Industrial Electrical Engineering Automation

Utveckling av 3D-Visionsystem

Med Robot och PLC-kommunikation

Edvin Fischer
Simon Lindell

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Sammanfattning

Projektet gick ut på att undersöka 3D-Vision med hjälp av 3D-visionkameran Cognex 3D-A5120 och tillhörande programvara Cognex Designer. Målet var att implementera ett 3D-visionsystem som extraherar data från ett eller flera objekt avbildat av kameran. Data som extraheras består av centrumkoordinater och orientering av objektets hittade yta, i förhållande till ett fixt koordinatsystem. Det implementerade programmet skulle ha samma funktionalitet som ett 2D-visionsystem samt kunna avgöra höjdskillnader och läsa av funna objekts orientering. En robot skulle sedan implementeras för att plocka upp och flytta de objekt som hittades av kameran, utifrån datan som extraheras med Cognex Designer. Arbetet resulterade i ett automatiserat system som kunde hitta och plocka upp intränade lådor i form av rätblock. Dessa lådor placerades sedan i ett led på en avsedd plats.

Systemet var baserat på olika kommunikationssekvenser där en PLC agerade som mellanhand mellan Cognex Designer och en ABB-robot.

Nyckelord: Cognex Designer, vision, PLC, koordinater, vinklar.

Abstract

The project was about studying 3D-vision using the 3D-vision camera Cognex 3D-A5120 along with associated software program Cognex Designer. The goal was to implement a 3D-vision system with the functionality to extract data in the form of center coordinates and orientation of an object's found surface, relative to a fixtured coordinate space.
The implemented 3D-vision system was meant to possess the same functionality as existing 2D-systems, with the main difference of being able to recognise height differences and being able to distinguish angles on the found objects. A robot was thereafter implemented to pick up and move the objects found by the camera. The pick information used by the robot was the coordinates and angles found by the centrepoint of the object's found surface in the image by Cognex Designer.

The project resulted in a system being able to find and pick pre trained boxes, these boxes was thereafter lined up by the robot in a predetermined area.
The system was based on trigger sequences handled by a PLC, the PLC acted as a communication link between the Visionsystem and ABB-robot.

Nyckelords: Cognex Designer, vision, koordinater, PLC, vinklar.

Förord

Examensarbetet beskrivet i denna rapport har varit en avslutande del i högskoleutbildningen för oss studenter som omfattar 22.5 högskolepoäng.

Vi vill tacka Automationsteknik AB för assistans och möjligheten att utföra detta arbete trots de omständigheter som rått med COVID-19. Vi vill tacka Håkan Marke och Eric Karlsson lite extra för hjälp och handledning under projektets gång.

Vi vill även tacka Eyetech AB tillgodoseende av Cognex 3D-A5120 kameran och 3D-licensen tillhörande Cognex Designer. Extra tack till Jim Ekström för hjälp och stöd vid frågor gällande Cognex Designer.

Tack till både vår handledare Mats Lilja för stöd och hjälp under projektets gång, samt Morten Hemmingsson för att ha tagit rollen som vår examinator.

Lund, november 2020

Edvin Fischer & Simon Lindell

Innehållsförteckning

1 Inledning

Visionsystem är teknologi som möjliggör att information extraheras från bilder och används som underlag för att automatiska system ska kunna interagera med omvärlden. Data som extraheras från bilden kan vara i form av referenspunkter och koordinatsystem, vilket kan användas vid till exempel kvalitetskontroll, processkontroll och för robotstyrning.

1.1 Bakgrund

Detta examensarbete görs i samarbete med Automationsteknik AB som är lokaliserat i Hässleholm. Automationsteknik startade 1994 och ingår sedan 1997 i Levinsgruppen. Verksamheten, som huvudsakligen sker från deras huvudkontor i Hässleholm, består av allt från tekniska konsulttjänster till "Turn-Key" lösningar med huvudområdena inom process-, trä-, verkstads- och livsmedelsindustrin. Automationsteknik sysslar med allt från högnivåspråk, till PLC-programmering, till elkonstruktion och byggnad av kompletta produktionsutrustningar.

Automationsteknik Hässleholm AB kan för tillfället erbjuda system med robot som plockar upp och förflyttar objekt, från exempelvis lastpallar, med hjälp av en 2D-visionkamera. Eftersom det nuvarande systemet endast kan detektera position på objekt i två dimensioner, det vill säga bredd och längd, så betyder det att föremålen som plockas upp måste ha förutbestämd höjd för att roboten ska kunna hantera objektet.

