CHALMERS

Kvantifiering av förares rörelser vid hårda inbromsningar baserat på video från verkliga körningar

Quantify drivers' motions at hard braking based on videos from real world driving

Examensarbete inom högskoleingenjörsprogrammet Elektroingenjör

ANDREAS BRING
KIM ROSBERG

Institutionen för signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige, 2013

EXAMINATOR:
Bertil Thomas

HANDLEDARE VOLVO:
Stina Carlsson

HANDLEDARE CHALMERS:
Manne Stenberg

PLATS:
Göteborg

DATUM:
Juni 2013

Kim Rosberg Andreas Bring: *Kvantifiering av förares rörelser vid hårda inbromsningar baserat på video från verkliga körningar,* Quantify drivers' motions at hard braking based on videos from real world driving

FÖRORD

Denna rapport är en beskrivning av vårt examensarbete, det avslutande momentet på elektroingenjörsprogrammet. Utbildningen omfattar 180 hp där examensarbetet utgör 15 hp.
Arbetet har utförts hos Volvo Personvagnar som ska ha ett stort tack för att de gjorde examensarbetet möjligt. Vi vill särskilt tacka Stina Carlsson som har varit en utmärkt handledare under vår tid hos Volvo. Vi vill även passa på att tacka Manne Stenberg som varit vår handledare på Chalmers, men även Håkan Gustavsson på Volvo för hjälpen vi fick med bilen. Vår ambition med arbetet har varit att bredda våra kunskaper genom att fördjupa oss i nya områden. Detta har vi till stor del lyckats med.

Andreas Bring
Kim Rosberg
Göteborg, Juni 2013

ABSTRACT

This thesis was carried out at Volvo Car Corporation. The department of active safety wanted to develop a model for the conversion of pixel positions in video, from a car's interior to a 3D position in the car. This model would then be used to quantify drivers positions in cars under hard braking.

This report describes a model for calculating the three-dimensional world coordinates from two-dimensional images, by creating scenarios of simulated brakings. Also described is how to analyze images and assess the possible distortion, correction of distortion, selection and measurement of reference points, the development of an algorithm and statistical analysis of the results.

The work resulted in an algoritm that could estimate the position of a driver from the scenario with approximately 14% maximum error, except for some deviations.

The conclusion of this thesis is that quantifying a driver's position in a car from a single camera is, while possible, unequivocally going to result in significant errors because of the small magnitude of motion when breaking.

SAMMANFATTNING

Detta examensarbete utfördes vid Volvo Personvagnar. Avdelningen för innovation, elektronisk säkerhet, aktiv säkerhet och chassi ville utveckla en metod för omvandling av pixelpositioner i filmer, från en bils interiör till en 3D-position i bilen. Denna metod skulle då kunna användas för att kvantifiera förares positioner i bilar under hård inbromsning.
Denna rapport beskriver en modell för beräkning av tredimensionella världskoordinater och rörelser i en bil från tvådimensionella bilder, genom att skapa scenarier av simulerade inbromsningar. Den beskriver dessutom hur man analyserar bilder och bedömer distorsion, korrigering av distorsion, urval och mätning av referenspunkter, utveckling av en algoritm och statistisk analys av resultat.
Arbetet resulterade i en algoritm som kan beräkna positionen av en förare från scenarier, med ca 14 % maximalt fel, bortsett från några avvikelser.
Slutsatsen av detta examensarbete är; uppskattning av förarposition i en bil från en kamera kommer att, även om det går att uppskatta, leda till omfattande fel p.g.a. den ringa magnituden av rörelse vid inbromsning.

INNEHÅLLSFÖRTECKNING

FIGURER

TABELLER

AKRONYMER

FOT Field Operation Test

CMM Coordinate Measuring Machine

ISO International Organization for Standardization

1 INLEDNING

1.1 BAKGRUND

Bilar är något som, på modern tid, revolutionerat människans sätt att färdas både långa och korta sträckor. Till en början var säkerhet inte något som lades mycket tid på, eftersom bilar var dyra och något bara de mest välställda hade råd med. Allteftersom bl.a. Henry Ford gjorde bilar tillgängliga till gemene man har antalet bilar på vägarna ökat, vilket oundvikligen resulterat i fler olyckor. Idag uppskattas att cirka 1.2 miljoner dör varje år i bilolyckor runt om i världen [Breen, 2004]. Detta faktum gör att biltillverkare satsar alltmer på bilens säkerhet.

Passiv säkerhet är ett område som biltillverkare lagt mycket tyngd på under tiotals år. Detta har resulterat i uppfinningar som trepunktsbälte och airbag. Det främsta syftet med passiv säkerhet är undvika, eller minimera skadorna när olyckan redan har skett.
Aktiv säkerhet till skillnad från passiv säkerhet, lägger fokus på att förhindra olyckor. Aktiv säkerhet är däremot relativt nytt, men något som ökar stort bland biltillverkare. System som varnar då risken för krock är mycket stor så att föraren kan bromsa, eller som autonomt bromsar bilen för att undvika krocken, är två exempel på aktiva säkerhetssystem. Förhållandet mellan aktiv- och passiv säkerhet har börjat studeras, men är relativt odokumenterat. Volvo vill därför utföra en undersökning där man studerar hur föraren rör sig vid både autonoma och manuella inbromsningar.

1.2 SYFTE

Syftet med detta projekt är att undersöka om det går att kvantifiera hur mycket kroppen rör sig under manuella och autonoma inbromsningar utifrån bilder från en bils interiör. Detta möjliggörs av Volvos Field Operation Test (FOT), där man samlade in data och film på över 100 utvalda förare i Volvos bilar under ett år. Filmerna på förare används sedan för att utveckla en metod och algoritm för att transformera bilders pixelpositioner till världsliga 3D-koordinater.

För att verifiera att metoden fungerar och ge en grund för vidareutveckling ska flera scenarier skapas som simulerar troliga förarsituationer vid inbromsningar. Bilderna från kamerorna som används i

FOT-projektet är distorderade, vilket medför att bilderna måste korrigeras innan de går att nyttja med algoritmen.

1.3 AVGRÄNSNINGAR

Från FOT-projektet finns det tillgång till flera kameror i bilen, varav två är riktade mot förarsätet. Den ena kameran är positionerad vid backspegeln och den andra bakom ratten, vilket medför att den förstnämnda kameran har ett större synfält. Dessa kameror är inte synkade i tid, har som sagt inte samma synfält och är inte vinkelräta mot varandra vilket utesluter stereoskopiskt seende[1]. Därför kommer endast kameran vid backspegeln användas. En allmän algoritm för extrahering av koordinater i all FOT-data kommer ej att utvecklas. Ett scenario kommer istället utvärderas för att ge en grund för hur lösningar av liknande problem går till. Vissa variabler i bilen, såsom förarsätesposition och rattposition kommer att standardiseras.

1.4 PRECISERING AV ARBETSUPPGIFTEN

- Val och uppmätning av referenspunkter för utveckling av algoritm.
- Val av scenarier för verifiering av algoritm.
- Analys av bilder från FOT-datan och bedömning av distorsion.
- Korrigering av distorsion i bilder.
- Utveckling av algoritm.
- Statistisk utredning av resultat.

1 Stereoskopiskt seende: Med hjälp av två kalibrerade kameror som observerar samma punkt kan 3D-koordinaterna av punkten uträknas [Sonka et al., 2008].

2 TEKNISK BAKGRUND

Nedan finns information om det tekniska sammanhanget för rapporten bl.a. vilken bil som använts, mjukvara, hårdvara samt teori.

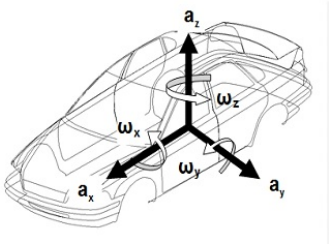
2.1 BIL

Bilen som användes till arbetet var en tredje generationens Volvo *XC70*. Det enda kriteriet för val av bil var att den skulle varit del av FOT-projektet. Bilen är en fyrhjulsdriven kombi som är 482 cm lång och 186 cm bred.

Figur 1: XC70

2.1.1 *Koordinatsystem i bil*

Uppmätningar i bilen görs utifrån ett fördefinierat världskoordinatsystem. Standarden för koordinatsystem i bilar är ett kartesiskt koordinatsystem i tre dimensioner. Positiva x utbreder sig mot motorhuven, positiva y mot förardörren och positiva z mot biltaket. I Volvos uppmätningar utgår origo framför motorhuven. Origo kan emellertid ändras vid behov till en definierad referenspunkt.



Figur 2: Kartesiskt världskoordinatsystem i bil

2.2 HÅRDVARA

2.2.1 *Kamera*

Monokroma kameran *KPC-S500* var installerad i bilen som användes till arbetet. En monokromatisk kamera har ett litet frekvensomfång för ljus, vilket gör att bilderna blir nästintill svart-vita. Kameran är 25 mm lång, 25 mm bred, 22 mm hög och väger endast 79 gram . Kamerans lins har en brännvidd på 2.45, vilket medför ett stort synfält på 150 grader. Bländarstorleken är F2.0, vilket är relativt litet, men beror framför allt på kamerans ringa storlek. Detta medför att bilder tagna under dålig ljussättning blir mörka, vilket medför att identifiering av objekt blir mycket svårare. Kameran tar 12.5 bilder per sekund, vilket är ungefär hälften av normal inspelningshastighet, och har valts för att filmerna ska ta mindre utrymme på hårddiskar. Upplösningen för bilderna är 352x288 pixlar med 24 bitars bitdjup.

Figur 3: KPC-S500, [KT&C, 2013].

2.2.2 *FaroArm*

Under uppmätningen av bilen användes en Coordinate Measuring Machine (CMM), som kallas FARO Edge. Den möjliggör mätningar i en arbetsvolym från 1.8-3.7 meter, med precision på mellan 0.024-0.064 millimeter.
Dessa mätningar utförs med ett eget definierat origo som används som referenspunkt för alla mätningar.

Figur 4: FARO Edge ©, [FARO Technologies, 2013].

2.3 MJUKVARA

2.3.1 *Matlab*

Algoritmutveckling och bildbehandling skedde till stor del i studentversionen av Matlab2013a. Detta program är särskilt utvecklat som hjälpmedel för matematiska beräkningar och ingenjörsmässigt arbete, men har också flera verktygslådor för t.ex. bildbehandling och reglertekniska simuleringar.

I studentversionen av Matlab ingick flera verktygslådor som var av intresse, men framför allt "*Image Processing Toolbox*", som användes i bildbehandlingsdelen av projektet. Kamerakalibreringen gjordes i "*Camera Calibration Toolbox*" som är särskilt utvecklat för att vara ett enkelt och smart sätt att kalibrera kameror i Matlab [Bouguet, 2010]. De statistiska beräkningarna gjordes i "*Statistics Toolbox*".

2.3.2 *FARO CAM2*

Punkterna som mäts upp av *FARO Edge*, evalueras och sparas av denna programvara. Datan kan sedan exporteras till ett dokument och användas i Matlab.

2.4 TEORI

2.4.1 *Distorsion*

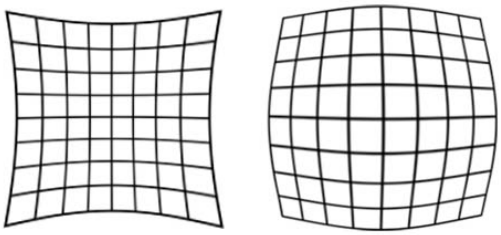
Brännvidd, skärpa, slutartid och bländare är några av inställningar på kameror som kan ändras vid behov. Dessa valmöjligheter illustrerar att kamerabilder har många variabler som måste stämma överens för bästa möjliga resultat. Oberoende av fotograf finns det emellertid alltid distorsion och/eller brus i bilden.

Det finns flera olika typer av distorsion, men den vanligaste är radiell distorsion. Denna distorsion är radiellt symmetrisk och ökande eller minskande med avståndet från mitten av bilden. Den kan delas in i flera olika typer av radiella distorsioner men de allra vanligaste är: nåldynsdistorsion och tunndistorsion (på eng: pincushion- och barrel distortion). Distorsionen är mest tydlig när bilden innehåller raka linjer, eftersom de ser böjda ut [Romero och Gomez, 2007].

Tunndistorsion är en effekt som ofta förekommer vid användning av vidvinkelslinser. Detta gör att bilden ser böjd ut, som om bilden lagts på ett runt objekt, t.ex. sidan av en tunna. Man brukar säga att bilden är sfäriserad. Effekten är mest tydlig i mitten av bilden, nära den optiska axeln. Denna effekt används som en tillgång i fisheye-linser[1], då

1 Fisheye-lins: Vidvinkelslins med ett synfält på upp till 180°

syftet med dessa linser är att få ett större synfält. Nåldynsdistorsion



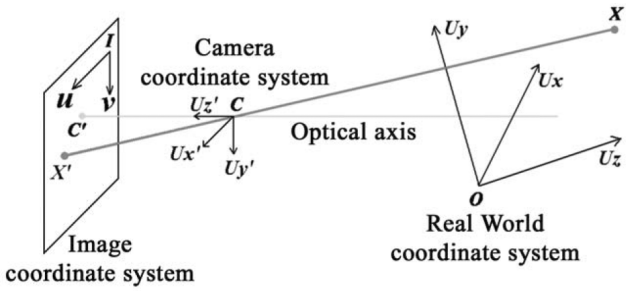
Figur 5: Nåldynsdistorsion t.v och Tunndistorsion t.h, [Starizona, 2013].

har motsatt effekt av tunndistorsion och inträffar oftast vid användning av telefotolinser[2]. Effekten gör att bilden ser ut att vara böjd inåt, mot mitten av bilden, som en nåldyna. Denna effekt är mest tydlig i kanterna av bilden, långt från den optiska axeln.

2.4.2 *Kamerans parametrar*

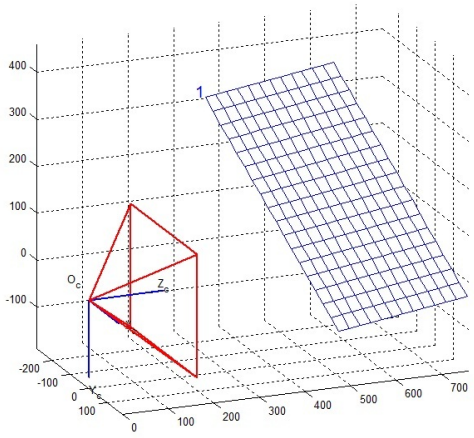
Kamerans inre- och yttre kalibreringsparametrar (på eng: intrinsics och extrinsics) kan räknas ut genom att relatera bildens koordinater till världens koordinater. De yttre kalibreringsparametrarna anger hur kameran är positionerad i rummet. De inre är däremot, som namnet anger, kamerans inre specifikationer såsom brännvidd, bildformat, bildhuvudpunkt, skalfaktor och en faktor som beskriver hur snedvriden bilden är. Dessa brukar samlas i en s.k. kalibreringsmatris.
Hålkameran, eller nålhålskameran, är en av de äldsta uppfinningarna som kunde projicera verkligheten som en bild i hålkameralådan. Hålkameramodellen definierar en geometrisk relation mellan en 3D-punkt och en den projicerade bildpunkten. Med hjälp av denna modell kan man göra en 3D-koordinattransformation för att ange transformationen från världs- till kamerakoordinater, det vill säga kamerans yttre parametrar [Szeliski, 2010]. Den matematiska relationen mellan bildkoordinater och världskoordinater definieras nedan med hjälp av denna modell, se fig 6. för visualisering.
Världskoordinater definieras som U_v, kamerakoordinater U'_k och bildkoordinater b_b. Kamerakoordinaterna är beskrivna på så sätt att origo ligger i mitten av projektionen från kameran (C') och z-axeln är längs med optiska axeln. Bildkoordinaterna b_u,b_v är skrivna som pixelkoordinater där $b_w = 1$ eftersom bilder har två dimensioner. Origo i bilden utgår från högst upp i höger hörn (högst upp i vänster brukar dock vara den vanligaste konventionen). Slutligen kallas avståndet mellan bildplanet och punkten C brännvidden f.
Dessa samband, variabler och definitioner gör det möjligt att hitta

2 Telefotolins: Lins som används vid fotografering av objekt på långa avstånd.



Figur 6: Hålkameramodell med kamera-, bild- och världskoordinatsystem, [Guerlain och Durand, 2006].

sambandet mellan bildkoordinater och världskoordinater och att sammanställa kalibreringsmatrisen.



Figur 7: Kamerans yttre parametrar utifrån kamerans perspektiv med hjälp av ett schackbräde [Bouguet, 2010].

2.4.3 *Rektifiering*

När kameramodellen är klar, d.v.s. kamerans parametrar är uträknade, kan bilder rektifieras. Detta betyder att man kompenserar för distorsioner i bilder och på så sätt gör objekt som är synliga i bilden mer proportionella mot verkligheten. Man transformerar det distorderade koordinatsystemet till ett normaliserat koordinatsystem.

2.4.4 *Brus*

Bilderna som är tagna av kameran i bilen är brusiga. Detta kan uppkomma av många orsaker, såsom ISO-värde[3], slutartid och bildbehandling, m.fl. I bilderna från FOT-projektet har bruset mest sannolikt tillkommit av lång slutartid och högt ISO-värde eftersom föraren ska

3 ISO-värde: Ett mått på fotografisk films känslighet för exponering av ljus.

vara synlig även när det är mörkt i bilen.

Rektifiering medför också att förekomsten av brus ökar eftersom bilden "plattas till" och pixlar "dras ut", medan bildens dimensioner ej förändras, för att ge en mer verklig bild. Detta skapar ett problem när information ska extraheras ur bilden, eftersom vissa detaljer kan försvinna ur bild. Matlab har emellertid en verktygslåda för bildbehandling, *Image Processing Toolbox*, där det finns flera funktioner för brusborttagning och filtrering av bilder.

