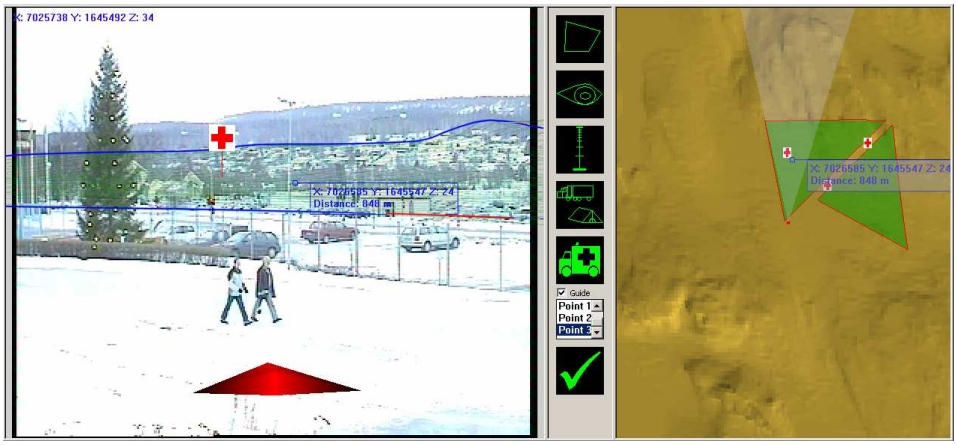
I tabellen nedan presenteras felprocenten för fyra olika distanser med de aktuella värdena på N och zNear.

Distans	500 m	1000 m	2000 m	3000 m
Felprocent	$1.16*10^{-7}$	$2.33*10^{-7}$	$4.66*10^{-7}$	$6.98*10^{-7}$

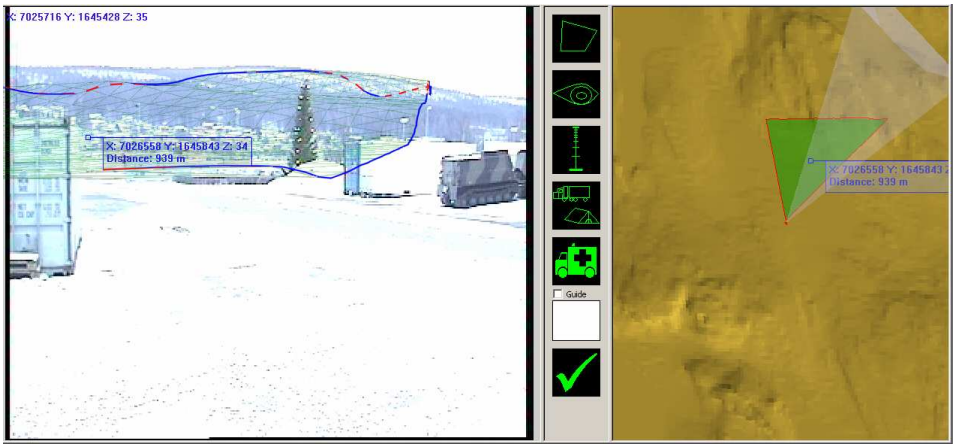
Dessa felvärden på djupbufferns precision är små nog för att med säkerhet kunna förkastas med tanke på den noggrannhet som andra möjliga felkällor i detta projekt har.

8.3 Navigeringsverktyg

Den pil som har implementerats för att peka ut riktningen mot valda objekts position fungerar väldigt bra. Det kan vara ett problem att se var pilen pekar, vilket beror på dess utseende och färgsättning.



Figur 8-3 pil som pekar mot en sjukvårdsplats.



Figur 8-4 markering av område i terräng.