För att åtgärda problemet så vill Automationsteknik AB därför undersöka en annan form av liknande system som även kan urskilja höjd, vilket är ett 3D-visionsystem. En avbildning från en 3D-visionkamera uppfattar alla tre parametrar det vill säga bredd, längd och höjd. För att använda 3D-visionkamera behövs ett helt nytt system för att hantera och processa indata från kameran, vilket ska implementeras med hjälp av den nya programvaran Cognex Designer som företaget inte utforskat tidigare. Med hjälp av en enkel sekvens från ett PLC-styrsystem så ska Vision triggas så att centrumkoordinater och orientering av objektets yta avläses, i ett fixt koordinatsystem. Värdena ska sedan bearbetas av PLC:n som i sin tur ska bidra med

instruktioner till roboten för hur den ska agera. Målet är att roboten ska kunna detektera och positionera olika objekt i form av rätblock oavsett mått och hur mycket objektet är lutat. Detta med hjälp av en vakuumsugpropp som robotarmen är försedd med.

1.2 Syfte

Syftet med examensarbetet är att implementera ett visionsystem som extraherar data som centrumkoordinater och orientering av objektets yta, i ett fixt koordinatsystem, från en avbildning av en 3D-visionkamera. Informationen ska användas genom att få en robot att plocka upp olika objekt med varierande höjd, position och lutning på föremålet. Det förväntade resultatet är att bidra till att kunna ersätta mänsklig arbetskraft med ett fullt automatiserat system och därmed dra ned på kostnader för företag som väljer att implementera systemet, samt att erbjuda ett system med hög funktionalitet.

1.3 Målformulering

Målet med examensarbete är att börja utveckla ett 3D-visionsystem som tar in data från 3D-visionkameran Cognex 3D-A5120 där de tagna avbildningarna behandlas i programvaran Cognex Designer. 3D-visionsystemet ska ha samma grundfunktionalitet som det 2D-visionsystem som redan är tillgängligt men det ska även kunna urskilja höjd och lutning på föremålet. Med hjälp av instruktioner från ett PLC-styrsystemet Siemens S7-1512SP så ska en robot med vakuumsugkopp programmeras till att automatiskt kunna lokalisera objekt, på en förutbestämd arbetsyta, och plocka upp dem i dess centrumkoordinat. Om höjden varierar över objektet, det vill säga om det är lutat, så ska roboten anpassa sig efter uträknade vinklar. Utifrån indata så ska robotarmen sedan också placera objekten på avsedd plats med millimeterprecision. Det som ska undersökas är:

1. Objekt som delvis ligger på varandra (det vill säga att ena objektet lutar mot ett annat så den får sned vinkel).
2. Flera objekt med olika höjder i en och samma bild.
3. Objekt som ligger tätt intill varandra.
4. Objekt staplade ovanpå varandra.

1.4 Problemformulering

1. Hur används programvaran Cognex Designer för att få ut rätt parametrar från bilderna som blir tagna av 3D-kameran?

2. Hur överförs koordinater och annan information till PLC styrsystemet och hur kommunicerar det i sin tur med roboten?

3. Hur hanteras objekt som har inte har samma höjd över hela föremålet?

4. Hur urskiljs objekt som är staplade exakt på varandra?

5. Hur hanteras objekt som ligger delvis på varandra?

6. Hur urskiljs objekt som ligger tätt intill varandra, så det inte uppfattas som ett stort objekt?

1.5 Motivering av examensarbete

Målet med examensarbetet var att hitta ett projekt där kunskaper från både elektroteknik och datateknik kunde appliceras. Området som projektet riktade in sig på bestämdes därmed till att vara inom styr- och reglerteknik med inslag av automation. Detta fastställdes eftersom det var kurser som båda utbildningarna har gemensamt och var ämnesområden som var utav intresse.

Examensarbetet på Automationsteknik AB valdes för att det de ville undersöka låg inom det önskade området och som dessutom lät väldigt intressant. I och med att 3D-visionsystem var outforskat territorium för företaget så var uppfattningen att projektet var av stor betydelse för dem, vilket var en av anledningarna till att det var lockande. Uppgifterna var variera under projektets gång där många olika tester, analyser och liknande utfördes. Att lösa problem var ett stort ansvarsområde eftersom projektet medförde fria tyglar. Detta gav i sin tur en god inblick hur det är att arbeta som ingenjör på Automationsteknik AB.

1.6 Avgränsningar

I projektet är objekten som används avgränsade till lådor med formen av ett rätblock. Roboten ska kunna urskilja staplade objekt på varandra, men inte kunna sortera bland olika föremål i en behållare (t.ex. en låda). Egenskaper som färg ska inte heller ha någon betydelse när roboten sorterar. Vissa andra funktioner visionkameran har, som till exempel kvalitetskontroll och identifiering av objekt med hjälp av streckkoder, kommer inte att utnyttjas. Utan huvuduppgiften är att detektera och förflytta objekt.