3 METOD

Följande delkapitel beskriver tillvägagångssättet för att lösa vårt problem. De innefattar skapandet och utförandet av våra scenarier, korrigering av bilder och utveckling av algoritm.

3.1 UPPMÄTNING FÖR ALGORITMUTVECKLING

Det är inte möjligt att räkna ut avstånd till ett objekt eller storleken på ett objekt i en bild, utan att utföra mätningar. Därför visste vi tidigt att vi var tvungna att utföra någon typ av uppmätning av både bilen och föraren. I bilder finns sambandet att objekt som flyttar sig närmare kameran blir större. Detta samband tänkte vi använda, men för att veta hur mycket objektet flyttat sig, måste man veta storleken för att kunna ta fram ett samband mellan bild- och världskoordinater. Det vill säga om exempelvis ett objekt blir fem pixlar större vid förflyttning, så är naturligtvis objektet lika stort i verkligheten, men befinner sig närmare kameran. Då kan man hitta ett samband mellan position av objektet och storlek i pixlar.

I vårt fall hade vi inget väldefinierat objekt i bilderna, bara förare. Det vi strävade efter var att ta bilder på något med kända dimensioner i rörelse i en bil och samtidigt mäta avståndet till kameran. Volvo hade en laborationslokal med FARO Edge som kunde mäta upp punkter i bilen. Detta medförde att vi kunde få positionerna av både stationära och rörliga objekt i bilen i förhållande till ett världskoordinatsystem. Bilen vi fick till vårt förfogande användes dessutom i FOT-projektet. Detta medförde att vi filmade alla uppmätningar med en kamera vars uträknade parametrar kunde användas i framtiden till FOT-projektets bilder. Detta ligger sedan till grund för att utveckla en algoritm för att korrelera bildkoordinaterna med världskoordinater.

3.1.1 *Virtuellt rum*

Ett schackbräde har väldefinierade dimensioner, på grund av att alla rutor är lika stora och symmetriska. Eftersom schackbräden är svartvita ger även det skarpa kontraster. Således kan ett nät av mätpunkter skapas utifrån schackbrädets hörn då antalet schackrutor och storleken per schackruta är känd. Vi tejpade därför fast schackbrädet på förarsätet och flyttade schackbrädet successivt framåt, samtidigt som vi mätte upp schackbrädets näst mest yttersta hörn. Detta skapade ett virtuellt rum med kända världskoordinater där föraren kunde rö-

ra sig, som vi sedan kunde utöka med hjälp av matematisk regression (se fig 8).

Figur 8: Uppmätning av schackbräden i bilen, med *FARO Edge*.

3.1.2 *Referenspunkter*

Vi mätte även upp några stationära punkter, såsom ratten och B-stolpen (se bilaga A). Dessa punkter tänkte vi använda till att lösa orienteringen av kameran, om behovet fanns. Dessa punkter kan även senare användas till förflyttning av origo i världskoordinatsystemet.

3.1.3 *Scenario*

Under uppmätningen utfördes dessutom åtta scenarier, då vi fick agera som förare under simulerade inbromsningar medan vi mätte upp vår position med hjälp av FARO Edge. Bildbehandlingsavdelningen på Volvo, som hade ansvar för bl.a. filmerna från FOT-projektet, gav oss bilder från de mest extrema autonoma och manuella inbromsningarna som filmades i projektet. Dessa bilder visade emellertid relativt små rörelser. Efter konsultation av bildbehandlingsavdelningen utformades fyra scenarier, på två olika individer, för efterlikna dessa inbromsningar från filmerna.

Resten av scenarierna simulerade mycket stora rörelser från inbromsningar, som emellertid inte fanns i filmerna från FOT-projektet, men fortfarande var realistiska. I varje scenario mättes två punkter, en på ansiktet och en på bröstkorgen (se fig 9).

Figur 9: Uppmätning av punkter från ett scenario, med *FARO Edge*.

3.2 BILDKORRIGERING

Bilder som i vårt fall ska användas till uträkning av avstånd till, eller storlekar av, objekt måste korrigeras. Detta beror på att om objekt är oproportionella mot dess verkliga storlek i bilden, blir givetvis beräkningar felaktiga. Det första steget mot en korrigerad bild blir därför att ta bort eller minska distorsionen.

I vårt fall kunde vi ta reda på vår ungefärliga distorsion genom att granska bilder från FOT-projektet, eftersom distorsionen var väldigt tydlig. Kamerorna som var uppsatta i bilen hade en fisheye-lins, för att kunna fånga så mycket av interiören av bilen som möjligt. Detta är ett bra exempel på hur man kan använda sig av kameror med vidvinkelslinser, för att få större synfält än med ”vanliga” linser. Denna distorsion gjorde emellertid att proportionerna i bilden var fel, vilket resulterade i att vi var tvungna att korrigera bilderna. För att kunna korrigera en bild måste de inre och yttre parametrarna i kameran vara kända, vilket betyder att kameran måste kalibreras.

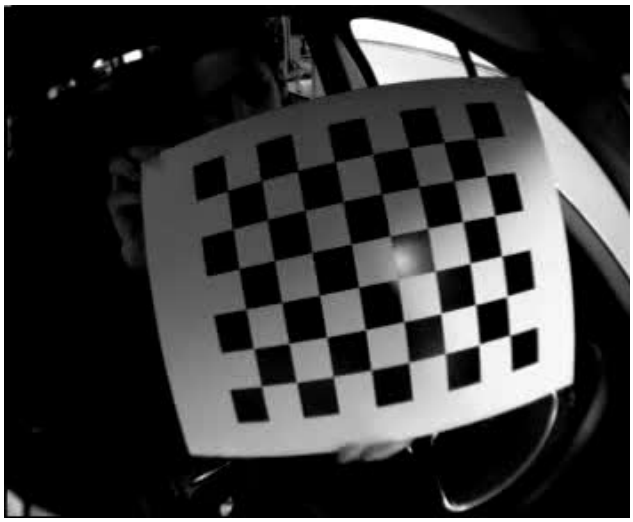
3.2.1 *Kamerakalibrering*

Det vanligaste sättet att kalibrera en kamera är att använda sig av kända dimensioner på föremål i bilden. Dessa kända dimensioner används för att kvantifiera hur distorderad bilden är och bestämmer kameramatrisen därefter. Detta kan göras på flera olika sätt, men i vårt fall valde vi att använda oss av schackbräden, där dimensionerna av varje schackruta är känd. Vi använde oss av kamerakalibreringsmetoden som finns beskriven av [Zhang, 1999] och [Heikkilä och Silvén,

1997]. Denna metod har vidareutvecklats till en verktygslåda i Matlab av [Bouguet, 2010]. En användarguide finns även upplagd på hans hemsida (se referenser).

Kalibreringstekniken denna verktygslåda använder sig av förutsätter att man har tagit flera bilder på schackbräden med kameran man vill kalibrera. Det viktigaste är att schackbrädets mönster är synligt i alla bilder och inte täcker hela bilden. Verktygslådan är dock lätthanterlig, så uteslutande av eventuella felaktiga bilder är lättgjort. För bästa resultat ska man använda sig av minst tio bilder med schackbrädet på olika positioner i förhållande till kameran, för att fånga flera vinklar.

Efter att ha klickat ut hörnen på schackbrädet i bilden i verktygs-



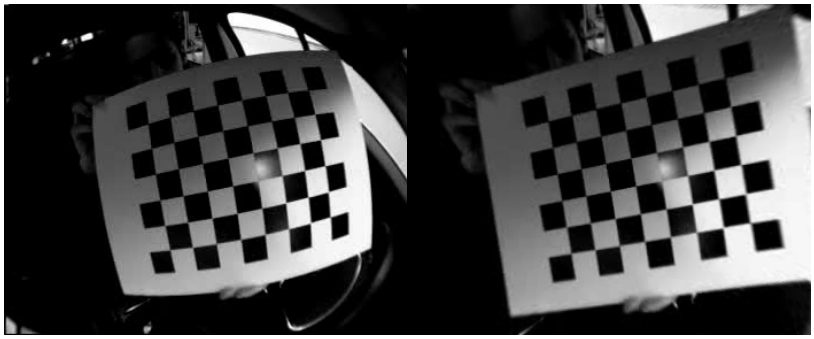
Figur 10: Ett schackbräde som användes vid kamerakalibrering. *Tunndistorsionen* är tydlig.

lådan, markeras varje schackruta ut automatiskt av programmet med hjälp av de kända dimensionerna av schackrutorna. Därefter kan man korrigera placeringen av markeringarna, om de hamnat fel på grund av radiell distorsion, med hjälp av en distorsions-koefficient. När denna procedur upprepats på alla bilder, kan man kalibrera bilderna. Detta medför att man kan se pixelfelet på bilderna, visualisera kamerans yttre parametrar och rektifiera bilderna.

3.2.2 *Rektifiering av bilder*

Verktygslådan vi använder har ett verktyg där man kan rektifiera vilken bild som helst med de uträknade parametrarna för kameran. Detta gör att det räcker att kalibrera kameran för en uppsättning av bilder och sedan använda denna kalibrering för andra bilder tagna med likadana kameror. Kvalitén på rektifieringen beror emellertid mycket

på kalibreringsbilderna och hur kalibreringsobjektet var positionerat i förhållande till kameran. För nära kameran, för långt bort eller utanför kameras synfält kan göra att täckningen för rektifieringen blir sämre på vissa ställen.



(a) **(b)**

Figur 11: Före- och efter rektifiering. Här är det mycket tydligt att schackbrädesrutorna är mer proportionella i bild (b) än i bild (a).

3.2.3 *Filter*

Kameran som används i FOT-projektet är som tidigare nämnt (se kap 2) monokrom och bilderna den tar har en relativt låg upplösning. Detta ger upphov till brus, vilket kan dölja detaljer som är vitala för algoritmen. Därför används Matlab-funktionen *histeq* för att förbättra ljus och kontrast i bilden.
Histeq förstärker kontrasten av bilder genom att omvandla värdena i intensitetsbilden. Detta görs med hjälp av förbestämda parametrar som ställs in för funktionen.
Histogrammet visar tonfördelningen i bilden med hjälp av ett grafiskt diagram, där x-axeln visar hur många pixlar som finns representerade vid en viss nivå. Om antalet pixlar är stort vid origo är bilden underexponerad, som i vårt fall.

3.3 ALGORITMUTVECKLING

Ett bra exempel för principen av vår algoritm är att vi mäter upp och filmar var en linjal befinner sig i bilen i tredimensionella koordinater, där vi tror att föraren rör sig. Därefter observeras hur stor linjalen är i bilden på olika positioner i bilen, samtidigt som avståndet till linjalen uppmäts. Antag sedan att linjalen ersätts med en förare, där vi tagit ut en proportion på föraren som är lika stor som linjalen. Om vi då vet att linjalen och förarens proportion har samma storlek i verkligheten, kan vi veta var föraren befinner sig i bilen om vi jämför storleken av linjalen och föraren tills de är lika stora i bilden.
Detta betyder i vårt fall att utifrån uppmätta punkter skapades ett tredimensionellt rum med världskoordinater. Från film på samma uppmätning skapades dessutom 30 skivor med tvådimensionella bildpunkter. Världsliga tredimensionella punkter och tvådimensionella bildpunkter blev givna samma index för samma punkt. Genom identifiering av ett objekts pixelsposition i bilden och information om längden av objektet i bilden och verkligheten, korrelerades de tvådimensionella bildpunkterna från schackbrädet med objektet.
Sambandet byggde på att längden som objektet hade i verkligheten i schackrutor, skulle ha samma längd i schackrutor i bilden. Därefter användes index från bildpunkter till att få fram världskoordinater.

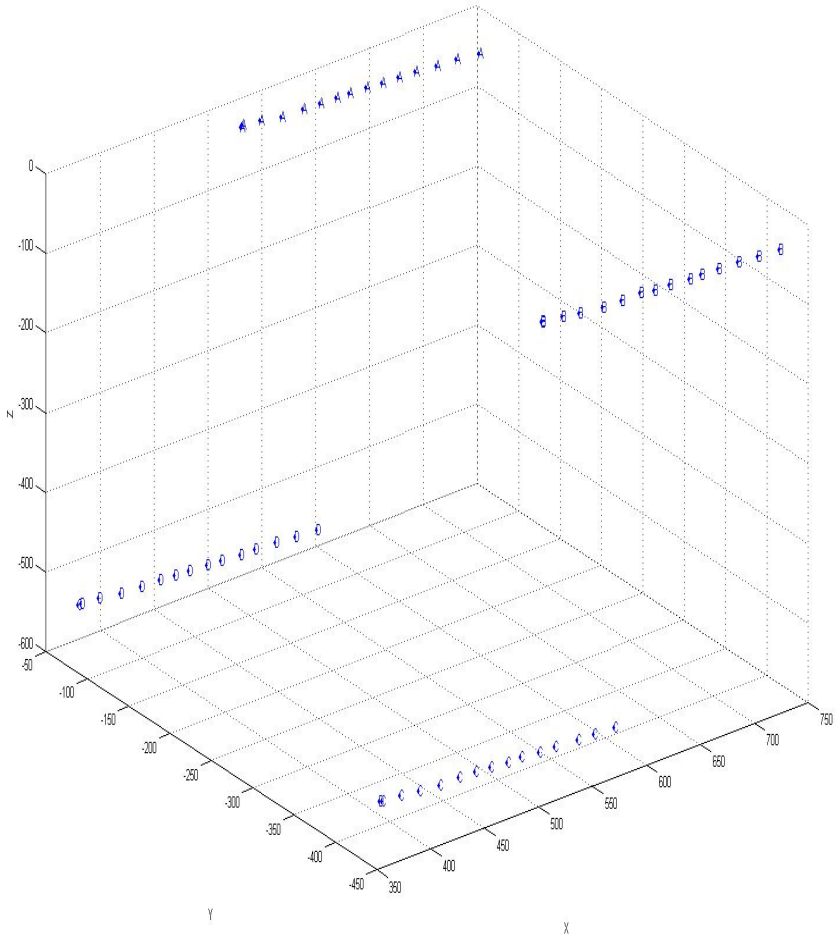
3.3.1 *Beräkning av sidlängd av schackruta*

De uppmätta tredimensionella världskoordinaterna från uppmätningen användes dessutom för att räkna ut hur lång en sida är på en schackruta. Detta för att få större noggrannhet, än manuell upppmätning med linjal. För beräkningarna användes Herons formel och vektoralgebra. Punkterna lästes in med start näst högst upp i vänster hörn på schackbrädet, sedan högst upp i höger hörn och fortsatte på samma sätt nedåt medurs och gavs namnen A,B,C och D, se fig.12. För att beräkna längden gjordes beräkningar med en förlängning av Herons triangel till en rektangel, där arean gavs av formeln:

$$A = \sqrt{(s-a)\cdot(s-b)\cdot(s-c)\cdot(s-d)} \tag{1}$$

I linjär algebra är området i ett plant paralellogram kryssprodukten av två intilliggande kantvektorer, där arean ges av formeln:

$$A = |(V_1 - V_0) \times (V_3 - V_0)| \tag{2}$$

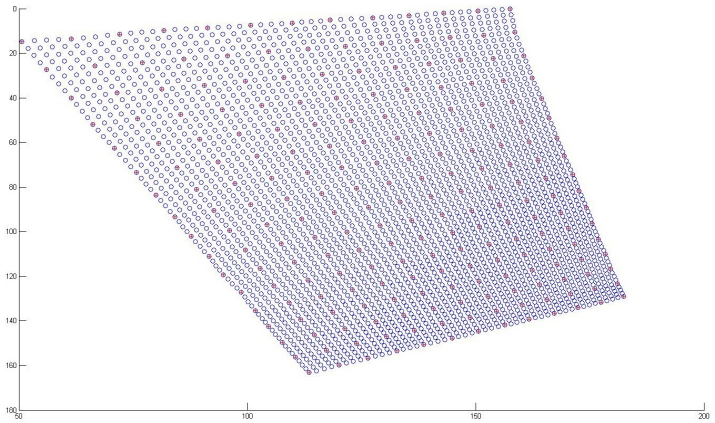


Figur 12: Fyra näst yttersta punkterna på schackbrädena i 15 olika positioner, i världskoordinatsystem i Matlab.

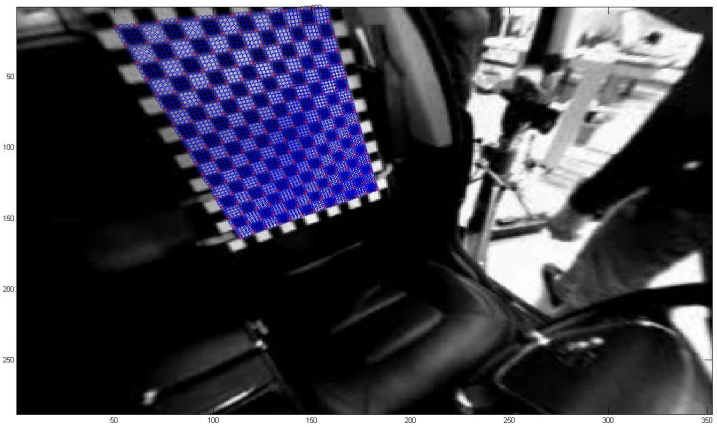
3.3.2 *Tvådimensionella bildpunkter*

Från de rektifierade bilderna av uppmätningen skapades en databas av bildpunkter som fick samma index som de tredimensionella uppmätta punkterna på schackbrädena. Databasen skapades genom att identifiera den ungefärliga pixelpositionen av hörnen på schackrutorna i bilderna, vilket gjordes med verktygslådan *Camera Calibration toolbox* i Matlab. I verktygslådan skapas en textfil för varje bild och eftersom vi i vårt fall hade 15 schackbrädesbilder blev det 15 textfiler. Algoritmen hämtar sedan x- och y koordinater från textfilerna och lägger in koordinaterna i en temporär matris efter vilken skiva, rad och kolumn punkten tillhörde. För att skapa ett större nät av bildpunkter delas sidlängden av schackrutorna in i fyra längder. Dessa värden blir nya punkter i matrisen. Samma princip gäller för att ska-

pa punkterna i vertikalled, med skillnaden att det är fler rader än kolumer, vilket medför fler iterationerna. Denna process fortsätter sedan för varje schackruta (se fig.13 & 14).