8.4 Områdes- och punktmarkering

Markering av områden i videobilden fungerar inte vidare bra. Detta beror på samma orsaker som problemet med avståndsmätning. Däremot fungerar markeringen av områden i toppvyn bra, samma sak gäller även markering av intressanta punkter. Demonstratorn använder inte transparenta ytor för att visa områden då detta inte gav ett tillfredställande utseende, vilket förmodades vid projektets början. Ytorna täcker videobilden och gör det svårt att se viktiga detaljer i videobilden. Om man istället sänker genomskinligheten blir ytan svår urskilja. Istället ritas området ut med en kontur och som *wire-frame* inom området. Inte heller det blir riktigt bra, vilket beror på att det är svårt att se de linjer som är inom området framför videobilden. Ett problem med punktmarkeringen som skulle likna skyltar finns. De täcker för stort område av bilden och kan skymma andra mer viktiga detaljer i videobilden.

9 Diskussion

Det är svårt att dra slutsatser om hur ett system som detta skulle fungera i verkligheten. Den demonstrator som har implementerats är inte tänkt att användas för annat än att visa upp vad som är praktiskt användbart och vad som är möjligt att implementera. Systemet visar att det mycket väl är möjligt att implementera, men hur det sedan fungerar i praktiken, till exempel i en vagn som framrycker i hög hastighet, är en helt annan fråga som inte omfattades av detta examensarbete.

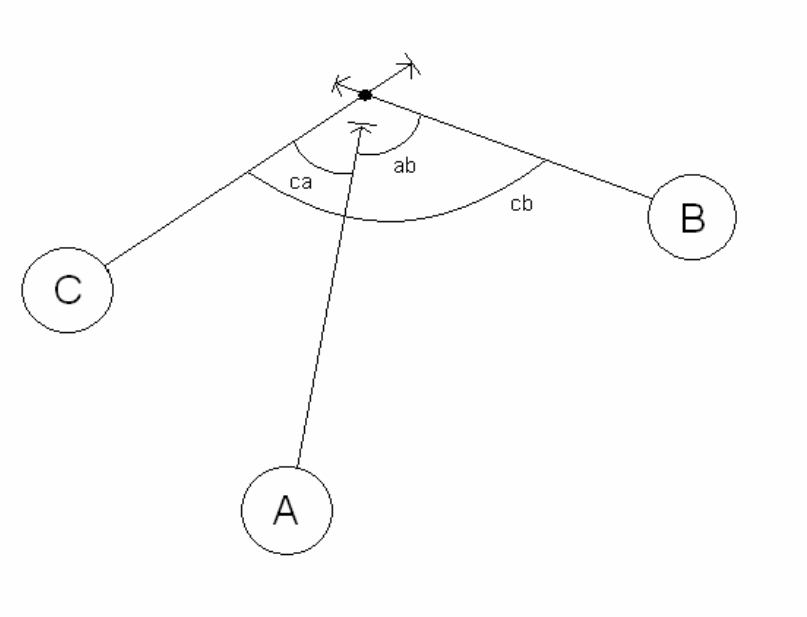
Att integrera en större mängd information som ska visas genom ett och samma media behöver inte heller betyda att det blir lättare för användaren att uppfatta och tolka informationen. Ett överflöd av information som presenteras på ett felaktigt sätt med möjligtvis svårhanterliga och svårförstådda funktioner, skulle med större sannolikhet kunna leda till en försämring av användarens prestationsförmåga.

9.1 Framtida arbete

9.1.1 Triangulering

För att göra avståndsberäkningen i systemet mer noggrann finns möjligheten att implementera någon form av triangulering av uppmätta avstånd mellan ett flertal olika fordon.

Ett enkelt förslag på en sådan algoritm skulle kunna vara att låta varje fordons individuella mätning viktas beroende på dess position samt vinkel mellan riktningsvektor relativt de andra fordonens. Värdet viktas sedan så att varje fordons värde, mätt i sidled, får störst relevans. Detta kommer att leda till att noggrannheten ökar eftersom felet som induceras i avståndet beror mest på hur inexakt höjden är och kommer därmed att viktas bort. Ett exempel med tre fordon illustreras i figur 9-1.