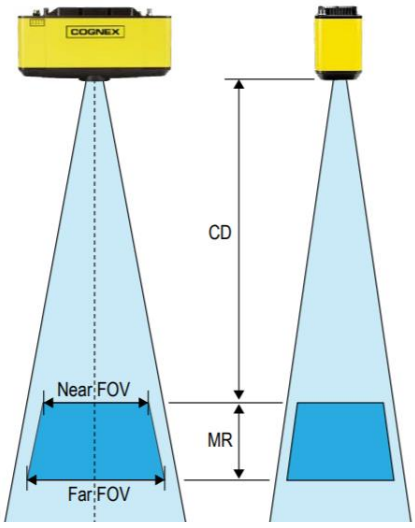
Systemet är utvecklat för att endast fungera med Cognex 3D-A5120 kamera med styrsystemet Siemens S7-1512SP och industriroboten IRB 1200 från ABB som Automationsteknik AB tillhandahåller. Det maximala måttet på objekt som roboten ska hantera begränsas av kamerans specifikationer och robotens räckvidd. Roboten kommer endast programmeras med eget arbete i mån av tid, då fokuset ska ligga på 3D-Vision i samband med styrsystemet.

2 Teknisk bakgrund

2.1 Kameran

3D-kameran Cognex 3D-A5120 är en områdesskannande kamera designad för att ta högupplösta 3D-bilder på över 1.5 miljoner datapunkter på kort tid. A5120 är den modell som har längst räckvidd och som kan skanna störst ytor, av vad Cognex kan erbjuda. Kameran har en XY-upplösningen på 506–1701 µm och en Z-upplösning 265–3246 µm. Bilden blir mer detaljrikt ju lägre värde på XY-upplösning och Z-upplösning. [5]

Denna modell har en så kallad Clearance Distance (CD) på 800 mm. Detta innebär att 800 mm är det minsta avståndet som krävs för att få en analyserbar avbildning mellan kamera och objekt. Vid optimal kameraplacering så är den maximala höjden på objekt 2000 mm som kameran kan hantera, detta beskrivs som kamerans Measurement Range (MR). Genom att addera de två beskrivna längdmåtten, CD och MR, så fås kamerans maximala räckvidd, vilket blir 2800 mm. Området som kameran kan skanna beror på var objektet befinner sig mellan arbetsområdets gränser, som konstaterades till vara mellan 800 - 2800 mm. Vid kamerans CD kan den avbilda ett område på 720 x 540 mm, detta kan refereras som dess Near FOV (Field Of View). Vid det maximala avståndet 2800 mm så kan kameran avbilda ett område på 2448 x 1836 mm, som kallas dess Far FOV. Figuren nedan visualiserar detta. [5]



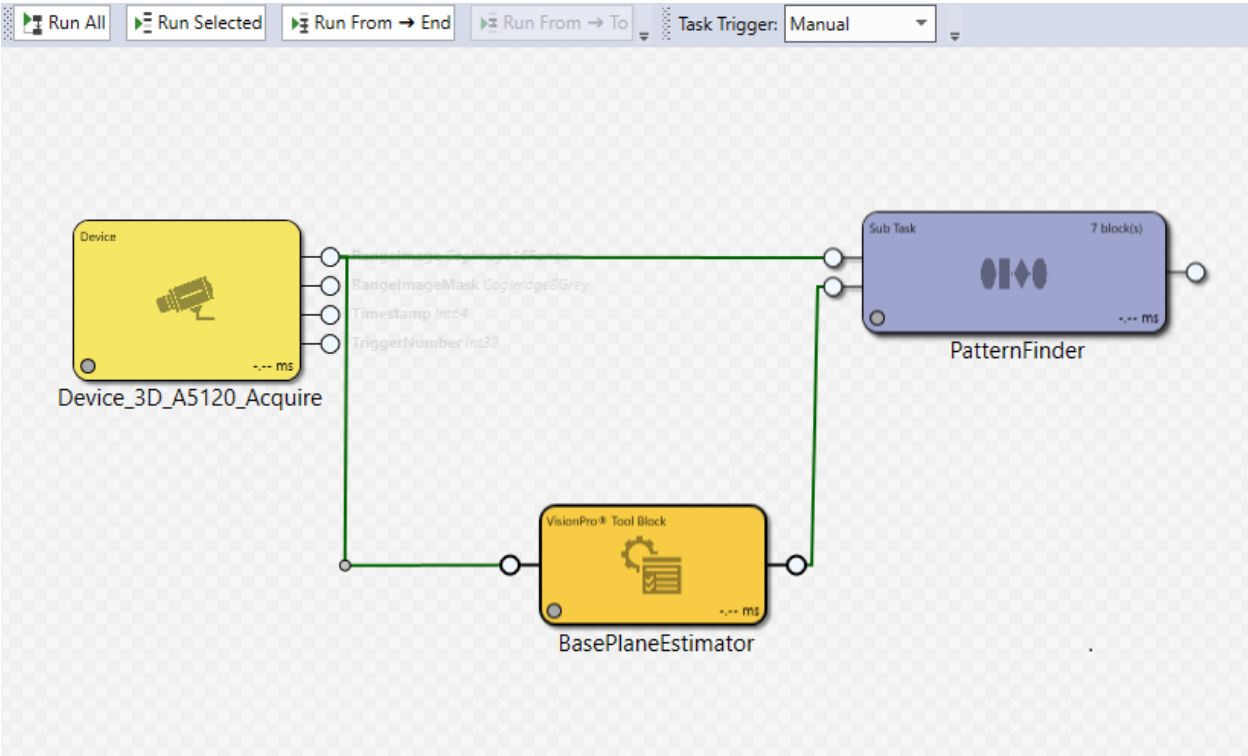
Figur 1: Cognex 3D-A5120 kamerans arbetsområde.