Figur 13: Utökning av punkter, där röda punkter är hörnen på schackrutor och blå är nya utökade punkter

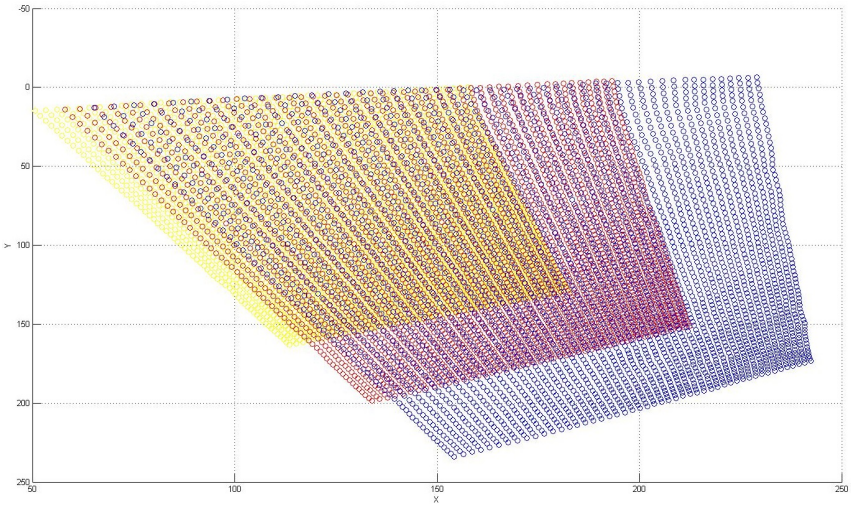


Figur 14: Tillämpning av dessa utökade punkter på en bild från uppmätning av schackbräde. Notera även att det tydligt syns att de röda punkterna träffar hörnen på schackrutorna och de blåa hamnar mellan.

Eftersom schackbrädet tejpades fast på stolen i bilen, kan schackbrädena bara skapa ett så stort rum som stolen kan flyttas. Detta medförde att 15 positioner kunde mätas upp, där stolen flyttades från längst bak, till längst fram. Eftersom rummet blev relativt litet skapades 15 nya skivor framåt med hjälp av linjär regression [Groß, 2003].

Skillnaden i x-koordinater för de tvådimensionella bildpunkterna mellan skivorna räknades ut och adderades ihop för att divideras med 15 skivor, vilket gav ett aritmetiskt medelvärde för avståndet mellan

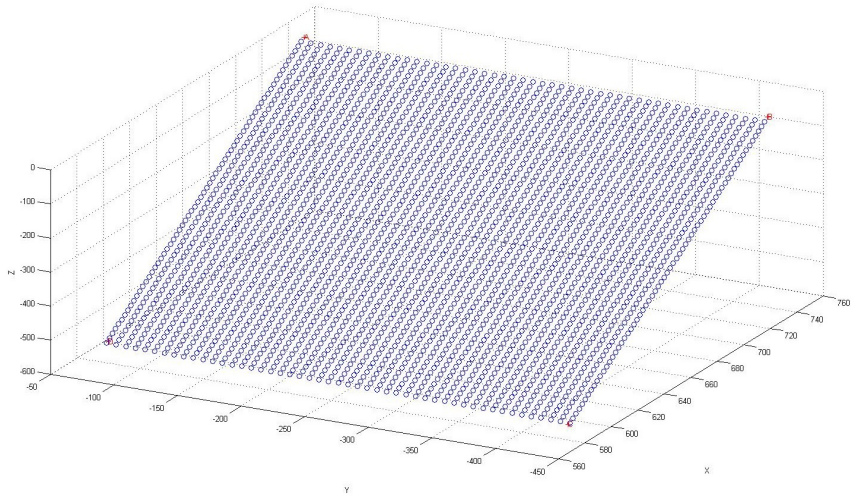
skivorna. Medelvärdet adderades sedan från sista skivan tills dess att 15 nya skivor skapats. Genom linjär regression hade linjens ekvation skapats och användes därefter för att räkna ut y-värdena. Detta gjordes genom att sätta in de nya x-värdena i funktionen och på så sätt skapa de 15 nya skivorna.
Totalt skapades 95550 punkter belägna på 30 skivor med 65 rader och 49 kolumner (se fig.15), vilka sparas i en databas. Dessa punkter ska bilda ett samband med de motsvarande tredimensionella världspunkterna som beskrivs i nästa delkapitel.



Figur 15: Skiva 1 i gult, 15 i rött och 30 i blått. Här syns sambandet att skivornas areor i pixlar, likt schackbrädena, blir större ju närmare kameran de kommer.

3.3.3 *Tredimensionella världspunkter*

Algoritmen läser in samtliga punkter från uppmätningen m.h.a. schackbrädet i förarstolen. Sedan indexeras mätpunkterna efter skivor och kallar hörnen på varje skiva för A, B, C och D, på samma sätt som för bildpunkterna. Avståndet mellan A och D i x, y och z räknas ut och divideras med 65 (antalet rader). Analogt beräknas avståndet B till C, där båda avståndsskillnaderna adderas och sedan divideras med två, d.v.s. aritmetiskt medelvärde. Algoritmen skapar sedan x-, y- och z-koordinater för punkter på 65 rader, 49 kolumner och 15 skivor. Dessa koordinater läggs sedan in från en rad nedanför B- och A punkten, på grund av att översta raden på schackbrädet inte blev synlig i de rektifierade bilderna då schackbrädet var långt fram. Därefter adderas A till B, dividerat med 49 (antal kolumner) och i samma iterering subtraheras medeldistansen för horisontalled. Detta görs för att x går i positiv riktning och y i negativ. Itereringen startar med värdet noll för rader, vilket görs för att första raden med kolumner inte skall



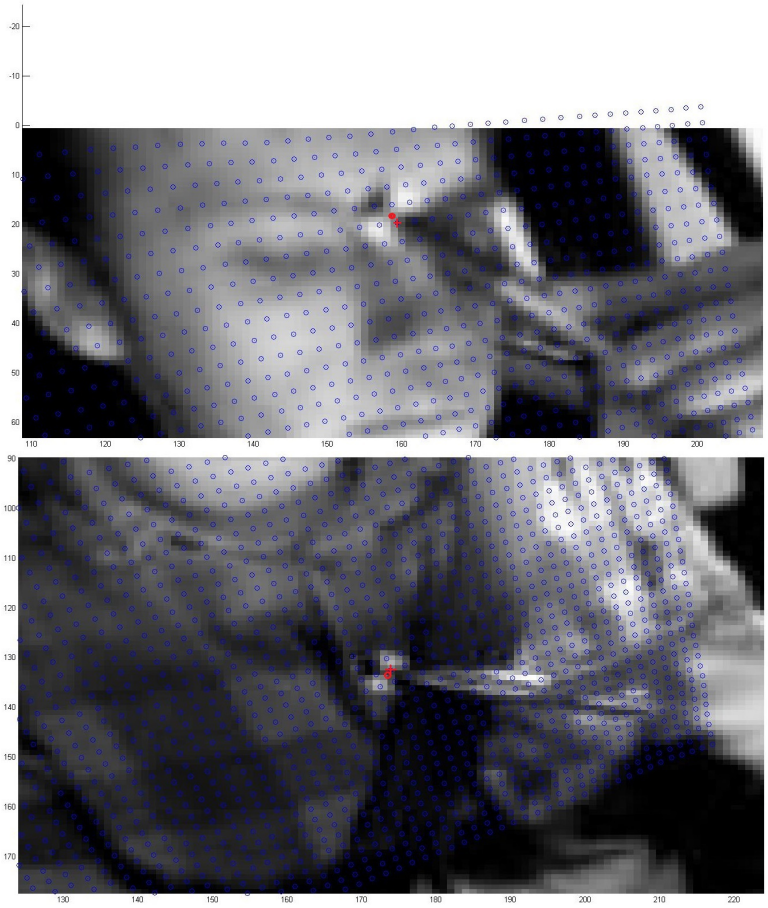
Figur 16: Samtliga tredimensionella punkter för en skiva där röda punkter är uppmätta hörn på schackbrädet och blåa punkter är utökade.

skapas. För att skapa 15 skivor framåt mot ratten räknas avståndet mellan skivorna ut och divideras med 15 (d.v.s. medelvärdet), i x-,y- och z. Medelvärdet adderas med koordinaterna för sista skivan, tills dess att 15 nya skivor har skapats.

3.3.4 *Sambandet mellan bild- och världspunkter.*

Beräkning av antalet verkliga schackrutor som ett objekt innefattar gjordes genom division av objektets längd med längden av en verklig schackruta. Markering av objektet i bilden genererar x och y koordinat för ändpunkterna på objektet. X- och y koordinater för objektets två ändpunkter användes sedan för att räkna ut det minsta avståndet till alla punkter från de tvådimensionella bildpunkterna. De punkter i varje skiva där avståndet var som minst sparas undan, se fig.17. Index av rad för varje punkt i varje skiva används för att beräkna rutbredden mellan ändpunkterna och index för kolumn används för att beräkna ruthöjden mellan ändpunkterna. Därefter beräknas hypotenusan mellan ändpunkterna för varje skiva. Avståndet i "schackrutor" mellan ändpunkterna jämförs sedan med antalet schackrutor objektet har i verkligheten.
De skivor som gav lägst differens väljs ut och då har motsvarande världspunker även hittats, eftersom bildpunkter och världspunkter indexerats på samma sätt. Dessa världspunkter från de utvalda skivorna jämförs sedan en i taget med alla motsvarande ändpunkter. Verkligt avstånd från ändpunkt till ändpunkt beräknas och det avstånd med minst devation från verkliga längden väljs ut. Då har båda ändpunkter på respektive skiva hittats. På detta sätt kringgick vi bristen av information i djupled för bilderna och gjorde så att ändpunkter

kunde ligga i olika skivor.



Figur 17: Algoritmen väljer ut den punkt (blå cirkel med rött kryss) från de tvådimensionella bildpunkterna (blåa cirklar) som är närmast den utpekade punkten (röd cirkel)

Algoritmens avståndsberäkningar när den valt världspunkter från bildpunkterna räknas ut med hjälp av tredimensionellt Euklidiskt avstånd.

$$s(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + (q_3 - p_3)^2} \tag{3}$$

Där p och q är två tredimensionella punkter och $s(q,p)$ är sträckan mellan punkterna. I vårt fall är den ena punkten satt till origo och den andra till koordinaterna för utvalda världspunkten.

3.3.5 *Feluppskattning för scenario*

För att räkna ut felet på algoritmens beräkningar från scenariot användes relativt fel.

$$\eta = \left| \frac{v - v_{approx}}{v} \right| \tag{4}$$

Där v är verkliga längden och v_{approx} är uträknade längden. Felet i avstånd till kameran eller ratten ses i både millimeter och procent.

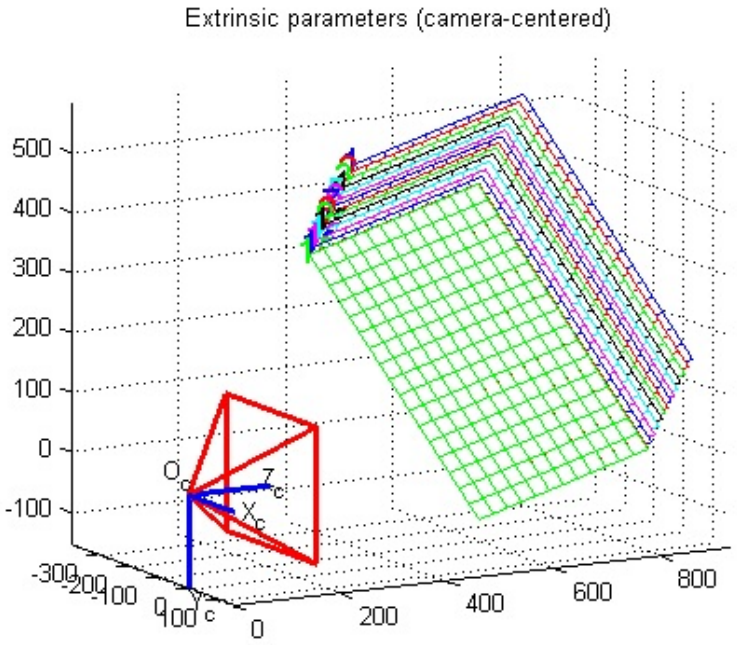
4

RESULTAT

Nedan följer resultaten från bildkorrigering, uppmätning och algoritmutveckling.

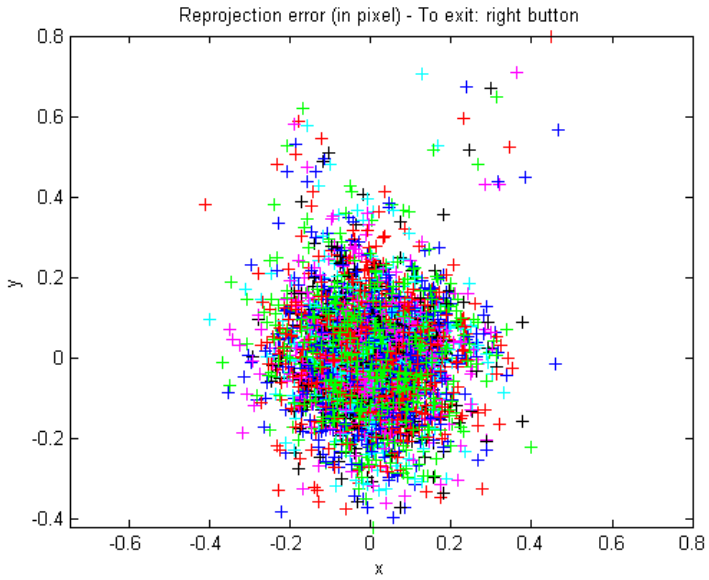
4.1 BILDKORRIGERING

Resultatet av vår kamerakalibrering från schackbrädesuppmätningen, som gjordes i *Camera Calibration Toolbox*, resulterade i visualiseringen av våra yttre kameraparametrar enl. fig.18.

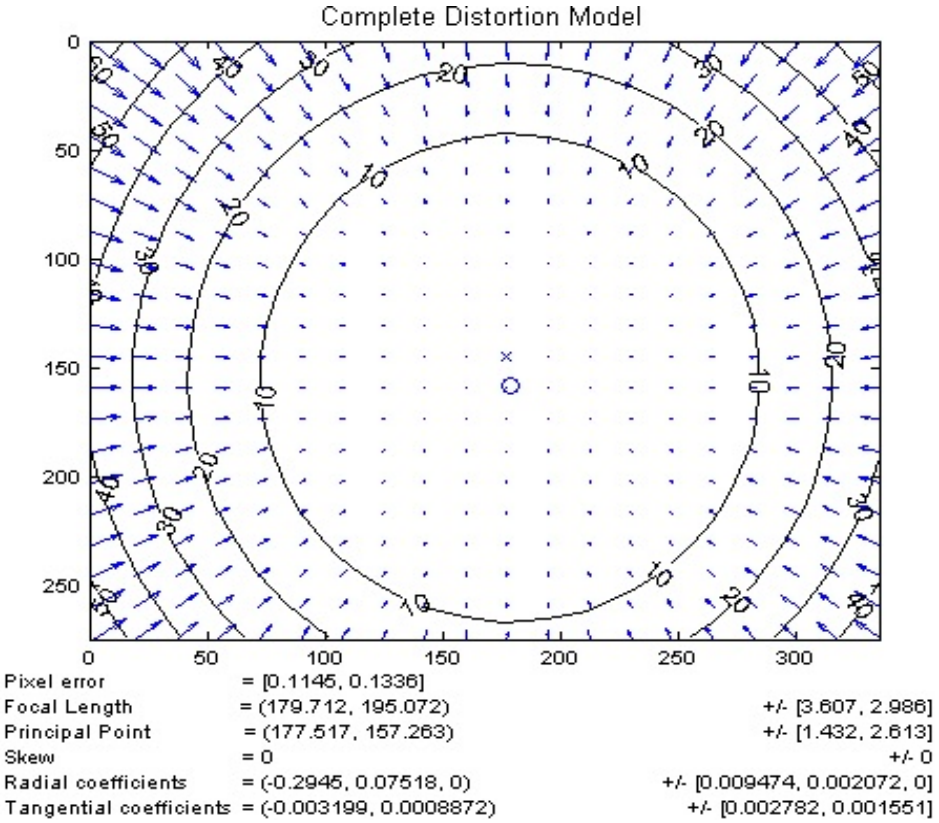


Figur 18: Yttre parametrar (Kameracentrerade), [Bouguet, 2010]

Pixelfelet visas i sin tur i ett diagram där de utprickade hörnen på schackbrädet i kalibreringsverktyget visas som kryss. Olika färger betyder olika bilder och schackrutshörn, se fig.19. Pixelfelet är lägre än en pixel per schackrutshörn, vilket tyder på en bra kalibrering av kameran. Den kompletta distorsionsmodellen, som visar magnituden av radiell distorsion, ses i fig.20. Under figuren står även information om bl.a. brännvidd, huvudpunkt och distorsionskoefficienter.