Figur 9-1 Trianguleringsalgoritm

I detta fall skulle B:s värde i sidled för punkten viktas högre än A:s värde för punkten räknat i A:s riktningsvektor. Detta för att B har större precision i sidled än vad A har i höjdled. Sedan kommer värdena från alla de olika fordonen att kombineras, utifrån mottsvarande vikt, på ett sätt som gör att det mest exakta värdet räknas högre.

9.1.2 Fragmentshaders

Programmet har inte stöd för fragmentshaders men det är tänkbart att någon gång i framtiden utöka programmet med stöd för det. Med fragmentshaders skulle det vara möjligt att göra avancerad och snabb bildbehandling, till exempel brusreducering, kantförbättring, histogramutjämningar. Tillsammans med vertexshaders skulle det även vara möjligt att göra grafiken mer trovärdig till exempel med en mer realistisk ljusberäkning, skuggor och reflektioner. Dock kan det diskuteras hur en användare skulle reagera på om det vore svårt att skilja grafiken från verkligheten.

9.1.3 Utökat navigeringsstöd

Det finns en stor mängd tänkbara idéer för att implementera navigationsstöd i en applikation som denna. Bland annat skulle exempelvis en kompass enkelt kunna implementeras som ständigt visar fordonets orientering eller riktning i väderstrecken.

En annan användbar funktion vore ifall användaren, efter att ha valt en slutdestination eller alternativt planerat och lagt upp sin färdväg i förväg, kunde få denna färdväg automatiskt presenterad i videobilden. Förslagsvis i form av en röd tråd som kan följas.

Något som vore intressant att titta närmare på i framtiden är möjligheten till att koppla samman flera olika instanser av Tactical Overlay System som körs på en mängd olika fordon, via någon typ av nätverk. Detta skulle ge upphov till nya möjligheter som att plutonchefen eller liknande kan delge sina gruppchefer olika strategiska beslut eller önskemål. Exempelvis kan detta röra sig om grafisk presentation av specifika fordons framryckningsvägar eller förtydliggöranden av andra viktiga saker.

9.1.4 IR-kamera

Från det föregående examensarbetet fanns ett förslag om att integrera information från IR-kamera. Tanken var att detta skulle göras i denna nya upplaga av demonstratorn men på grund av tidsbrist uteblev denna funktion. Men grundläggande rutiner för mottagning av informationen är implementerad och det enda som saknas är metoder för utritning.

9.1.5 Integration av ljud/taktil feedback

En annan attraktiv tanke vore om några av de implementerade, eller ovan nämnda, idéerna för visuellt stöd kunde kombineras med ljud eller möjligtvis någon form av taktil feedback för att på så vis förstärka och underlätta förmågan att ta in information.

10 Tack

Under detta arbete har ett flertal personer bidragit med sin kompetens och hjälp. Vi skulle vilja tacka dessa personer som är Mikael Knutsgård, Johan Ohlsson, Niclas Börlin för deras idéer och stöd. Utan denna hjälp skulle nog inte detta projekt kunnat genomföras. Tack!

Till sist skulle vi vilja tacka Britt-Inger Norberg och Ulrika Johansson för korrekturläsning av rapporten.

11 Referenser

1. Markus Isaksson och Jesper Wide, *Integration av data från yttre källor i videobild.*
   Luleå Tekniska Universitet, Systemteknik: Examensarbete 2007-02-07.

2. Donald Hearn och M. Pauline Baker, *Computer Graphics with OpenGL.*
   Pearson Prentice Hall
   Tredje upplagan, 2004.