2.2 Vision Mjukvara

I detta avsnitt så beskrivs all vision mjukvara från Cognex som används i projektet.

2.2.1 Cognex Designer

Programvaran som gör det möjligt att analysera avbildningarna från kameran heter Cognex Designer. Cognex Designer är en grafisk, flödesschema-baserad utvecklingsmiljö som med hanterbara Tasks och scripting i språket C# möjliggör både 2D och 3D applikationer. Mjukvarans basidé ligger i dess dra-och-släppfunktionalitet. Från en sidomeny kan olika komponenter dras och släppas ner på en tom sida som i programmet definieras som en Task. De neddragna komponenterna på sidan är utformade som block och innehåller in -och utgångar som kan länkas ihop med andra block för att bilda ett händelseförlopp vid exekvering. Ett Cognex Designer projekt kan ha flera Tasks som exekveras parallellt med varandra, men inuti en Task så exekveras blocken sekventiellt från vänster till höger. Ordningen som blocken körs kan påverkas med användning av vissa komponenter som till exempel Parallell Region komponenten. Följande figur illustrerar hur en Task kan se ut. [4]



Figur 2: Bild över den implementerade tasken

Det är även möjligt att scripta i Cognex Designer. Detta görs i det objektorienterade högnivåspråket C# som är snarlikt Java [3].

2.2.2 VisionPro

Processeringen av bilderna och vision-algoritmerna sker i programvaran VisionPro som är integrerad i Cognex Designer. Med VisionPro så kan användaren utföra ett antal funktioner till följd av både 2D- och 3D-verktyg. Genom att kombinera och konfigurera verktyg så kan VisionPro leverera resultat för automatiserade applikationer i realtid där streckkodsläsning, mätning, objektidentifiering - och inspektering är exempel på användningsområden. Varje verktyg har ett specifikt gränssnitt som är unikt för just det verktyget, där man bland annat kan justera körparametrar, och visualisera resultatet. En funktion som är tillgänglig i VisionPro är möjligheten att kunna skriva anpassad kod till verktyg i en inbyggd utvecklingsmiljö, där kodningen kan göras i de objektorienterade språken C#, C++ eller VB.net. Det är i denna utvecklingsmiljö alla verktyg kan manipuleras, med allt från att bestämma när ett särskilt verktyg ska köras, till att ändra både verktygsparametrar från gränssnittet och andra mer avancerade parametrar som är dolda i gränssnittet. [3]

2.2.3 Cognex A5000 Viewer

A5000 Viewer är en programvara där 3D-A5000 sensorernas egenskaper kan justeras för att få de optimerade bilderna i miljön som kameran är placerad. A5000 Viewer stöder ett grafiskt gränssnitt som visar avbildningen i punktmolnet (mer om punktmolnet i avsnitt 2.3.1). Exempel på egenskaper som kan justeras är hur många bilder kameran ska ta, hur exponerad kameran är för ljus, hur känslig den är för avvikelser i bilden och vilket arbetsområde som kameran ska analysera datapunkterna i. [6]

2.2.4 Cognex 3D Hand-Eye Calibration

Denna programvara tillhandahåller en metod för att omvandla kamerans 3D rymd till den rymd som är känd för roboten. Detta genomförs genom att ta kort på en kalibreringsplatta som är monterad på önskat ställe på robotens arm. [6]

2.3 Verktyg

I detta avsnitt så beskrivs de mest centrala verktygen från VisionPro biblioteket kortfattat.