Figur 19: Pixelfel i kalibreringsbilderna efter kamerakalibrering.



Figur 20: Distorsionsmodell av schackbrädesuppmätningsbilderna, [Bouguet, 2010]

Den slutgiltliga rektifieringen av våra mockupbilder efter kamerakalibrering ses nedan, i fig.21. Synfältet är mindre, eftersom bilden "plattas till" och objekt i bilden är inte lika avrundade och oproportionerliga.

(a) **(b)**

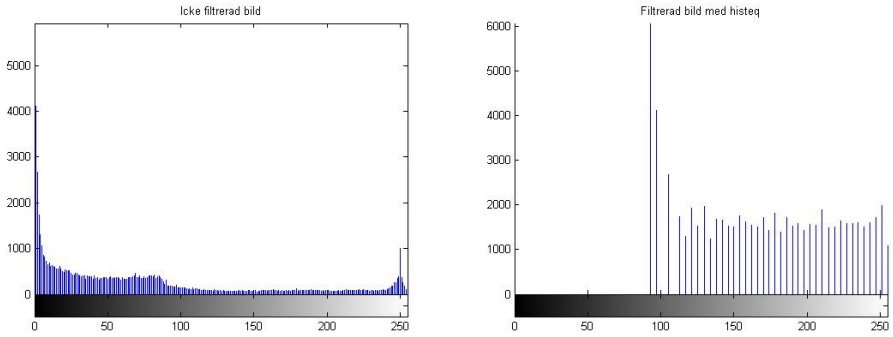
Figur 21: Scenariobild (a) före- och (b) efter rektifiering, [Bouguet, 2010].

4.1.1 *Filter*

Testning visade att värdet 65 som invariabel för matlabfunktionen *histeq* gav en bättre normalfördelning av histogrammet, vilket är önskvärt eftersom bilderna var mörka tidigare. En ljusare bild medför att det är lättare att mäta upp objekt i algoritmen, som kan varit för mörka att se tidigare.

(a) **(b)**

Figur 22: Före (a)- och efter (b) filtrering.



(a) **(b)**

Figur 23: Histogram före- och efter filtrering från *Image Processing Toolbox*, [Mathworks, 2013]. Underexponerat i bild (a) och mer normalfördelat i (b).

4.2 SIMULERADE SCENARIER

Våra resultat för algoritmen grundar sig på sex scenarier. Samtliga punkter prickas ut och beräknas två gånger. Röda punkter är första beräkningen, blå punkter är andra beräkningen. De relevanta punkterna för oss är på bröstet och i ansiktet vid näsroten. Avslutningsvis, presenteras rörelserna mellan de olika scenarierna, både uppmätt och uträknat.

4.2.1 *Scenario 1: normal förarposition, 2 meter lång man*

Figur 24: Scenario 1: normal förarposition, 2 meter lång man

Tabell 1: Punkter i ansiktet, röda och blåa kryss i fig.24

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	484.50	1.70	324.60	583.19
Uträknat	566.44	-53.12	334.27	659.86
Fel	-81.94	54.82	-9.67	-76.67

Algoritmen valde samma punkt för både det röda och det blåa krysset i ansiktet. Felet från ratten till ansiktet är således 13.15 % för båda punkterna.

Tabell 2: Punkter i bröstet, röda och blåa kryss i fig.24

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	487.00	11.4	-48.30	489.52
Uträknat	455.83	15.38	-27.35	456.91
Fel	31.17	-3.98	-20.95	32.61

Algoritmen valde samma punkt för både det röda och det blåa krysset i bröstet. Felet från ratten till bröstkorgen är således 6.66 % för båda punkterna.

4.2.2 *Scenario 2: autonom inbromsning, 2 meter lång man*

Figur 25: Scenario 2: autonom inbromsning, 2 meter lång man

Tabell 3: Punkter i ansiktet, röda och blåa kryss i fig.25

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	276.00	7.50	313.00	417.37
Uträknat	482.51	-144.90	282.39	577.54
Fel	-206.51	152.40	30.61	-160.17

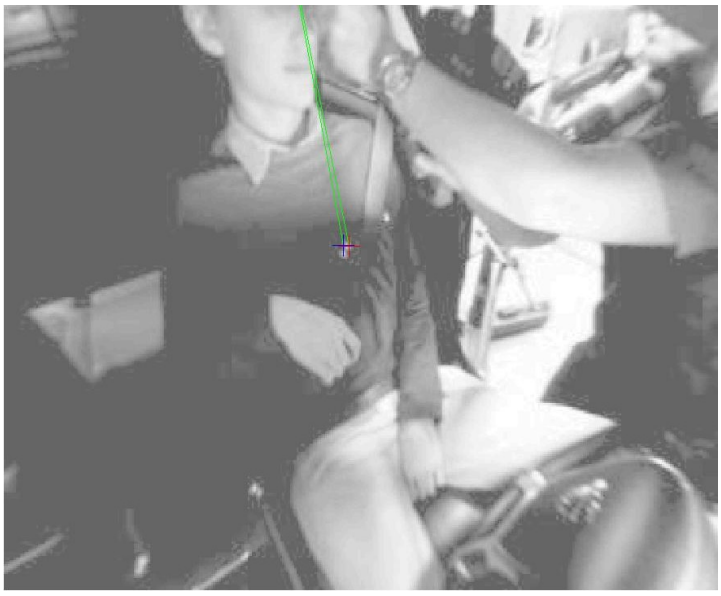
Algoritmen valde samma punkt för både det röda och det blåa krysset i ansiktet. Felet från ratten till ansiktet är således 38.38 % för båda punkterna.

Tabell 4: Punkter i bröstet, röda och blå kryss i fig.25

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	361.20	-48.40	0.10	364.43
Uträknat	384.17	-84.18	-32.07	394.59
Fel	-22.97	35.78	32.17	-30.16

Algoritmen valde samma punkt för både det röda och det blåa krysset i bröstet. Felet från ratten till bröstkorgen är således 8.28 % för båda punkterna.

4.2.3 *Scenario 3: manuell inbromsning, 2 meter lång man*



Figur 26: Scenario 3: manuell inbromsning, 2 meter lång man

Tabell 5: Punkt i ansiktet, rött kryss i fig.26

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	555.40	11.00	342.20	652.45
Uträknat	572.33	0.30	348.52	670.10
Fel	-16.93	10.70	-6.32	-17.65

Felet från ratten till ansiktet är 2.71 % för första punkten, röda krysset.

Tabell 6: Punkt i ansiktet, blått kryss i fig.26

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	555.40	11.00	342.20	652.45
Uträknat	592.58	-15.15	346.27	686.50
Fel	-37.17	26.15	-4.05	-34.05

Felet från ratten till ansiktet är 5.22 % för andra punkten, blåa krysset.

Tabell 7: Punkt i bröstet, rött kryss i fig.26

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	493.50	1.30	13.60	493.69
Uträknat	472.45	-6.40	27.96	473.32
Fel	21.05	7.7	-14.36	20.37

Felet från ratten till bröstkorgen är 4.13 % för första punkten, röda krysset.

Tabell 8: Punkt i bröstet, blått kryss i fig.26

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	493.50	1.30	13.60	493.69
Uträknat	489.81	-6.97	10.12	489.96
Fel	3.69	8.27	2.94	3.73

Felet från ratten till bröstkorgen är 0.75 % för andra punkten, blåa krysset.

4.2.4 *Scenario 4: normal förarposition, 1.80 meter lång man*

Figur 27: Scenario 4: normal förarposition, 1.80 meter lång man

Tabell 9: Punkt i ansiktet, rött kryss i fig.27

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	442.80	34.30	309.10	541.10
Uträknat	470.39	11.07	327.06	573.03
Fel	-27.59	23.23	-17.96	-31.93

Felet från ratten till ansiktet är 5.90 % för första punkten, röda krysset.

Tabell 10: Punkt i ansiktet, blått kryss i fig.27

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	442.80	34.30	309.10	541.10
Uträknat	482.37	3.13	318.07	577.81
Fel	-39.57	31.17	-8.97	-36.71

Felet från ratten till ansiktet är 6.78 % för andra punkten, blåa krysset.

Tabell 11: Punkt i bröstet, rött kryss i fig.27

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	414.30	37.70	15.10	416.29
Uträknat	378.88	33.79	37.30	382.21
Fel	35.42	3.91	-22.20	34.08

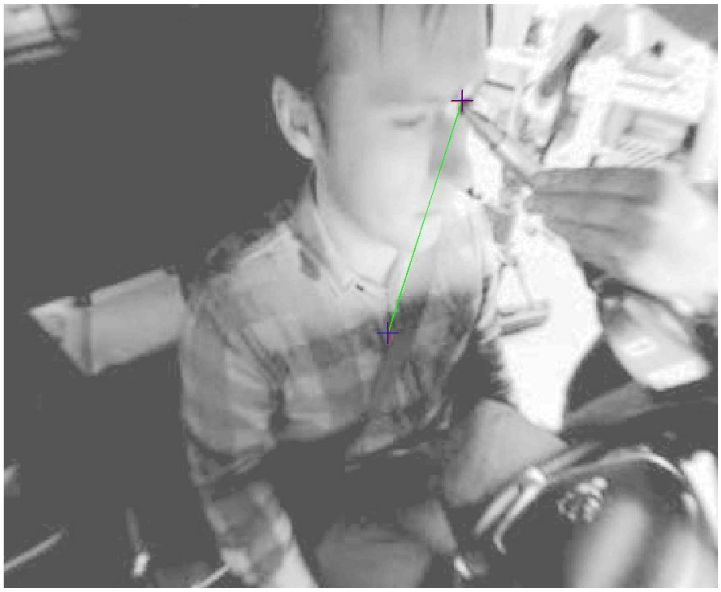
Felet från ratten till bröstkorgen är 8.19 % för första punkten, röda krysset.

Tabell 12: Punkt i bröstet, blått kryss i fig.27

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	414.30	37.70	15.10	416.29
Uträknat	391.01	33.50	28.04	393.44
Fel	23.29	4.20	-12.94	22.84

Felet från ratten till bröstkorgen är 5.49 % för andra punkten, blåa krysset.

4.2.5 *Scenario 5: autonom inbromsning, 1.80 meter lång man*



Figur 28: Scenario 5: autonom inbromsning, 1.80 meter lång man

Tabell 13: Punkter i ansiktet, röda och blåa kryss i fig.28

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	210.40	61.30	301.90	373.05
Uträknat	365.70	-145.13	245.95	463.99
Fel	-155.30	206.43	55.95	-90.94

Algoritmen valde samma punkt för både det röda och det blåa krysset i ansiktet. Felet från ratten till ansiktet är således 24.38 % för båda punkterna.

Tabell 14: Punkter i bröstet, röda och blåa kryss i fig.28

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	306.30	41.90	-18.00	309.68
Uträknat	280.79	24.01	-33.18	283.76
Fel	25.51	17.89	15.18	25.92

Algoritmen valde samma punkt för både det röda och det blåa krysset i bröstet. Felet från ratten till bröstkorgen är således 8.37 % för båda punkterna.

4.2.6 *Scenario 6: manuell inbromsning, 1.80 meter lång man*

Figur 29: Scenario 6: manuell inbromsning, 1.80 meter lång man

Tabell 15: Punkt i ansiktet, rött kryss i fig.29

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	471.80	52.00	317.70	571.17
Uträknat	473.29	42.55	333.84	580.74
Fel	-1.49	9.45	-16.14	-9.57

Felet från ratten till ansiktet är 1.68 % för första punkten, röda krysset.

Tabell 16: Punkt i ansiktet, blått kryss i fig.29

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	471.80	52.00	317.70	571.17
Uträknat	461.21	42.78	343.07	576.40
Fel	10.59	9.22	-25.37	- 5.23

Felet från ratten till ansiktet är 0.92 % för andra punkten, blåa krysset.

Tabell 17: Punkt i bröstet, rött kryss i fig.29

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	419.60	44.30	31.20	423.08
Uträknat	383.95	41.97	52.65	389.81
Fel	35.65	2.33	-21.45	33.27

Felet från ratten till bröstkorgen är 7,87 % för första punkten, röda krysset.

Tabell 18: Punkt i bröstet, blått kryss i fig.29

	X [mm]	Y [mm]	Z [mm]	Euklidiskt avstånd till ratten[mm]
Uppmätt	419.60	44.30	31.20	423.08
Uträknat	372.02	57.71	61.65	381.48
Fel	47.58	-13.41	-30.45	41.60

Felet från ratten till bröstkorgen är 9.83 % för andra punkten, blåa krysset.

4.2.7 *Rörelse från scenarier*

Alla rörelser är uträknade med ratten i origo. Positiva rörelser betyder rörelse mot ratten och negativa från ratten.

Rörelse för bröstet mellan scenarier 1, 2 och 3:

Tabell 19: Rörelse i millimeter för bröst mellan scenarier.

Rörelse	Scenarier 1 och 2	Scenarier 1 och 3	Scenarier 2 och 3
Uppmätt [mm]	125.09	4	129.26
Uträknat [mm]	62.32	16.41	78.73
Fel [mm]	62.77	12.41	50.53

Rörelse för ansikte mellan scenarier 1, 2 och 3:

Tabell 20: Rörelse i millimeter för ansikte mellan scenarier.

Rörelse	Scenario 1 och 2	Scenario 1 och 3	Scenario 2 och 3
Uppmätt [mm]	165.82	-69.26	235.08
Uträknat [mm]	82.33	-26.64	108.96
Fel [mm]	83.49	-42.62	126.12

Rörelse för bröstet mellan scenarier 4, 5 och 6:

Tabell 21: Rörelse i millimeter för bröst mellan scenarier.

Rörelse	Scenarier 4 och 5	Scenarier 4 och 6	Scenarier 5 och 6
Uppmätt [mm]	106.61	-6.79	113.4
Uträknat [mm]	109.68	3.63	106.5
Fel [mm]	3.07	-10.42	7.35

Rörelse för ansikte mellan scenarier 4, 5 och 6:

Tabell 22: Rörelse i millimeter för ansikte mellan scenarier.

Rörelse	Scenarier 4 och 5	Scenarier 4 och 6	Scenarier 5 och 6
Uppmätt [mm]	168.05	-30.07	198.12
Uträknat [mm]	109.01	-7.71	116.75
Fel [mm]	59.04	-22.36	81.37

5

DISKUSSION OCH SLUTSATS

I denna rapport har vi utvecklat en metod för att bedöma tredimensionella positioner i en Volvo XC70 från tvådimensionella bilder. Detta är tillämpat i ett specialfall, för att utvärdera om vårt tillvägagångssätt fungerade. Metoden går att tillämpa till FOT-projektet om vissa korrigeringar görs, (se framtida tillämpningar och förbättringar).
Detta arbete har till stor mån testat vår förmåga att utveckla ingenjörsmässiga lösningar med hjälp av bildbehandling, matematik och programmering. Beräkningar av koordinater och dimensioner av objekt från bilder är ett relativt outforskat område, som på senare tid blivit mer aktuellt. Detta betydde att vår uppgifts område inte utforskats i stor omfattning tidigare, vilket medförde att vi fick tänka i okonventionella banor.

5.1 BEDÖMNING AV RESULTAT

Vi förstod i ett tidigt skede att bedömning av resultat skulle bli svårt. Därför skapade vi ett scenario där vi kunde kontrollera alla omständigheter och samtidigt bestämma alla variabler. På så sätt skapade vi ett facit som kunde användas vid bedömning av vår algoritm. Från början tänkte vi ha åtta bilder på scenarion där vi simulerade olika hårda inbromsningar, men efter rektifiering insåg vi att i vissa bilder hamnade huvudet utanför kamerans synfält och i andra var markören för huvudet inte synlig. Detta resulterade i att endast 6 bilder gav bra resultat.

Vi hade som mål att algoritmen skulle uppskatta positionen av förarens huvud eller bröst med max 10% fel. Vi uppnådde detta mål i 10 av 12 punkter, men resterande gav ett fel mellan 14 och 38 procent. Samtliga fel var dock med huvudet som mätpunkt. De största felen förekom även endast vid simulering av de kraftiga inbromsningarna. Dessa fel kan emellertid förklaras, men orsakerna är flera.

- Lutningen mellan punkten i bröstet och huvudet är nära vinkelrät mot de uppmätta schackbrädena. I detta fall behövs information om storlek i schackrutor mellan schackbrädena i bilder för beräkning. Detta är den störst bidragande orsaken till att scenariot *Scenario 2* har ett stort fel i x-led. Viktigt att veta är emellertid att mätpunkterna **inte** måste tillhöra samma schackbrädesskiva. Resultatet blir dock bättre om punkternas valda skivor ligger nära varandra, eftersom vi som tidigare nämnt in-

te har information om avståndet mellan schackbrädena i bildkoordinatsystemet.

- Avståndet mellan bildpunkterna är större nära de övre kanterna av bilden, på grund av att de är närmare kameran, vilket ger sämre uppskattning av världspunkt från bildpunkt.
- Ändpunkterna bildar en vertikal linje. Om punkterna väljs horisontellt, d.v.s. ligger i ungefär samma djup, fås en bättre approximation.
- Sambandet att objekt blir större ju närmare kameran objektet kommer, vilket är grunden i vår algoritm, försämras när betraktningsvinkeln inte är framför objektet. Därför ökar inte storleken av objektet linjärt i bilderna och kan därför ha avvikelser, vilket algoritmen ibland inte uppfattar.