3. Tomas Akenine-Möller och Eric Haines, *Real-Time Rendering.*
   A K Peters, Ltd
   Andra upplagan, 2002.

4. Rikets koordinatsystem 1990[Elektronisk] i Lantmäteriets hemsida.
   Tillgänglig: http://lantmateriet.se/templates/LMV_Page.aspx?id=4766
   [Läst 2008-01-09]

5. *Axis 250S MPEG-2 Video Server User's manual.*
   Revision 2.0, 2003.
   Axis Communications AB
   Emdalavägen 14
   SE-223 69 Lund

6. Greg Welch och Gary Bishop, *An Introduction to Kalman Filter.*
   University of North Carolina, Chapel Hill: SIGGRAPH 2001, Course 8.
   [Elektronisk]
   Tillgänglig:
   http://w3.impa.br/~pcezar/cursos/mpcg/papers/SIGGRAPH2001_CoursePack_08.pdf
   [Läst 2008-01-08]

7. Försvarets Materielverk, *DART 380 Beskrivning DART-format.*
   Utgåva 2, 1995.

8. Mikael Knutsgård, *Seriell datakommunikation med norrsökande gyro.*
   Mitthögskolan: Examensarbete 2001-06-08

9. OpenGIS*, Transducer Markup Language Implementation Specifikation*,
   [Tech. Rep. OGC® 06-010r6]
   Tillgänglig:
   http://portal.opengeospatial.org/modules/admin/license_agreement.ph

p?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/index.php?artifact_id=19371
[Läst 2008-02-05]
OpenGIS hemsida: http://www.opengeospatial.org/

10. Bayesian Network, [Elektronisk] Wikipedia.
Tillgänglig: http://en.wikipedia.org/wiki/Bayesian_network
[Läst 2008-01-08]

11. Kalman Filter, [Elektronisk] Wikpedia.
Tillgänglig: http://en.wikipedia.org/wiki/Kalman_filter
[Läst 2008-01-09]

12. *Sensor fusion*, [Elektronisk] Wikipedia
Tillgänglig: http://en.wikipedia.org/wiki/Sensor_fusion
[Läst 2008-01-04]

13. *Data fusion*, [Elektronisk] The Data fusion server.
Tillgänglig: http://www.data-fusion.org/article.php?sid=70
[Läst 2007-12-20]

14. David Eberly, *Triangulation by Ear Clipping.* [Elektronisk]
Tillgänglig:
http://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf
[Läst 2007-10-03]

15. *Radio 180/480 DART Ra 180 mobil ra 480 380 DART* [Elektronisk]
Tillgänglig: http://www.soldf.com/ra180.html
[Läst 2008-01-08]

16. B*aye's theorem,* [Elektronisk] Wikipedia.
Tillgänglig: http://en.wikipedia.org/wiki/Bayes%27_rule
[Läst 2008-01-10]

17. *Dempster-Shafer theory*, [Elektronisk] Wikipedia.
Tillgänglig: http://en.wikipedia.org/wiki/Dempster-Shafer_theory
[Läst 2008-01-10]

18. Don Koks och Subhash Challa, *An Introduction to Bayesian and Dempster-Shafer Data Fusion*, [Tech. Rep. DSTO–TR–1436]
Tillgänglig:
http://robotics.caltech.edu/~jerma/research_papers/BayesChapmanKolmogorov.pdf
DSTO Systems Sciences Laboratory
Edinburgh, SA 5111

Australia
[Läst 2008-01-10]

19. *Extensible Markup Language (XML) 1.0 (Fourth Edition)* [Elektronisk]
Tillgänglig: http://www.w3.org/TR/2006/REC-xml-20060816/
[Läst 2007-12-06]

20. *XML Binary Characterization Working Group Public Page* [Elektronisk]
Tillgänglig: http://www.w3.org/XML/Binary/
[Läst 2008-01-12]

21. *EBML - Technical Specifications* [Elektronisk]
http://ebml.sourceforge.net/specs/
[Läst 2007-12-04]

22. *Learning to Love your Z-buffer* [Elektronisk]
Tillgänglig:
http://www.sjbaker.org/steve/omniv/love_your_z_buffer.html
[Läst 2008-01-10]

23. Lawrence A. Klein, *Sensor and Data Fusion: A Tool for Information Assessment and Decision Making.*
SPIE, 2004