2.3.1 PatMax 3D Tool

Ett verktyg där ett objekts former kan tränas in utifrån punktmolnet som skapas när en objektet avbildas av kameran. Punktmolnet består en uppsättning av datapunkter i rymden där varje punkt har sin individuella X, Y och Z koordinat, där alla punkterna tillsammans bildar tredimensionella former och figurer. Genom att kapsla in datapunkterna i ett träningsområde så tränas mönstret som datapunkterna inom området ger upphov till. Det intränade mönstret används sedan som mall för objektidentifiering och orientering av liknande mönster. Varje resultat från PatMax 3D Tool består av en transformeringsmatris för varje resultatyta, därmed kan ett flertal andra parametrar utvinnas såsom mittpunkt, storlek, samt poäng för hur bra ytan matchar mönstret. [6]
I verktygets användargränssnitt så kan noggrannheten och andra parametrar justeras för vad som anses ska vara en godtagbar matchning. PatMax 3D Tool kräver en CogImage16Range som input, vilket är ett särskilt bildformat som gör det möjligt att visa en 3D-modell av den tagna bilden i punktmolnet. Ett verktyg kan endast innehålla ett tränat mönster, men kan få flera resultat med det mönstret. [6]

2.3.2 Cross Section Tool

Detta verktyg kan användas för att göra ett tvärsnitt genom bilden för att få ut en 2D-profil över det område som är specificerat av användaren. I gränssnittet för verktyget så kan sen ytterligare funktioner läggas till för att få ut önskad data utifrån profilen. Exempel på mätningar man kan göra på profilen är:

- ExtractLineSegment - Utvinner ett linjesegment från profilens kontur inom det område som är specificerat med start från lägst till högst X. [6]
- ExtractPoint - Utvinner en punkt på profilens kontur inom området som är specificerat. Toleransen för vilka punkter som är godtagbara kan justeras genom att ange ett intervall med acceptabla X och Y-värden. Det går också att ställa in genom att välja önskad punkt från en fördefinierad lista av egenskaper som punkten kan ha. Exempelvis så kan alla

punkter i profilen läggas ihop för att få fram den genomsnittliga positionen för en punkt.[6]

- ExtractCorner - Utvinner hörn på profilens kontur inom området som är specificerat. Hörnen hittas genom att undersöka området kring varje punkt på konturen. Skiljer det sig i antingen lutning eller i Z värde på konturen kring det potentiella hörnet, så anses det vara ett hörn. Se figur 16 för visualisering, anledningen till att de fyra punkterna hittas är för att det skiljer cirka 90 grader mellan punkten och konturen i någon riktning. [6]

2.3.3 Cog3D Plane Estimator Tool

Används för att få ut ett matematiskt plan utifrån antingen ett fördefinierat antal punkter, minst tre, utplacerade i XY-planet i CogImage16Range-bilden (se avsnitt 2.3.1) eller så beräknas ett medelvärde av alla datapunkter i den tagna bilden, under runtime. Planet bildas genom att de olika punkternas Z-värden jämförs med varandra. Punkterna kan ställas in till att vara allt från en pixel stora, till att ta medelvärdet i ett område kring punkten. Förutom att returnera planet så räknar detta verktyg även ut Residual Root Mean Square (RMS) felet, det vill säga hur stort spann det är mellan Z-värdena på punkterna. Desto högre RMS, desto sämre är planet anpassat till bilden. [6]

2.3.4 Cog3DFixtureScript

I VisionPro så har varje bild ett tillhörande Coordinate Space Tree som definierar root space och user spaces. VisionPro använder Coordinate Space Tree för att kartlägga koordinater från root space till andra användardefinierade koordinatsystem. [6]

FixtureScript genererar ett nytt fixerat Coordinate Space till ingående bilds Coordinate Space Tree och fäster därmed det på den utgående bilden, som i sin tur kan användas av andra verktyg. [6]

2.3.5 Cog3DVisionDataRerenderTool

Detta verktyg framställer in ny bild baserat på ett valt koordinatutrymme som finns i Coordinate Space Tree och som är skilt från det som används i ordinarie bild. Detta förbättrar effektiviteten av andra verktyg som analyserar den framställda bilden. [6]

2.3.6 CogIPOneImageTool

Ett verktyg som implementerar grundläggande bildprocesseringsfunktioner som till exempel högpassfiltrering och operationer vid saknade pixlar. [6]

2.4 Dynamic Link Library

En Dynamic Link Library (DLL) är en modul som innehåller funktioner eller data som i sin tur kan användas i en annan modul eller applikation. En DLL-fil kan omfatta två olika sorters funktioner, interna och exporterade. De exporterade DLL-funktionerna är avsedda att bli anropade från utomstående applikationer, medan de interna endast blir anropade inifrån DLL-filen där de är definierade. [9]

Det finns två typer av dynamisk länkning:

- Vid *Load-time dynamic linking* så gör systemet uttryckliga anrop till exporterade DLL-funktioner som om de vore lokala funktioner, vilket kräver en länk till dess importer library som innehåller DLL-funktionen. Import library förser systemet med nödvändig information om hur DLL-filen öppnas och laddas när applikationen körs. [9]
- Vid *Run-time dynamic linking* så använder systemet sig av LoadLibrary för att ladda DLL-filen under körningstid, för att sedan anropa GetProcAddress för att få adressen för den exporterade DLL-filen. [9]