Det maximala felet på 38% resulterade i en feluppskattning på 16 cm och rörelser i djupled vid inbromsning brukar inte överstiga 20 cm. Detta gör att uppskattningen av huvudet i detta scenario blir väldigt otillförlitligt. I de andra bilderna blev emellertid felet på punkterna mellan 2-4 cm, vilket givetvis är bättre. För att sätta detta i sammanhang, var rörelsen i djupled för bröstet mellan *Scenario 1* och *Scenario 2* 12 cm i verkligheten. Vår uppskattning gav ett samlat fel på ungefär 5 cm mellan scenarierna, även om felet för båda punkterna var under 8%. Dessa fel ger en residual på 5 cm av en rörelse på 12 cm. Detta innebär givetvis att även detta resultat är ganska otillförlitligt.

Trots detta kan fortfarande riktningen av rörelsen uppskattas, så länge rörelsen inte är mindre än några centimeter, eller större än schackbrädenas definitionsområde.

Efter att ha undersökt på internet efter liknande problemställningar kom vi bland annat fram till att uppskatta var en punkt befinner sig i rummet kan också göras med att enbart använda kamerans parametrar. För korta avstånd finns det ett OpenCV-program för uppskattning av avstånd från en kamera [Khosravi, 2009]. Därför använde vi även detta program för att se hur bra uppskattningar detta program gjorde med våra kameraparametrar och rektifierade bilder. Programmets beräkningar gav ett genomgående fel på ungefär 10 cm, oberoende av scenario. Detta tycker vi ger en bra uppskattning på hur stora fel man kan räkna med att ha med vår kamerauppsättning.

5.2 FELKÄLLOR

Algoritmen bygger på mätningar, approximeringar och korrigeringar. Detta har medfört att vi har flera felkällor.

- En mänsklig felkälla för algoritmen är att användaren kan träffa olika nära mätpunkten när den ska trycka på positionen av ett objekt. En annan är att uppmätningen av schackbrädets hörn inte var exakt.
- Beräkningarna för att skapa ett tredimensionellt rum av mätpunkter bygger på medelvärden och linjär regression, vilket skapar approximerade värden. Detta gäller även för beräkningar av nya tvådimensionella bildpunkter.
- Schackbrädesmätningen skedde från stolens ändpunkter med ett intervall av en sekund framåt, vilket gjordes för att vi inte hade någon information om stolskenan, vilket medför att mätpunkterna är kontinuerliga, men inte har ett symmetriskt intervall.
- Kamerans vinkel gör att objektet från scenarion ligger väldigt högt upp i bilden. Tunndistorsion i bilderna medför därför att antalet pixlar som approximerades fram för att "dra ut" bilden har även ett fel. Om objektet emellertid befunnit sig mer centrerat i bilden hade antalet korrekta pixlar varit högre.

5.3 FRAMTIDA TILLÄMPNINGAR OCH FÖRBÄTTRINGAR

Vår researchdel av examensarbete visade att *Stereoskopisk* kamerauppsättning brukar vara att föredra när avstånds- och storleksberäkningar på bilder ska utföras. Då fanns det dessutom mycket material, såsom programkod i Matlab, att tillgå. Det finns till och med en programmeringsmiljö "OpenCV", som är skräddarsydd för framför allt stereoskopisk bildbehandling.

Problemet med att tillämpa vår algoritm till FOT-projektet var alla variabler. Stolsposition, vinkel på kameran och mångfalden av förare är bara några få exempel på saker som måste generaliseras eller på något sätt kvantifieras för att ge en precis och generell lösning. Detta hade emellertid förenklats om informationsinsamlingen inför projektet varit mer omfattande och kamerauppsättningen varit speciellt utformad för avståndsmätning.
En önskvärd kamerauppsättning hade som tidigare nämnt varit stereoskopisk med, om möjligt, en lasermätare som kontinuerligt mätte avståndet till ett eller flera objekt. Denna teknik är mycket etablerad på marknaden och tillämpas bland annat av Microsoft i produkten Kinect. Om endast en monoskopisk kamerauppsättning är möjligt att tillgå, rekommenderar vi att placeringen av kameran ger den ett synfält som är vinkelrätt mot det man avser att beräkna avståndet från. I vårt fall hade vi placerat kameran vid passagerarsätet eller B-stolpen vid förarstolen. Detta hade gett oss ett synfält som fångat förarens

djupledsrörelsen mot ratten, eftersom avståndet till ratten var det väsentliga. I detta fall hade vi använt stationära referenspunkter för att kvantifiera var föraren befinner sig. En analogi till detta tankesätt är att om man kan uppskatta med ett öga hur långt en längdhoppare hoppat, kollar man hellre från sidan, än framför.

För att tillämpa vår metod på FOT-projektet, bör ungefär samma synfält som scenarierna användas. Vår uppskattning är att samma stationära punkter måste synas i bild efter rektifiering för att det ska fungera. Om detta ej är fallet behövs de variabla schackbrädesmätningarna mätas om och rektifieras, för att skapa både ett nytt tredimensionellt och tvådimensionellt nät av mätpunkter. Om tre stationära referenspunkter finns i synfältet för kamerorna i FOT, kan dess tredimensionella position beräknas och då kan även synfältet för alla kameror i FOT-bilarna beräknas. På så vis kan möjligen en "offset"-vinkel beräknas och en kompensationskoefficient användas för vår metod, för att undvika en upprepning av uppmätningar.

Bouguet, J.-Y. [2010], *Camera Calibration Toolbox for Matlab*. Tilllgänglig på: <http://www.vision.caltech.edu/bouguetj/calib_doc> (Åtkomstdatum: 28.05.2013).

Breen, J. [2004], *World Report On Road Traffic Injury Prevention*, World Health Organization. <http://who.int/violence_injury_prevention/publications/road_traffic/world_report/en> (Åtkomstdatum: 28.05.2013).

FARO Technologies [2013]. Tillgänglig på: <http://www.faro.com/en-us/products/metrology/measuring-arm-faroarm/overview> (Åtkomstdatum: 04.06.2013).

Groß, J. [2003], *Linear Regression*, Springer Berlin Heidelberg.

Guerlain, P. och Durand, B. [2006], *Digitizing and Measuring of the Human Body For the Clothing Industry*. Tillgänglig på: <http://www.emeraldinsight.com/content_images/fig/0580180301006.png> (Åtkomstdatum: 04.06.2013).

Heikkilä, J. och Silvén, O. [1997], *A Four-step Camera Calibration Procedure with Implicit Image Correction*, Infotech Oulu and Department of Electrical Engineering.

Khosravi, M. R. [2009], *Range Finder*. Tillgänglig på: <http://www.codeproject.com/Articles/35029/Range-Finder> (Åtkomstdatum: 06.06.2013).

KT&C [2013]. Tillgänglig på: <http://www.ktnc.co.kr/english/viewtopic.php?t=438#> (Åtkomstdatum: 04.06.2013).

Mathworks [2013], *Image Processing Toolbox*. Tillgänglig på: <http://www.mathworks.se/products/image/> (Åtkomstdatum: 04.06.2013).

Romero, L. och Gomez, C. [2007], *Correcting Radial Distortion of Cameras With Wide Angle Lens Using Point Correspondences, Scene Reconstruction Pose Estimation and Tracking*, InTech Europe. Tillgänglig i biblioteket på: <http://www.intechopen.com/books> (Åtkomstdatum: 27.05.2013).

Sonka, M., Hlavac, V. och Boyle, R. [2008], *Image Processing, Analysis, and Machine Vision*, Thomson.

Starizona [2013]. Tillgänglig på: <http://starizona.com/acb/basics/optics/distortion.jpg> (Åtkomstdatum: 04.06.2013).

Szeliski, R. [2010], *Computer Vision: Algorithms and Applications*, Springer.

Zhang, Z. [1999], *Flexible Camera Calibration By Viewing a Plane From Unknown Orientations*, Microsoft Research.

BILAGOR

A

STATIONÄRA PUNKTER

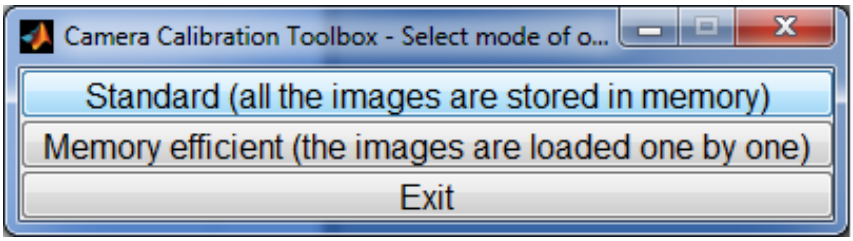
Figur 30: Bild på stationära punkter, se lista nedan för information om resp. punkt.

1. Volvologga på ratten
2. Bstolpe nere
3. Låsknopp
4. Handtag
5. Säte höger mitt
6. Säte vänster mitt
7. Huvud höger
8. Konsol vänster
9. Konsol höger

B

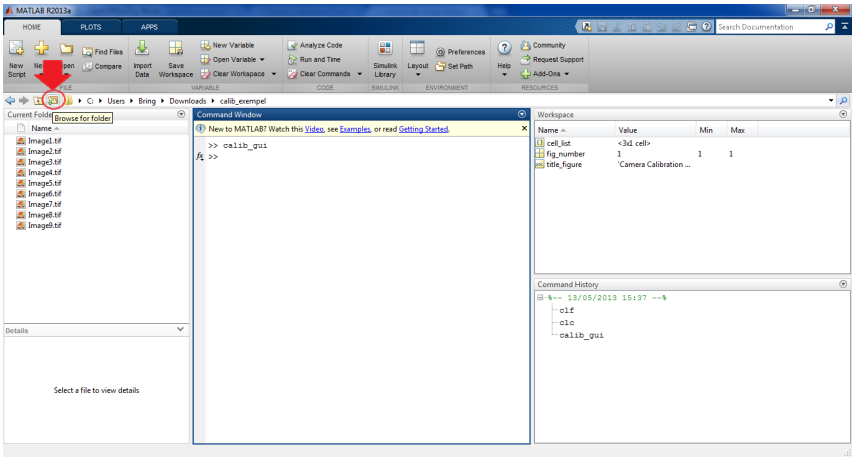
ANVÄNDARGUIDE FÖR BILDKORRIGERING

1. Ladda ned *Camera Calibration Toolbox* från `http://www.vision.caltech.edu/bouguetj/calib_doc/download/index.html` (och vid behov, exempelbilderna jag använder från `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html`)
2. Extrahera innehållet.
3. Lägg till Camera calibration toolbox till Matlab's sökväg genom Set path → Add folder.
4. Skriv calib_gui i kommandofönstret så ska detta fönster komma upp.



Figur 31

5. Välj hur du vill ladda in bilderna.
6. Ändra mappen Matlab arbetar ur till den som bilderna ligger i.

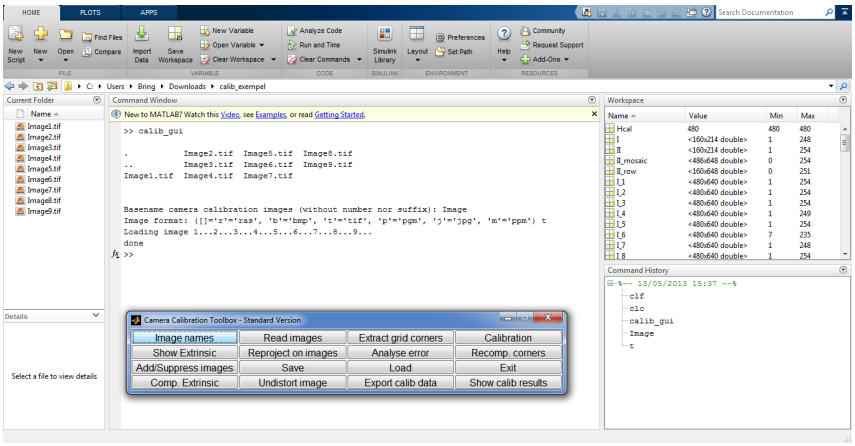


Figur 32

7. Tryck “Image names” i GUI’n.

8. Skriv in kalibreringsbildernas basnamn (d.v.s. utan suffix eller nummer). I detta fall “Image”.

9. Skriv formatet enl. prompten I kommandofönstret. I detta fall “t” som i tif.



Figur 33

10. Nu visas bilderna i en figur i Matlab. Tryck på “Extract grid corners” i GUI’n.

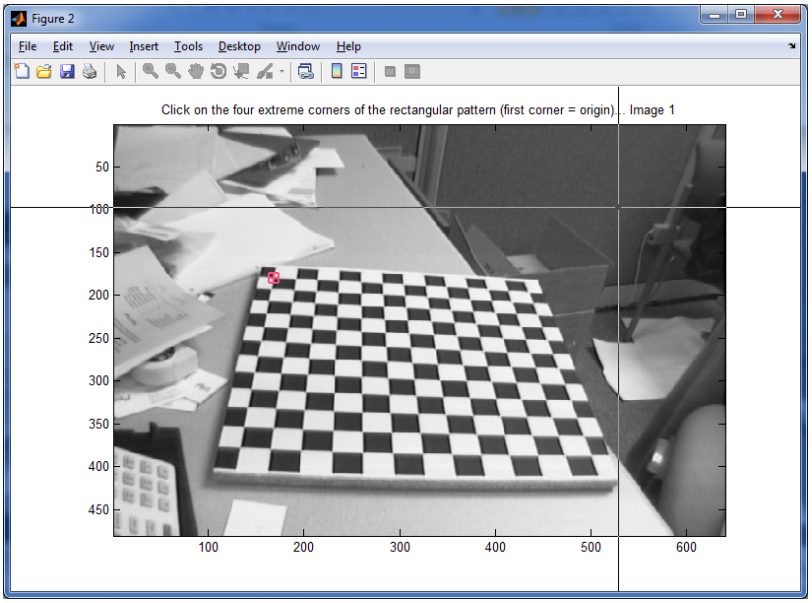
11. Ange antalet bilder som ska kalibreras. Tryck “Enter” för att kalibrera alla.

12. Ange storleken på schackbrädesrutorna i x- och y-led. Tryck åter igen “Enter” för att använda standard.

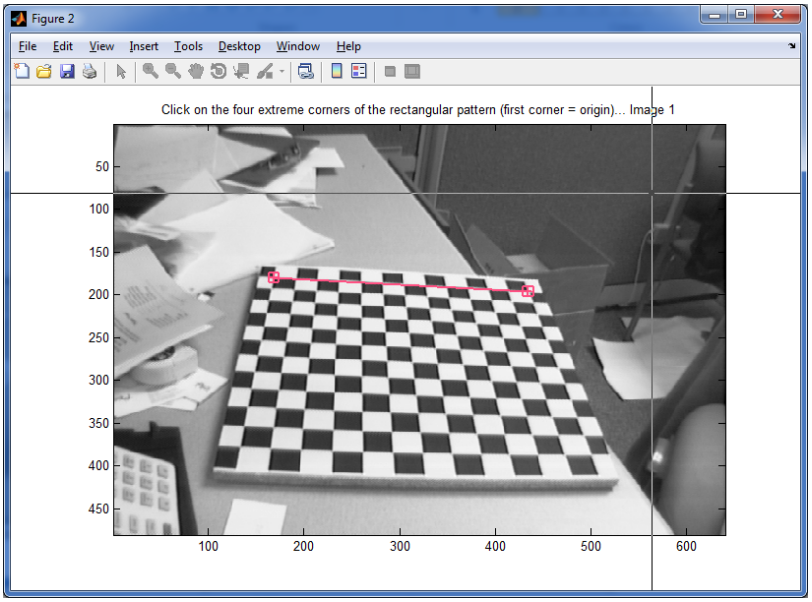
13. Därefter, ange om rutorna ska räknas manuellt eller om toolbox-en ska räkna automatiskt. Tryck “Enter” för automatisk.

14. Klicka på de fyra näst mest extrema hörnen på schackbrädet från uppe i vänster hörn moturs till längst ner i vänster hörn.

    Var god se nästa sida vid osäkerheter. De röda kvadraterna visar musklickningarna.

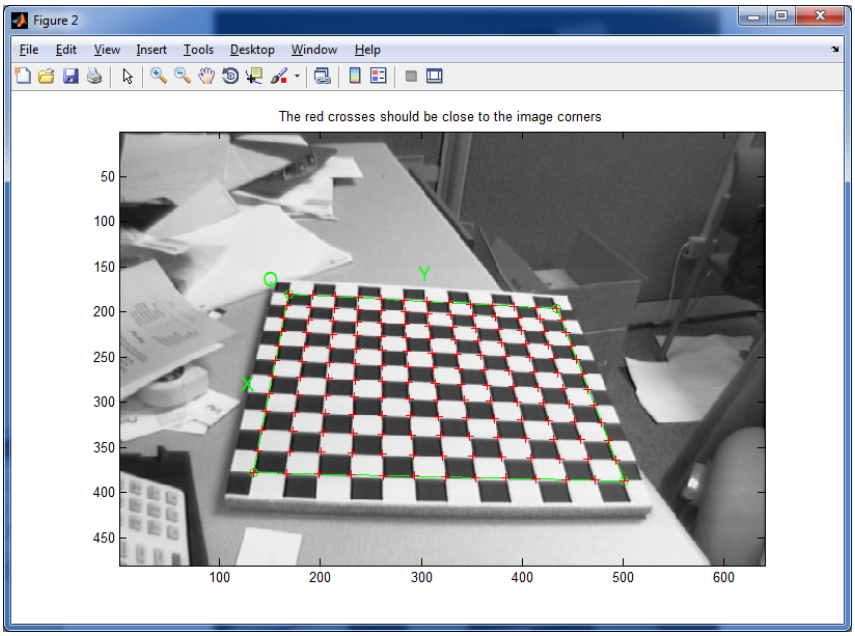


Figur 34



Figur 35

15. Efter alla hörn är markerade ska storleken i x- och y-led på schackbrädesrutorna i verkligheten anges i kommandofönstret. Då markerar programmet automatiskt ut kryss i alla hörn på schackbrädesrutorna i bilden.