2.5 PLC

Programmable Logic Controllers (PLC) är robusta datorer som används inom framförallt industrier, men kan appliceras i de flesta applikationer tack vare dess flexibilitet. Dessa styrenheter kan automatisera specifika processer som till exempel maskinfunktionalitet genom att erhålla information från insignaler, till att skicka ut lämpliga utsignaler med hjälp av förprogrammerade parametrar. PLC-styrsystemet som används i projektet är S7-1512SP. Enligt

IEC 61131 så finns där fem standardiserade PLC programmeringsspråk. Två av de språken som används i projektet är:

- Ladder (LD)
- Strukturerad text (ST)

Ladder är ett grafiskt språk vilket innebär att programmeringen görs genom att kombinera olika grafiska element för att bilda logiska funktioner, istället för att skriva textbaserad kod. Structured Text är ett textbaserat språk som är framtaget för att ha likadan syntax som ett vanligt högnivåspråk, med exempelvis if-satser och loopar. [2]

2.6 TCP/IP

Internets nuvarande huvudprotokoll TCP/IP är sammansatt av två individuella protokoll, Transmission Control Protocol (TCP) och Internet Protocol (IP), som tillsammans bildar ett säkert sätt att skicka och erhålla data, mellan klienter och host, genom routrar. TCP är ett anslutningsorienterat protokoll vilket innebär att det krävs en säker förbindelse för att kunna utbyta data med varandra. Detta uppfylls genom att en handskakning inleds i början av en förbindelse mellan de båda parterna. När handskakningen har genomförts så kan data skickas tills anslutningen bryts. I det tredje skiktet ligger IP som är protokollet som för vidare data från transportlagret och då sköts adresseringen med IP-adresser. [7]

2.7 Robot

IRB 1200 är en av ABB Robotics senaste generation av små 6-axliga industrirobotar med en räckvidd på 0,9 meter. Tillsammans med robot-styrenheten IRC5 så kan alla ABB system programmeras med högnivåspråket Rapid. Detta kan göras på tillhörande operatörspanel FlexPendant eller i programvaran RobotStudio. FlexPendant är försedd med en joystick så det finns möjlighet att köra roboten manuellt, vilket bland annat används vid inställning av verktyg som är fastmonterade på roboten så att spetsen på verktyget, även kallat dess TCP, har den korrekta utgångspunkten för var man befinner sig i rymden.[1]

Roboten kopplas ihop med PLC-styrsystem via PROFINET som är ett Ethernet-baserat kommunikationssystem vilket inkluderar ett cykliskt utbyte av IO-data. [1]

2.8 Rymdpolära koordinater

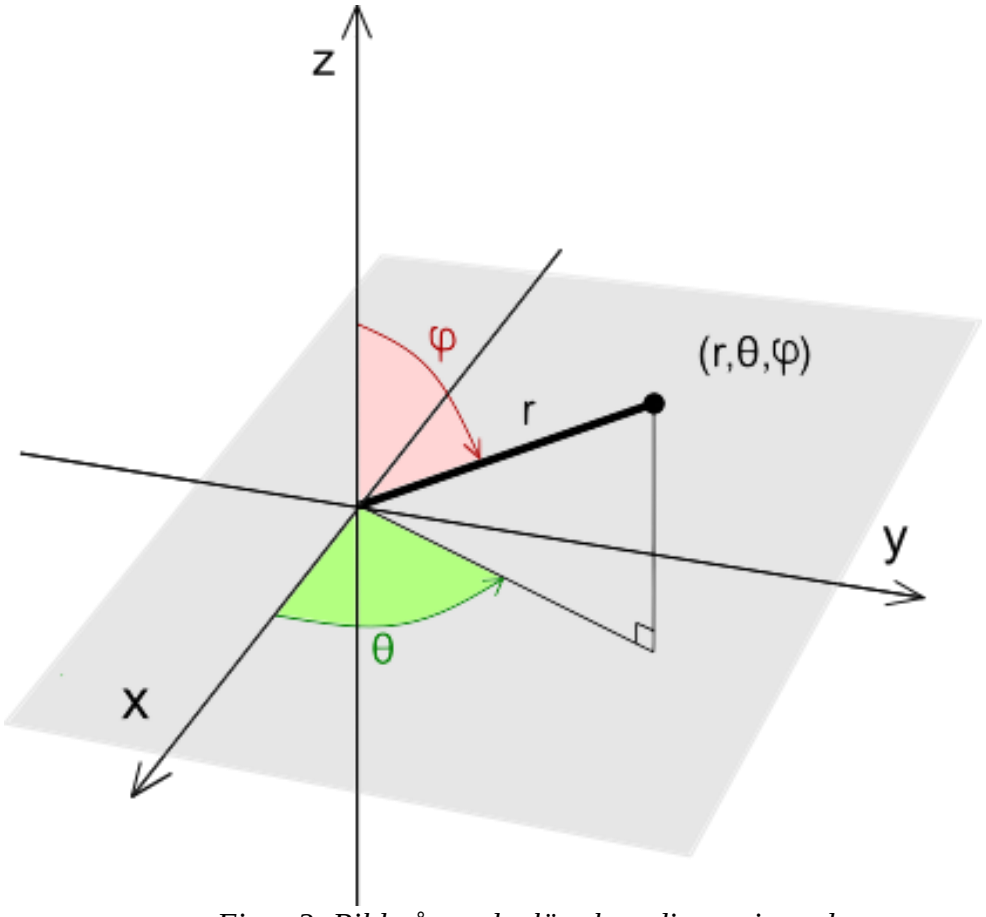
En generalisering av polära koordinater till tre dimensioner är det som kallas Rymdpolära eller sfäriska koordinater. Ett substitut till de rätvinkliga koordinaterna (x, y, z) i rummet är i rymdpolära koordinater alltså (r, φ, θ). Sambandet mellan dessa kan beskrivas på följande sätt [8].

Enligt bilden nedan så är P: (x, y, z) punkten som vill beskrivas, och Q betecknas som den ortogonala projektionen av P på XY-planet. Vidare så benämns r till att vara avståndet mellan origo och den sökta punkten (OP), vinkeln omega mellan den positiva Z-axeln och OP och vinkeln θ i XY-planet mellan den positiva x-axeln och linjen OQ. [8] Genom att använda samma trigonometriska tillvägagångssätt som med uträkning av polära koordinater så fås följande koordinatsamband med de nya definitionerna till slut:

$$x = r\,sin\varphi\,cos\theta$$

$$y = r\,sin\varphi\,sin\theta$$

$$z = r\,cos\varphi$$ [8]



Figur 3: Bild på rymdpolära koordinater i rymden

3 Metoder

Under arbetets gång har kommunikation med handledare skötts via e-mail, telefonsamtal och videomöten, inga möten har skett fysiskt till följd av Corona pandemin
Kommunikation mellan skribenter har skett via Google Drive, e-mail och telefon
Kommunikationen med företaget har skett genom mejl, videomöten, telefonsamtal men främst genom fysiska möten eftersom majoriteten av arbetet tagit plats på företaget, vilket varit en följd av att utrustningen för projektet funnits där.

Genomgående har det praktiska arbetet utförts som parprogrammering till följd av två anledningar. För det första har endast funnits hårdvara och mjukvarulicens för en dator och det har inte varit ekonomiskt försvarbart att införskaffa mer materiel. För det andra har Cognex Designer och 3D-vision varit ett nytt område för skribenterna.

3.1 Planering

Planering av arbetet gjordes tillsammans med en liten grupp av anställda på uppdragsgivarens sida, samt med handledare på skolan. Utöver de hjälpmedel som var fastställda av företaget så diskuterades vilka extra redskap som krävdes för att utföra uppgiften. Det som redan var bestämt i förväg var vilken sorts kamera och tillhörande mjukvara som skulle användas till projektet. I samråd med handledare på företaget så bestämdes det sedan också att testobjekten skulle bestå av papplådor i form av rätblock, då framförallt storleken var passande. Eftersom de har ett samarbete med företaget Eyetech (som är samarbetspartner med Cognex) så skapade det möjligheten att låna deras 3D-visionkamera 3DA5120 med tillhörande huvudprogramvara Cognex Designer med integrerat VisionPro, och de andra mer begränsade programvarorna A5000 Viewer och Cognex 3D Hand-Eye Calibration.

Därefter gjordes en övergripande tidsplan för hur projektet troddes fortlöpa. Denna tidsplan fick förbises redan efter första veckan av arbetet då hårdvara och mjukvara inte fanns tillgängligt. Följaktligen infördes en veckoplanering som tog plats i slutet av varje arbetsvecka, där veckoplaneringen avsåg kommande veckas uppgifter och mål. Utöver veckoplaneringen

gjordes även en sammanfattning av vilka delar som kunde tänkas vara nödvändiga att inkludera för att uppfylla kraven.

3.2 Förstudier

Som introduktion till mjukvaran Cognex Designer anordnade företaget en föreläsningsdag med Eyetech AB vilka har ett partnerskap med Cognex Corporation. Denna introduktion omfattade mestadels 2D-vision. För mer förståelse för 3D-vision utnyttjades Cognex hemsida och Cognex Designers Dokumentation, exempelvis hur träning av Patmax3D fungerar, som är ett förprogrammerat verktyg i Cognex. För vidare förståelse för hur Cognex Designer fungerar utfördes övningsuppgifter som tillhandahölls från Cognex hemsida. Övningsuppgifterna gav en övergripande förståelse för hur utvecklingsmiljön var uppbyggd, exempelvis förståelse för flödesschemat.