Figur 36

16. Nu har du möjlighet att korrigera kryssen med hjälp av en distorsionskoefficient som du skriver in i kommandofönstret. Värdet brukar ligga mellan -1 och $+1$.

17. Repetera steg 14) och 16) tills du markerat alla hörn på alla kalibreringsbilder.

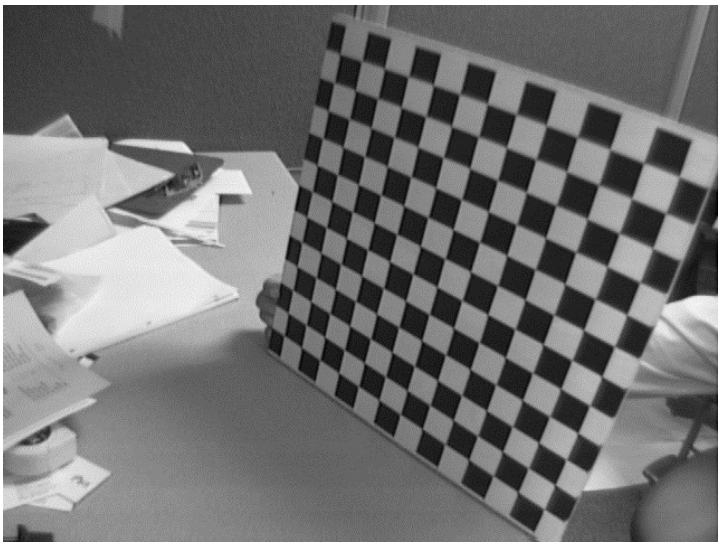
18. Tryck på "Calibration" i GUI-fönstret. Tryck dessutom på "Save" om du vill spara.

19. Välj hur du vill ladda in bilderna.

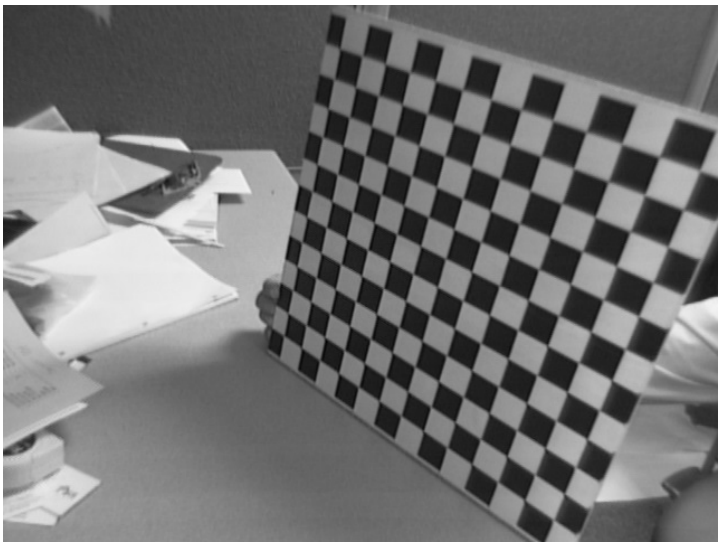
    Efter detta är kalibreringen klar. En 3D-visualisering av schackbrädena i förhållande till kameran kan ses genom att trycka på "Show extrinsics" i GUI'n. Pixelfelet syns om "Analyse error" trycks. Man kan även spara bildkoordinaterna av kryssen vid schackbrädeshörnen genom att trycka på "Export calib data". Om "visualize_distortions" skrivs i kommandofönstret syns dessutom distorsionen i bilderna. Det finns även fler funktioner, som kan läsas mer om på `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html`.

20. Tryck på "Undistort image" för att rektifiera bilderna. Då har du möjlighet att välja att rektifiera alla kalibreringsbilder eller

att välja en annan genom att skriva “1” i kommandofönstret och sedan repetera steg 8) och 9). De rektifierade bilderna läggs i mappen med dina kalibreringsbilder som “basnamn_rect”.



Figur 37: innan rektifiering



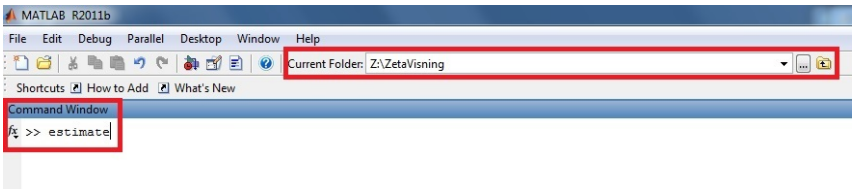
Figur 38: Efter rektifiering

C

ANVÄNDARGUIDE FÖR ALGORITM

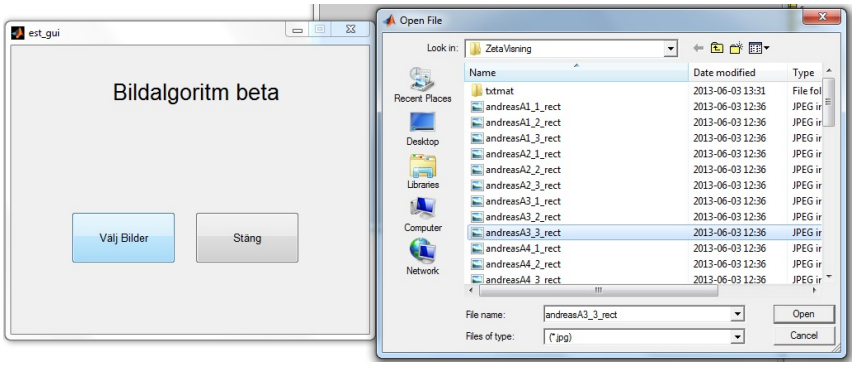
1. Starta Matlab och ställ in “Current Folder“ till den mapp där scriptet för algoritmen ligger, se fig.35.

2. Skriv estimate i “Command Window“, se fig.35.



Figur 39

3. Välj två eller fyra bilder som skall analyseras.

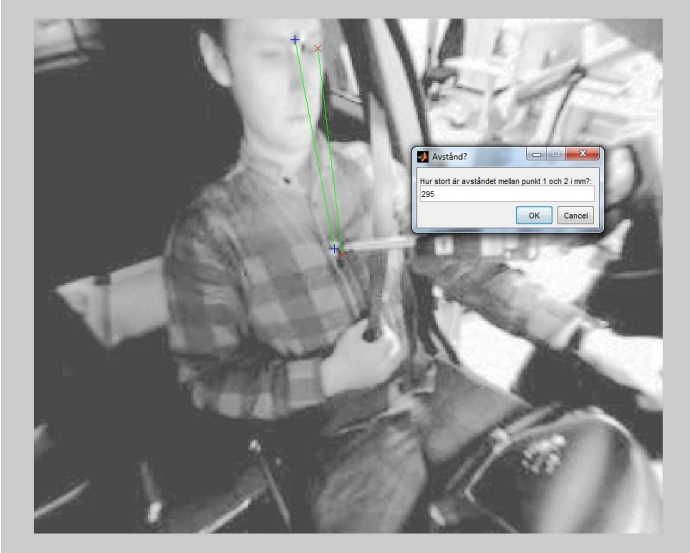


Figur 40

4. Pricka ut fyra punkter för ett objekt se fig.37.

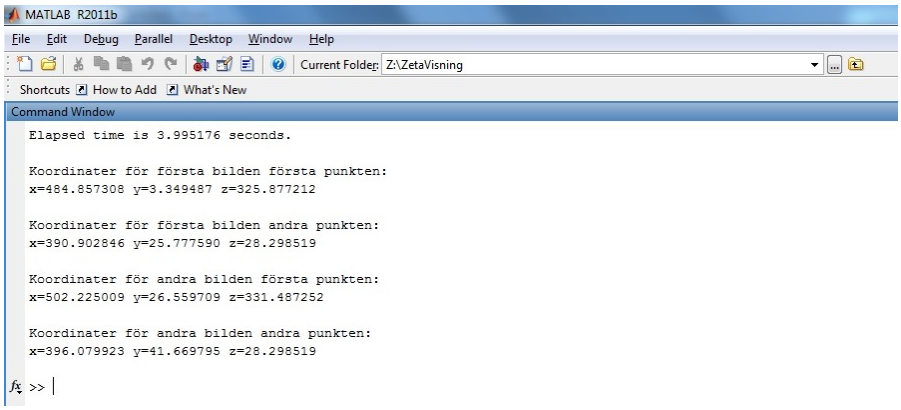
Figur 41

5. Plottar ut var du valt att pricka ut objektet se fig.38.



Figur 42

6. Skriv in hur långt objektet är i mm se fig.38.



Figur 43

7. Koordinater för samtliga punkter ges i "Command Window".

D

PROGRAMKOD

```
% -- Company:       Volvo/Chalmers
% -- Engineers:      Kim Rosberg Andreas Bring
% -- Date of Creation:   08:00:00 04/22/2013
% -- Design Name:   estimate.m
% -- Project Name:  Quantify drivers' motions at hard brakings based
%                   on videos from real world driving.
% -- Tool versions: Matlab R2013a
% -- Description:   Input two or four pictures from FOT data to get 3D
%                   coordinates for 4 points.
% -- Errors: None
% -- Additional Comments:

clc
clf
clear all
tic %Räknar ut tiden för initiering
%allapkt(): Alla mätpunkter i ett skript, sparar till pkt.m. Behöver bara
% köras en gång. Sätter ratten i origo.
allapkt();
%init_b_skiva: Skapar tvådimensionella skivor med numeriska xy värden från
%film/bild från 15 skivor med 221 mätpunkter till 30 skivor 65 rader och
%49 kolumner
init_b_skiva();
%Skapar tredimensionellt rum från verkliga mätpunkter (15*4st) till
%30 skivor 65 rader och 49 kolumner
init_v_skiva();
sida = sruta();
toc
fprintf('\n');
%GUI för att välja bilder för analys
s = est_gui();
waitfor(s);
%Hämtar antalet bilder och om programmet ska stängas
load('antal_bilder')
load('stang')
if stang==1
    return;
end
%Skickar bilder och antalet bilder. Användaren pekar ut var
```

```
%objektets ändpunkter befinner sig, filter för höjd intensitet
    och mer
%konturer, ritar ut och sparar koordinater.
[ax ay bx by]=valjpkt(im,nr_of_pics);
%Spara undan koordinater
savefile = 'var.mat';
save(savefile, 'ax','ay','bx','by');
%Hämtar koordinater från mätning på schackbräde, ref-punkter och
    scenario
load('pkt.mat');
%Punkter i verkligheten, för felberäkning.
APUNKT=[KIMHUVUD1
    KIMHUVUD3];
BPUNKT=[KIMBROST1
    KIMBROST3];
%Eller bestäm avståndet.
prompt = {'Hur stort är avståndet mellan punkt 1 och 2 i mm?:'};
dlg_title = 'Avstånd?';
num_lines = 1;
def = {'2'};
c = inputdlg(prompt,dlg_title,num_lines,def);
x = cellfun(@(x)str2double(x), c);
rwdist=[x x];

load('var.mat'); %Hämta koordinater som är utpekade
% Hur många rutor är objektets verkliga längd på det
% tvådimensionella schackbrädet?
antalrutor=[rwdist(1)/sida rwdist(2)/sida];
%Hittar bästa 2D-koordinats index
[arad1,akolumn1,brad1,bkolumn1,arad2,akolumn2,brad2,bkolumn2,
    mskiva,nskiva]= bkoordfinder(ax,ay,bx,by,antalrutor);
%Hämtar 3D-koordinater
[Akoord1,Akoord2,Bkoord1,Bkoord2] = vkoordfinder(arad1,akolumn1,
    brad1,bkolumn1,arad2,akolumn2,brad2,bkolumn2,rwdist,mskiva,
    nskiva);

%Skriver ut samtliga 3D-koordinater programmet funnit
k=0;
fprintf('Koordinater för första bilden första punkten:\n')
for i=['x' 'y' 'z']
    k=k+1;
    fprintf('%s=',i)
    fprintf('%f ',Akoord1(k));
end
fprintf('\n\n');
k=0;
fprintf('Koordinater för första bilden andra punkten:\n')
for i=['x' 'y' 'z']
    k=k+1;
    fprintf('%s=',i)
    fprintf('%f ',Bkoord1(k));
end
```

```
87 fprintf('\n\n');
88 k=0;
89 fprintf('Koordinater för andra bilden första punkten:\n')
90 for i=['x' 'y' 'z']
91     k=k+1;
92     fprintf('%s=',i)
93     fprintf('%f ',Akoord2(k));
94 end
95 fprintf('\n\n');
96 k=0;
97 fprintf('Koordinater för andra bilden andra punkten:\n')
98 for i=['x' 'y' 'z']
99     k=k+1;
100    fprintf('%s=',i)
101    fprintf('%f ',Bkoord2(k));
102 end
103 fprintf('\n\n');
104
105 %Beräknar fel från ratten(origo) till verkliga koordinaten
106 felupp(Akoord1,Akoord2,Bkoord1,Bkoord2,APUNKT(1,:),APUNKT(2,:),
        BPUNKT(1,:),BPUNKT(2,:));
```

```
1  function [] = allapkt()
2
3  %RATTVOLVOLOGGAV(Ratten) i origo
4  RATTVOLVOLOGGAV=        [2726.3 -398.3 1065.0];
5  KAMERA=              [2743.0     -117.3 1450.4];
6  BSTOLPEUPPE=         [3317.5 -626.8 1469.1]-RATTVOLVOLOGGAV;
7  BSTOPLENERE=         [3258.1 -800.0 1139.6]-RATTVOLVOLOGGAV;
8  LASKNOPP=            [3170.8 -771.2 1136.7]-RATTVOLVOLOGGAV;
9  HANDTAG=             [2816.9     -730.7 1032.1]-RATTVOLVOLOGGAV;
10 SATE_VANSTER_MITT=       [3389.1 -245.0 963.2]-RATTVOLVOLOGGAV;
11 SOTE_HOGER_MITT=         [3388.0 -494.1 959.2]-RATTVOLVOLOGGAV;
12 HUVUDVANSTER=        [3439.1     -294.9 1404.9]-RATTVOLVOLOGGAV;
13 HUVUDHOGER=          [3439.7     -458.9 1402.1]-RATTVOLVOLOGGAV;
14 KONSOLHOGER=         [3059.9     -101.3 810.0]-RATTVOLVOLOGGAV;
15 KONSOLVANSTER=       [3058.6     +91.90 813.7]-RATTVOLVOLOGGAV;
16 ANDREASHUVUD1=       [3210.8     -396.6 1389.6]-RATTVOLVOLOGGAV;
17 ANDREASBROSTK1=      [3213.3     -386.9 1016.7]-RATTVOLVOLOGGAV;
18 ANDREASHUVUD2=       [3002.3     -390.8 1378.0]-RATTVOLVOLOGGAV;
19 ANDREASBROSTK2=      [3087.5     -446.7 1065.1]-RATTVOLVOLOGGAV;
20 ANDREASHUVUD3=       [3281.7     -387.3 1407.2]-RATTVOLVOLOGGAV;
21 ANDREASBROSTK3=      [3219.8     -397.0 1078.6]-RATTVOLVOLOGGAV;
22 ANDREASHUVUD4=       [2896.      -397.6 1319.3]-RATTVOLVOLOGGAV;
23 ANDREASBROSTK4=      [3066.6     -400.3 1018.2]-RATTVOLVOLOGGAV;
24 KIMHUVUD1=           [3169.1     -364.0 1374.1]-RATTVOLVOLOGGAV;
25 KIMBROST1=           [3140.6     -360.6 1080.1]-RATTVOLVOLOGGAV;
26 KIMHUVUD2=           [2936.7     -337.0 1366.9]-RATTVOLVOLOGGAV;
27 KIMBROST2=           [3032.6     -356.4 1047.0]-RATTVOLVOLOGGAV;
28 KIMHUVUD3=           [3198.1     -346.3 1382.7]-RATTVOLVOLOGGAV;
29 KIMBROST3=           [3145.9     -354.0 1096.2]-RATTVOLVOLOGGAV;
30 KIMHUVUD4=           [2830.1     -367.7 1330.6]-RATTVOLVOLOGGAV;
31 KIMBROST4=           [2973.9     -364.6 1041.1]-RATTVOLVOLOGGAV;
```

```
32
33 %4 punkter definierar ett schackbräde
34 SCHACKPKT=[+3489.3 -173.1 +1397.7
35     +3483.3      -544.3 +1408.2
36     +3322.0      -555.2  +900.2
37     +3330.5      -184.2  +889.7
38     +3469.0      -172.5 +1399.7
39     +3463.4      -543.9 +1410.6
40     +3302.8      -555.4  +902.6
41     +3310.2      -184.4  +891.7
42     +3451.3      -171.6 +1401.1
43     +3445.1      -544.0 +1412.2
44     +3287.2      -555.3  +903.2
45     +3292.3      -183.9  +893.3
46     +3431.7      -171.5 +1403.8
47     +3427.2      -543.2 +1413.5
48     +3266.3      -555.6  +906.3
49     +3273.3      -184.1  +894.9
50     +3415.5      -171.8 +1404.7
51     +3410.5      -544.2 +1414.8
52     +3251.3      -555.9  +907.4
53     +3259.2      -184.5  +895.8
54     +3400.3      -172.1 +1406.1
55     +3400.1      -543.4 +1414.9
56     +3234.1      -556.2  +909.7
57     +3241.2      -184.7  +898.0
58     +3385.8      -172.0 +1407.7
59     +3382.3      -543.0 +1416.7
60     +3221.5      -556.3  +910.3
61     +3228.0      -185.2  +899.3
62     +3370.1      -171.6 +1408.5
63     +3367.7      -543.2 +1417.9
64     +3206.0      -556.0  +911.6
65     +3211.2      -184.5  +900.8
66     +3357.8      -172.1 +1410.0
67     +3355.0      -543.0 +1420.4
68     +3192.3      -555.3  +913.2
69     +3198.9      -184.0  +901.8
70     +3342.1      -172.7 +1411.0
71     +3337.5      -543.3 +1420.8
72     +3176.7      -556.0  +914.7
73     +3184.0      -185.1  +904.2
74     +3325.4      -173.8 +1413.4
75     +3319.1      -544.6 +1422.2
76     +3158.8      -555.9  +914.7
77     +3166.8      -185.0  +905.2
78     +3305.3      -174.2 +1415.6
79     +3298.3      -543.1 +1425.1
80     +3139.6      -555.9  +917.6
81     +3147.3      -185.0  +906.7
82     +3288.6      -170.3 +1416.9
83     +3283.3      -542.3 +1428.3
```

```
    +3122.5      -556.0  +920.9
    +3128.8      -183.6  +909.3
    +3272.1      -169.9 +1418.9
    +3264.8      -541.8 +1431.6
    +3105.5      -555.9  +922.7
    +3112.2      -183.7  +910.9
    +3270.4      -170.4 +1418.4
    +3265.2      -541.2 +1430.3
    +3109.6      -183.8  +910.6
    +3103.6      -555.6  +923.3];
[M N]=size(SCHACKPKT);
%Sätter RATTVOLVOLOGGAVn i origo
for i=1:M
    SCHACKPKT(i,:)=SCHACKPKT(i,:)-RATTVOLVOLOGGAV;
end
%Sparar alla variabler i matfil
save pkt.mat
end
```

```
function [] = init_b_skiva()
%Allokerar plats
xtemp = zeros(15,221);
ytemp = zeros(15,221);
xradsteg= zeros(15,16,13);
yradsteg= zeros(15,16,13);
x=zeros(30,65,49);
y=zeros(30,65,49);
xt=zeros(15,17,13);
yt=zeros(15,17,13);
xte=zeros(15,65,13);
yte=zeros(15,65,13);
xkolumnsteg=zeros(15,65,12);
ykolumnsteg=zeros(15,65,12);
%Läser in mätvärde från bilder på schackbrädet, från en textfil
%n=antal skivor
for n=1:15
    var = load(sprintf('txtmat/image%d.txt', n));
    [s,N]=size(var);
    k=1;
    for i=1:s
        for s=1:2:N-1
            xtemp(n,k)=var(i,s);
            ytemp(n,k)=var(i,1+s);
            k=k+1;
        end
    end
end

%Delar in mätpunkterna efter s=skivindex r=rad k=kolumn
for s=1:15
    k=1;
    r=1;
    for i=1:221
```

```
        if r>(N/2)
            r=1;
            k=k+1;
        end
        xt(s,r,k)=xtemp(s,i);
        yt(s,r,k)=ytemp(s,i);
        r=r+1;

    end
end

%Räknar ut skillnaden i radavstånd mellan punkter
for s=1:15
    for k=1:13
        for r=1:16
            xradsteg(s,r,k)=(xt(s,r,k)-xt(s,r+1,k))/4;
            yradsteg(s,r,k)=(yt(s,r,k)-yt(s,r+1,k))/4;
        end
    end
end

%För att skapa tre nya punkter mellan utpekade punkter från
    schackhörn på film
for s=1:15
    for k=1:13
        l=0;
        for r=1:65
            if mod(r,4)==1
                l=l+1;
                p=0;
                xte(s,r,k)=xt(s,l,k);
                yte(s,r,k)=yt(s,l,k);
            else
                p=p+1;
                xte(s,r,k)=xt(s,l,k)-xradsteg(s,l,k)*p;
                yte(s,r,k)=yt(s,l,k)-yradsteg(s,l,k)*p;
            end
        end
    end
end

%Räknar ut skillnaden i kolumnavstånd mellan punkter
for s=1:15
    for r=1:65
        for k=1:12
            xkolumnsteg(s,r,k)=(xte(s,r,k)-xte(s,r,k+1))/4;
            ykolumnsteg(s,r,k)=(yte(s,r,k)-yte(s,r,k+1))/4;
        end
    end
end

%För att skapa tre nya punkter mellan utpekade punkter från
```

```
%schackhörn på film
for s=1:15
    for r=1:65
        n=0;
        l=0;
        for k=1:49
            if mod(k,4)==1
                n=0;
                l=l+1;
                x(s,r,k)=xte(s,r,l);
                y(s,r,k)=yte(s,r,l);
            else
                n=n+1;
                x(s,r,k)=xte(s,r,l)-xkolumnsteg(s,r,l)*n;
                y(s,r,k)=yte(s,r,l)-ykolumnsteg(s,r,l)*n;
            end
        end
    end
end

%Linjariseringsfunk för approximering av 15 skivor framåt
    beroende av 15
%föregående
for r=1:65
    for k=1:49
        steg1=0;
        for s=1:14
            steg1=x(s+1,r,k)-x(s,r,k)+steg1;
        end
        steg(r,k)=steg1/15;
    end
end

for s=1:15
    for r=1:65
        for k=1:49
            x1=x(15,r,k)+(steg(r,k)*s);
            g = regress(y(1:15,r,k),[ones(15,1,1) x(1:15,r,k)]);
            h=@(x) g(1)+g(2)*x;
            x(s+15,r,k)=x1;
            y(s+15,r,k)=h(x1);
        end
    end
end

%Plottar
%Visa bakgrundsbild? Om ja, ta bort kommentar
% clf
% b = imread('schack_rect1.jpg');
% imagesc(b);
% set(gca,'YDir','reverse');

```

```
% Plottar
% grid on;
% xlabel('X');
% ylabel('Y');
% hold on
% plot(x(1,:),y(1,:),'o','MarkerEdgeColor','y')
% plot(xt(1,:),yt(1,:),'+','MarkerEdgeColor','r')

% plot(x(15,:),y(15,:),'o','MarkerEdgeColor','r')
% plot(x(30,:),y(30,:),'o','MarkerEdgeColor','b')

savefile = 'bk_var.mat';
save(savefile, 'x','y');
clear all;
end
```

```
function [] = init_v_skiva()
%Laddar in alla uppmätningspunkter
load('pkt.mat')
%M= antal skivor N=antal mätpunkter per skiva
[M N]=size(SCHACKPKT);
%s=skiva r=rad k=kolumn
for s=0:(M/4)-1
    %Läser in 4 hörnor per skiva och kallar dessa punkt A,B,C,D
    A=SCHACKPKT(1+(s*4),1:end);
    B=SCHACKPKT(2+(s*4),1:end);
    C=SCHACKPKT(3+(s*4),1:end);
    D=SCHACKPKT(4+(s*4),1:end);
    %Mellan A och B, schackbräde horisontal-led ovan
    ABdistx=(A(1)-B(1))/48;
    ABdisty=(A(2)-B(2))/48;
    ABdistz=(A(3)-B(3))/48;
    %Mellan D och C, schackbräde horisontal-led nedan
    DCdistx=(D(1)-C(1))/48;
    DCdisty=(D(2)-C(2))/48;
    DCdistz=(D(3)-C(3))/48;
    %Medelavstånd i horizontal-led
    MHdistx=(ABdistx+DCdistx)/2;
    MHdisty=(ABdisty+DCdisty)/2;
    MHdistz=(ABdistz+DCdistz)/2;


    %A till D, vertikal-led närmast origo
    ADdistx=(A(1)-D(1))/65;
    ADdisty=(A(2)-D(2))/65;
    ADdistz=(A(3)-D(3))/65;
    %B till C, vertikal-led längst från origo
    BCdistx=(B(1)-C(1))/65;
    BCdisty=(B(2)-C(2))/65;
    BCdistz=(B(3)-C(3))/65;
    %Medelavstånd vertikal-led
    MVdistx=(ADdistx+BCdistx)/2;
    MVdisty=(ADdisty+BCdisty)/2;
```

```
38      MVdistz=(ADdistz+BCdistz)/2;
39
40      %Skapar 65x49 nya punkter (samma som bildpunkter)
41      for k=0:48
42          for r=1:65
43              x(s+1,r,k+1)=[B(1)+(ABdistx*k)-(MVdistx*r)];
44              y(s+1,r,k+1)=[B(2)+(ABdisty*k)-(MVdisty*r)];
45              z(s+1,r,k+1)=[B(3)+(ABdistz*k)-(MVdistz*r)];
46          end
47      end
48      %Ritar ut alla mätpunkter
49      %     plot3(A(1), A(2), A(3),'.')
50      %     hold on;
51      %     text(A(1), A(2), A(3), 'A', 'Color', 'b');
52      %     plot3(B(1), B(2), B(3),'.')
53      %     text(B(1), B(2), B(3), 'B', 'Color', 'b');
54      %     plot3(C(1), C(2), C(3),'.')
55      %     text(C(1), C(2), C(3), 'C', 'Color', 'b');
56      %     plot3(D(1), D(2), D(3),'.')
57      %     text(D(1), D(2), D(3), 'D', 'Color', 'b');
58      %  grid on;
59      %  xlabel('X');
60      %  ylabel('Y');
61      %  zlabel('Z');
62  end
63
64  %Avstånd mellan punkter mellan olika plan, skapar medelvärde för
        ett
65  %steg framåt
66  for r=1:65
67      for k=1:49
68          sx=0;
69          sy=0;
70          sz=0;
71          for s=1:14
72              sx=x(s+1,r,k)-x(s,r,k)+sx;
73              sy=y(s+1,r,k)-y(s,r,k)+sy;
74              sz=z(s+1,r,k)-z(s,r,k)+sz;
75          end
76          stegx(r,k)=sx/15;
77          stegy(r,k)=sy/15;
78          stegz(r,k)=sz/15;
79      end
80  end
81  X = [ones(size(x(:))) x(:) y(:) x(:).*y(:)];
82  b = regress(z(:),X);
83  f = @(x,y) b(1)+b(2)*x+b(3)*y + b(4)*x.*y;
84
85  %Använder linjär multipel regression för att skapa 15 skivor
        framåt
86  for s=1:15
87      for r=1:65
```

```
        for k=1:49
            x(15+s,r,k)=x(15,r,k)+(stegx(r,k)*s);
            y(15+s,r,k)=y(15,r,k)+(stegy(r,k)*s);
            %z(15+s,r,k)=f(x(15+s,r,k),y(15+s,r,k));
            z(15+s,r,k)=z(15,r,k)+(stegz(r,k)*s);
        end
    end
end

%Sparar och rensar värden, behöver bara köra detta skript en gång
savefile = 'vk_var.mat';
save(savefile, 'x','y','z');
clear all;
% plot3(x(1,:),y(1,:),z(1,:),'o','MarkerEdgeColor','b')
% hold on;
% grid on;
% xlabel('X');
% ylabel('Y');
% zlabel('Z');
% plot3(x(15,:),y(15,:),z(15,:),'o','MarkerEdgeColor','r')
% plot3(x(30,:),y(30,:),z(30,:),'o','MarkerEdgeColor','b')
end
```

```
function [ s ] = sruta()
load('pkt.mat')

[M N]=size(SCHACKPKT);
%Allokerar matriser
sKim=zeros(M/4,1);
sHeron=zeros((M/4),1);
sidavektor=zeros(M/4,1);

for i=0:(M/4-1)
A=SCHACKPKT(1+(i*4),1:end);
B=SCHACKPKT(2+(i*4),1:end);
C=SCHACKPKT(3+(i*4),1:end);
D=SCHACKPKT(4+(i*4),1:end);

% Kommentera bort för att se vilka rutor.
% plot3(A(1),A(2),A(3),'-x')
% hold on;
% plot3(B(1),B(2),B(3),'-x')
% plot3(C(1),C(2),C(3),'-x')
% plot3(D(1),D(2),D(3),'-x')
% text(A(1), A(2),A(3), 'A', 'Color', 'r');
% text(B(1), B(2),B(3), 'B', 'Color', 'r');
% text(C(1), C(2),C(3), 'C', 'Color', 'r');
% text(D(1), D(2),D(3), 'D', 'Color', 'r');
% grid on;
% xlabel('X');
% ylabel('Y');
% zlabel('Z');

```

```
%Avstånd
AB= sqrt((A(1)-B(1))^2 + (A(2)-B(2))^2 + (A(3)-B(3))^2);
AD= sqrt((A(1)-D(1))^2 + (A(2)-D(2))^2 + (A(3)-D(3))^2);
BC= sqrt((B(1)-C(1))^2 + (B(2)-C(2))^2 + (B(3)-C(3))^2);
CD= sqrt((C(1)-D(1))^2 + (C(2)-D(2))^2 + (C(3)-D(3))^2);

%Approx. sidor
L=(BC+AD)/2;
Bredd=(AB+CD)/2;

%Area
AKim=L*Bredd;
%sidlängd
sKim(i+1)=[sqrt(AKim/(17*12))];

%Area med Heron's triangel formula
a=AB;
b=AD;
c=BC;
d=CD;
s=(a+b+c+d)/2;
Heron=@(s,a,b,c)sqrt((s-a)*(s-b)*(s-c)*(s-d));
AHeron=Heron(s,a,b,c);
%Sida Heron
sHeron(i+1)=[sqrt(AHeron/(17*12))];
%Area Vektor
Vektora=@(V0,V1,V3)abs(cross((V1-V0),(V3-V0)));
aVektor=Vektora(A,B,C);
areavektor=sqrt((aVektor(1)^2+aVektor(2)^2+aVektor(3)^2));
sidavektor(i+1)=[sqrt(areavektor/(17*12))];
end

sKim;
sHeron;
sidavektor;
medelsKim=0;
medelsHeron=0;
medelsvektor=0;
for i=1:size(sKim,1)
medelsKim=sKim(i)+medelsKim;
medelsHeron=medelsHeron+sHeron(i);
medelsvektor=medelsvektor+sidavektor(i);
end
medelsKim=medelsKim/size(sKim,1);
medelsHeron=medelsHeron/size(sHeron,1);
medelsvektor=medelsvektor/size(sidavektor,1);

% Antar att sidlängd av schackkvadrater är 31,1355 då sista tal i
     sKim och sHeron är avikande
s=medelsvektor;
end
```

```
function varargout= est_gui(varargin)
```

```
% est_gui MATLAB code for est_gui.fig
%      est_gui, by itself, creates a new est_gui or raises the existing
%      singleton*.
%
%      H = est_gui returns the handle to a new est_gui or the handle to
%      the existing singleton*.
%
%      est_gui('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in est_gui.M with the given input arguments.
%
%      est_gui('Property','Value',...) creates a new est_gui or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before est_gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to est_gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help est_gui

% Last Modified by GUIDE v2.5 29-Apr-2013 11:23:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @est_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @est_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
44 % End initialization code - DO NOT EDIT
45
46
47 % --- Executes just before est_gui is made visible.
48 function est_gui_OpeningFcn(hObject, eventdata, handles, varargin
       )
49 % This function has no output args, see OutputFcn.
50 % hObject    handle to figure
51 % eventdata  reserved - to be defined in a future version of
       MATLAB
52 % handles    structure with handles and user data (see GUIDATA)
53 % varargin   command line arguments to est_gui (see VARARGIN)
54
55 % Choose default command line output for est_gui
56 handles.output = hObject;
57
58 % Update handles structure
59 guidata(hObject, handles);
60
61 % UIWAIT makes est_gui wait for user response (see UIRESUME)
62 % uiwait(handles.figure1);
63
64
65 % --- Outputs from this function are returned to the command line
       .
66 function varargout = est_gui_OutputFcn(hObject, eventdata,
       handles)
67 % varargout  cell array for returning output args (see VARARGOUT)
       ;
68 % hObject    handle to figure
69 % eventdata  reserved - to be defined in a future version of
       MATLAB
70 % handles    structure with handles and user data (see GUIDATA)
71 % Get default command line output from handles structure
72 varargout{1} = handles.output;
73
74
75 % --- Executes on button press in pushbutton1.
76 function pushbutton1_Callback(hObject, eventdata, handles)
77 % hObject    handle to pushbutton1 (see GCBO)
78 % eventdata  reserved - to be defined in a future version of
       MATLAB
79 % handles    structure with handles and user data (see GUIDATA)
80 stang=1; % Stänger GUIn
81 save('stang')
82 close(gcf)
83
84 % --- Executes on button press in pushbutton3.
85 function pushbutton3_Callback(hObject, eventdata, handles)
86
87 % hObject    handle to pushbutton3 (see GCBO)
```

```
88 % eventdata  reserved - to be defined in a future version of
       MATLAB
89 % handles    structure with handles and user data (see GUIDATA)
90 stang = 0;
91 save('stang');
92 prompt = {'Antal bilder:'};
93 dlg_title = 'Hur många bilder vill du använda?';
94 num_lines = 1;
95 def = {'2'};
96 a = inputdlg(prompt,dlg_title,num_lines,def); %Inputdialog
97  nr_of_pics = cellfun(@(x)str2double(x), a);
98  if (nr_of_pics == 2 || nr_of_pics == 4) %Hur många bilder?
99     im = setpics(nr_of_pics);
100  else
101      while(true)
102          s = warndlg(sprintf('Du kan endast använda 2 eller 4
                bilder!'));
103          waitfor(s);
104          a = inputdlg(prompt,dlg_title,num_lines,def);
105          nr_of_pics = cellfun(@(x)str2double(x), a);
106          if (nr_of_pics == 2 || nr_of_pics == 4)
107              break
108          end
109      end
110    im = setpics(nr_of_pics);
111  end
112
113  save('antal_bilder.mat', 'nr_of_pics','im'); %Sparar val
114  close(gcf);
115
116
117
118 % --- If Enable == 'on', executes on mouse press in 5 pixel
       border.
119 % --- Otherwise, executes on mouse press in 5 pixel border or
       over pushbutton3.
120 function pushbutton3_ButtonDownFcn(hObject, eventdata, handles)
121
122 % hObject    handle to pushbutton3 (see GCBO)
123 % eventdata  reserved - to be defined in a future version of
       MATLAB
124 % handles    structure with handles and user data (see GUIDATA)
125
126
127 % --- If Enable == 'on', executes on mouse press in 5 pixel
       border.
128 % --- Otherwise, executes on mouse press in 5 pixel border or
       over pushbutton1.
129 function pushbutton1_ButtonDownFcn(hObject, eventdata, handles)
130 % hObject    handle to pushbutton1 (see GCBO)
131 % eventdata  reserved - to be defined in a future version of
       MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on key press with focus on pushbutton1 and none of its controls.
function pushbutton1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%       Key: name of the key that was pressed, in lower case
%       Character: character interpretation of the key(s) that was pressed
%       Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
```

```
% Välj bilder
function [im] = setpics(nr_of_pics)

for i=1:1:nr_of_pics
[filename, pathname] = uigetfile({'*.jpg';'*.bmp';'*.*'},'Open File');
im{i} = imread(filename);
end
end
```

```
function [ax ay bx by] = valjpkt(im,hurmangabilder)
%Skickar med bilder och antalet bilder. Användaren peckar ut var
%objektets ändpunkter befinner sig, filter för höjd intensitet och mer
%konturer och sparar koordinater
%ax=ändpunkts x koordinater
%ay=ändpunkts y koordinater
%bx=andra ändpunkts x koordinater
%by=andra ändpunkts y koordinater
if hurmangabilder==2
    imshow(filterbild(im{1}));
    [ax(1),ay(1)]=ginput(1);
    hold on;
    imshow(filterbild(im{1}));
    [bx(1),by(1)]=ginput(1);
    imshow(filterbild(im{2}));
    [ax(2),ay(2)]=ginput(1);
    imshow(filterbild(im{2}));
    [bx(2),by(2)]=ginput(1);
    plot([ax(1) bx(1)],[ax(2) bx(2)],'color','g')
    plot([ay(1) by(1)],[ay(2) by(2)],'color','g')
elseif hurmangabilder==4
    imshow(filterbild(im{1}));
    [ax(1),ay(1)]=ginput(1);
    hold on;
    imshow(filterbild(im{2}));
    [bx(1),by(1)]=ginput(1);
    imshow(filterbild(im{3}));
```

```
    [ax(2),ay(2)]=ginput(1);
    imshow(filterbild(im{4}));
    [bx(2),by(2)]=ginput(1);
    plot([ax(1) bx(1)],[ay(1) by(1)],'color','g')
    plot([ax(2) bx(2)],[ay(2) by(2)],'color','g')
end
%Ritar ut
plot(ax(1),ay(1),'x', 'MarkerEdgeColor','r','MarkerSize',10)
plot(bx(1),by(1),'x', 'MarkerEdgeColor','r','MarkerSize',10)
plot(ax(2),ay(2),'+', 'MarkerEdgeColor','b','MarkerSize',10)
plot(bx(2),by(2),'+', 'MarkerEdgeColor','b','MarkerSize',10)
end
```

```
function [im] = filterbild(f)
%  g=imread('kim1ansikt_rect.jpg');
  g=f(1:end,1:end,1);
%  f = wiener2(f,[5 5]);
 im=histeq(g);
 end
```

```
function [arad1,akolumn1,brad1,bkolumn1,arad2,akolumn2,brad2,
    bkolumn2,mskiva,nskiva] = bkoordfinder(ax,ay,bx,by,antalrutor
    )
load('bk_var.mat')
%Tittar på varje skiva, vilka mätpunkter/linjariserade punkter
    som är
%närmast objektets ändar.
%30 skivor 65 rader 49 kolumner
size(x)
for m=1:30
    for o=1:65
        for k=1:49
            PUNKTAVSTAND1(o,k)=avstand(ax(1),x(m,o,k),ay(1),y(m,o
                ,k),0,0);
            PUNKTAVSTAND2(o,k)=avstand(bx(1),x(m,o,k),by(1),y(m,o
                ,k),0,0);
            PUNKTAVSTAND3(o,k)=avstand(ax(2),x(m,o,k),ay(2),y(m,o
                ,k),0,0);
            PUNKTAVSTAND4(o,k)=avstand(bx(2),x(m,o,k),by(2),y(m,o
                ,k),0,0);
        end
    end
    %Hittar punkten som gav minst avstånd
    %arad1=första bilden första punkten
    %arad2=andra bilden första punkten
    %brad1=första bilden andra punkten
    %brad2=andra bilden andra punkten
    %Analogt med rader och kolumner
    [arad1(m) akolumn1(m)]=find(PUNKTAVSTAND1==min(min(
        PUNKTAVSTAND1)));
    [brad1(m) bkolumn1(m)]=find(PUNKTAVSTAND2==min(min(
        PUNKTAVSTAND2)));
```

```
24      [arad2(m) akolumn2(m)]=find(PUNKTAVSTAND3==min(min(
            PUNKTAVSTAND3)));
25      [brad2(m) bkolumn2(m)]=find(PUNKTAVSTAND4==min(min(
            PUNKTAVSTAND4)));
26      %Tar reda på katet längd mellan objektets ändar för bredden
27      bredkatet1=abs(arad1(m)-brad1(m));
28      bredkatet2=abs(arad2(m)-brad2(m));
29      %Tar reda på katet längd mellan objektets ändar för höjden
30      hojdkatet1=abs(akolumn1(m)-bkolumn1(m));
31      hojdkatet2=abs(akolumn2(m)-bkolumn2(m));
32      %Pythagoras sats för längden mellan punkterna
33      hyp1(m)=sqrt(bredkatet1^2+hojdkatet1^2);
34      hyp2(m)=sqrt(bredkatet2^2+hojdkatet2^2);
35  end
36  %Avståndsskillnad från varje skiva mellan objektets ändpunkter
37  %subtraherat med antalet rutor som täcker samma avstånd
38  for m=1:30
39      avstandsdiff1(m)=abs(hyp1(m)-(antalrutor(1)*4));
40      avstandsdiff2(m)=abs(hyp2(m)-(antalrutor(2)*4));
41  end
42  %Hittar vilka plan som gav minsta avståndsskillnaden från alla
        skivorna
43  %och skickar till nästa funktion
44  mskiva=find(avstandsdiff1==min(min(avstandsdiff1)));
45  nskiva=find(avstandsdiff2==min(min(avstandsdiff2)));
46
47  %Plottar
48  %Visa att algoritm funkar, ritar ut bild, utpekade punkter och
        punkter som
49  %hittats
50  % hold on;
51  % set(gca,'YDir','reverse');
52  % b = imread('kimA1_2_rect.jpg');
53  % imagesc(b);
54  % plot(x(mskiva(end),arad1(mskiva(end)),akolumn1(mskiva(end))),y(
        mskiva(end),arad1(mskiva(end)),akolumn1(mskiva(end))),'b+')
55  % plot(x(mskiva(end),brad1(mskiva(end)),bkolumn1(mskiva(end))),y(
        mskiva(end),brad1(mskiva(end)),bkolumn1(mskiva(end))),'b+')
56  % plot(ax(1),ay(1),'-o','MarkerEdgeColor','b')
57  % plot(bx(1),by(1),'-o','MarkerEdgeColor','b')
58  % plot(x(mskiva(end),arad2(mskiva(end)),akolumn2(mskiva(end))),y(
        mskiva(end),arad2(mskiva(end)),akolumn2(mskiva(end))),'r+')
59  % plot(x(mskiva(end),brad2(mskiva(end)),bkolumn2(mskiva(end))),y(
        mskiva(end),brad2(mskiva(end)),bkolumn2(mskiva(end))),'r+')
60  % plot(ax(2),ay(2),'-o','MarkerEdgeColor','r')
61  % plot(bx(2),by(2),'-o','MarkerEdgeColor','r')
62
63  % Plottar ut 2D punkter som används för att relatera mellan 2D
        till 3D
64  % for m=[11 19] %skiva
65  %  for o=1:60    %rad
66  %      for k=1:49 %kolumn
```

```
%          hold on
%          plot(x(m,o,k),y(m,o,k),'-o','MarkerEdgeColor','b')
%      end
%  end
% end
end
```

```
function [a] = avstand(x1,x2,y1,y2,z1,z2)
%Räknar ut euklidiskt avstånd där n=3
a=sqrt((x1-x2)^2+(y1-y2)^2+(z1-z2)^2);
end
```

```
function [Akoord1,Akoord2,Bkoord1,Bkoord2] = vkoordfinder(arad1,
    akolumn1,brad1,bkolumn1,arad2,akolumn2,brad2,bkolumn2,rwdist,
    mskiva,nskiva)
load('vk_var.mat');
%Om fler än ett plan är relevant, kollar minsta avståndet i
    verkligheten
for m=mskiva(1):mskiva(end)
    for n=mskiva(1):mskiva(end)
        avstandbild1(n,m)=abs(avstand(x(n,arad1(n),akolumn1(n)),x
            (m,brad1(m),bkolumn1(m)),y(n,arad1(n),akolumn1(n)),y(
            m,brad1(m),bkolumn1(m)),z(n,arad1(n),akolumn1(n)),z(m
            ,brad1(m),bkolumn1(m)))-rwdist(1));
    end
end
%Som ovan
for m=nskiva(1):nskiva(end)
    for n=nskiva(1):nskiva(end)
        avstandbild2(n,m)=abs(avstand(x(n,arad2(n),akolumn2(n)),x
            (m,brad2(m),bkolumn2(m)),y(n,arad2(n),akolumn2(n)),y(
            m,brad2(m),bkolumn2(m)),z(n,arad2(n),akolumn1(n)),z(m
            ,brad2(m),bkolumn1(m)))-rwdist(2));
    end
end
%Om värdet 0 finns, blir det oändligheten, för att finna minsta
    värdet
avstandbild1(avstandbild1==0) = Inf;
avstandbild2(avstandbild2==0) = Inf;
%Hittar bästa skiva
[n1 m1]=find(avstandbild1==min(min(avstandbild1)));
[n2 m2]=find(avstandbild2==min(min(avstandbild2)));
%Skickar vidare koordinater
Akoord1=[x(n1,arad1(n1),akolumn1(n1)) y(n,arad1(n1),akolumn1(n1))
     z(n1,arad1(n1),akolumn1(n1))];
Bkoord1=[x(m1,brad1(m1),bkolumn1(m1)) y(m1,brad1(m1),bkolumn1(m1)
    ) z(m1,brad1(m1),bkolumn1(m1))];
Akoord2=[x(n2,arad2(n2),akolumn2(n2)) y(n2,arad2(n2),akolumn2(n2)
    ) z(n2,arad2(n2),akolumn2(n2))];
Bkoord2=[x(m2,brad2(m2),bkolumn2(m2)) y(m2,brad2(m2),bkolumn2(m2)
    ) z(m2,brad1(m2),bkolumn1(m2))];

```

```
%Plottar ut punkter som var mest lämpade
% plot3(x(n1,arad1(n1),akolumn1(n1)),y(n,arad1(n1),akolumn1(n1)),
    z(n1,arad1(n1),akolumn1(n1)),'+r')
% for m1=[12]
% plot3(x(m1,brad1(m1),bkolumn1(m1)),y(m1,brad1(m1),bkolumn1(m1))
    ,z(m1,brad1(m1),bkolumn1(m1)),'+r')
% ends
% plot3(x(n2,arad2(n2),akolumn2(n2)),y(n2,arad2(n2),akolumn2(n2))
    ,z(n2,arad2(n2),akolumn2(n2)),'+r')
% for m2=[12]
% plot3(x(m2,brad2(m2),bkolumn2(m2)),y(m2,brad2(m2),bkolumn2(m2))
    ,z(m2,brad1(m2),bkolumn1(m2)),'+r')
% end

% Plottar ut m=skiva o=rad k=kolumn
% for m=[18]
%     for o=1:65
%         for k=1:49
%             plot3(x(m,o,k),y(m,o,k),z(m,o,k),'-o');
%         end
%     end
% end
% xlabel('X');
% ylabel('Y');
% zlabel('Z');
end
```

```
function [] = felupp(Akoord1,Akoord2,Bkoord1,Bkoord2,koord1,
    koord2,koord3,koord4)
k=0;
%Räknar ut fel och returnerar i procent i x y z
for i=['x' 'y' 'z']
    k=k+1;
    fel(koord1(k),Akoord1(k),i,1)
end
%Skriver och räknar ut fel och avstånd för första punkten i
    första bilden
feltillkamera=@(koord)sqrt((koord(1)-0)^2+(koord(2)-0)^2+(koord
    (3)-0)^2);
felavstand=abs((feltillkamera(koord1)-feltillkamera(Akoord1))/
    feltillkamera(koord1))*100;
felavstandmm=abs(feltillkamera(koord1)-feltillkamera(Akoord1));
avstandetv=avstand(Akoord1(1),0,Akoord1(2),0,Akoord1(3),0);
avstandetb=avstand(koord1(1),0,koord1(2),0,koord1(3),0);
fprintf('Totala felet från ratten till objekt är %f %%\n',
    felavstand);
fprintf('Totala felet från ratten till objekt är %f mm\n',
    felavstandmm);
fprintf('Avstånd från ratten till punkten är %f mm\n',avstandetv)
    ;
fprintf('Avstånd från ratten till punkten enligt programmet är %f
    mm\n',avstandetb);
fprintf('\n')
```

```
k=0;
%Andra punkten första bilden
for i=['x' 'y' 'z']
    k=k+1;
    fel(koord3(k),Bkoord1(k),i,2)
end
felavstand=abs((feltillkamera(koord3)-feltillkamera(Bkoord1))/
    feltillkamera(koord3))*100;
felavstandmm=abs(feltillkamera(koord3)-feltillkamera(Bkoord1));
avstandetv=avstand(Bkoord1(1),0,Bkoord1(2),0,Bkoord1(3),0);
avstandetb=avstand(koord3(1),0,koord3(2),0,koord3(3),0);
fprintf('Totala felet från ratten till objekt är %f %%\n',
    felavstand);
fprintf('Totala felet från ratten till objekt är %f mm\n',
    felavstandmm);
fprintf('Avstånd från ratten till punkten är %f mm\n',avstandetv)
    ;
fprintf('Avstånd från ratten till punkten enligt programmet är %f
     mm\n',avstandetb);
fprintf('\n')
%Första punkten andra bilden
k=0;
for i=['x' 'y' 'z']
    k=k+1;
    fel(koord2(k),Akoord2(k),i,3)
end
felavstand=abs((feltillkamera(koord2)-feltillkamera(Akoord2))/
    feltillkamera(koord2))*100;
felavstandmm=abs(feltillkamera(koord2)-feltillkamera(Akoord2));
avstandetv=avstand(Akoord2(1),0,Akoord2(2),0,Akoord2(3),0);
avstandetb=avstand(koord2(1),0,koord2(2),0,koord2(3),0);
fprintf('Totala felet från ratten till objekt är %f %%\n',
    felavstand);
fprintf('Totala felet från ratten till objekt är %f mm\n',
    felavstandmm);
fprintf('Avstånd från ratten till punkten är %f mm\n',avstandetv)
    ;
fprintf('Avstånd från ratten till punkten enligt programmet är %f
     mm\n',avstandetb);
fprintf('\n')
%Andra punkten andra bilden
k=0;
for i=['x' 'y' 'z']
    k=k+1;
    fel(koord4(k),Bkoord2(k),i,4)
end
felavstand=abs((feltillkamera(koord4)-feltillkamera(Bkoord2))/
    feltillkamera(koord4))*100;
felavstandmm=abs(feltillkamera(koord4)-feltillkamera(Bkoord2));
avstandetv=avstand(Bkoord2(1),0,Bkoord2(2),0,Bkoord2(3),0);
avstandetb=avstand(koord4(1),0,koord4(2),0,koord4(3),0);
```

```
59 fprintf('Totala felet från ratten till objekt är %f %%\n',
       felavstand);
60 fprintf('Totala felet från ratten till objekt är %f mm\n',
       felavstandmm);
61 fprintf('Avstånd från ratten till punkten är %f mm\n',avstandetv)
       ;
62 fprintf('Avstånd från ratten till punkten enligt programmet är %f
       mm\n',avstandetb);
63
64 end
```

```
1  function [] = fel(PUNKT,koord,x,val)
2  %Räknar ut felet i procent och millimeter;
3  FELp=abs((PUNKT-koord)/PUNKT)*100;
4  FELmm=abs(PUNKT-koord);
5  if val==1
6      fprintf('Fel från %skoord på första punkten i bild ett är %f
           %%\n',x,FELp)
7      fprintf('Fel från %skoord på första punkten i bild ett är %f
           mm\n',x,FELmm)
8  elseif val==2
9      fprintf('Fel från %skoord på andra punkten i bild ett är %f
           %%\n',x,FELp)
10     fprintf('Fel från %skoord på andra punkten i bild ett är %f
           mm\n',x,FELmm)
11 elseif val==3
12     fprintf('Fel från %skoord på tredje punkten i bild två är %f
           %%\n',x,FELp)
13     fprintf('Fel från %skoord på tredje punkten i bild två är %f
           mm\n',x,FELmm)
14 elseif val==4
15     fprintf('Fel från %skoord på fjärde punkten i bild två är %f
           %%\n',x,FELp)
16     fprintf('Fel från %skoord på fjärde punkten i bild två är %f
           mm\n',x,FELmm)
17 end
18 end
```