3.3 Laboration

Som del av projektet behövdes kommunikation mellan Cognex Designer, PLC och Robot. Eftersom Cognex Designer var en ny utvecklingsmiljö gjordes mindre laborationer för att undersöka hur en fungerande kommunikation mellan PLC och Designer kunde upprättas.

3.4 Programmering

Programmeringen av visionsystemet gjordes huvudsakligen i tre utvecklingsmiljöer vilket var Cognex Designer, Siemens Tia Portal och Robot Studios. Det första utvecklingsmomentet berörde Cognex Designer, som var det som upptog störst andel tid under denna fas. Anledningen till detta var för att var det en främmande miljö vilket gjorde mjukvaruverktyget mer komplicerat, samt att det var denna del som företaget var mest intresserade av.
Som styrenhet för hela systemet programmerades en PLC med utvecklingsverktyget Siemens Tia Portal.

Sista momentet var det som tog minst tid då detta krävde minst kodning och här användes Robot Studios för att skapa ett önskat rörelsemönster för roboten.

3.5 Källkritik

[7], [8] Dessa källor anses vara trovärdiga då det är akademisk litteratur som använts vid undervisning av kurserna flerdimensionell analys respektive dator- och telekommunikation vid Lunds Tekniska Högskola.

[3],[4],[5],[6] Cognex är det företag som utvecklat programvaran Cognex Designer. Cognex Designer har undersökt och utvecklat kommersiella vision-lösningar sedan 1980-talet, vilket gör källan trovärdig.

[9] Microsoft Corporation är ett av världens ledande företag inom software development och är det företag som utvecklat DLL därför kan denna källa anses vara trovärdig.

[1] ABB automation Company är ett multinationellt företag som bland annat utvecklar robotar och automationslösningar. Denna källa kan anses vara trovärdig då detta företag är utvecklare av IRB 1200.

[2] Denna källa anses vara trovärdig eftersom den är inkluderad i Lunds universitets bibliotekskatalog på LUBcat.

4 Analys

4.1 Målsättning

Målet med arbetet var att skapa ett nytt 3D-visionsystem med tre kommunicerande parter som kunde uppfylla följande krav.

Vision

- Hitta objekt. Få ut centrumkoordinater på objektets hittade ytor.
- Hitta flera objekt i samma bild.
- Hitta objekt som ligger mot varandra
- Hitta objekt som ligger delvis på varandra
- Hitta objekt som ligger bredvid varandra på/med olika höjder

PLC

- Kommunikation till vision, läsa in objektkoordinater.
- Kommunikation till robot, skicka plock- och läggpositioner.
- Enkel sekvens som triggar vision, skickar koordinater till roboten, startar roboten att plocka.

Robot

- Definiera användarkoordinatsystem och verktyg för roboten.
- Plocka och placera objekt med roboten.

4.2 Konfigurering av kameran

För att få en analyserbar avbildning så krävs det att kamerans egenskaper anpassas till de förhållandena som finns i den miljön kameran befinner sig i. För att lättast visualisera och testa ändringarna som görs så används den separata programvaran A5000 Viewer. Det första som gjordes var att ändra exponeringstiden, vilket är den totala tiden som A5120 sensorn är exponerad för ljus. En låg exponeringstid kan leda till underexponerad bild vilket orsakar mörka bilder med många saknade pixlar som inte går att analysera. För hög exponeringstid leder till överexponerade bilder där ljuset blir så skarpt att det även då bildas saknade pixlar i bilden. Då det är denna parameter som påverkar förvärvstid mest, så laborerades det noga med den för att få de skarpaste avbildningarna med så kort exponeringstid som möjligt. Testobjekten som användes under projektet var identiska rektangulära lådor se figur 4.

Ytan som bildades av de avbildade lådorna var inte slät nog för att få den precision som krävdes. Detta berodde på att det fortfarande förblev kvar små variationer i ljusstyrka över hela den bearbetade bilden. För att åtgärda detta så ökades antalet bilder kameran skulle ta från 24 till 40, som därmed slätade ut ytan. Detta gjorde att fick en mer detaljerad rekonstruktion som skulle leda till mer konsekventa resultat från verktyg som skötte bildbearbetningen. Detta ökade förvärvstiden markant och det har förblivit en stor nackdel med systemet. Genom att specificera en mindre kameraupplösning, för att begränsa bildens storlek så det matchade arbetsområdet, så kunde dock förvärvs- och bearbetningstiden sänkas.

När förvärvsparametrarna för kameran var optimerade så exporterades de in i ett Deviceblock som är beläget först i händelsekedjan för programmet i Cognex Designer. Utgångarna från detta block består bland annat av den tagna bilden i önskat format. För att utveckla detta system så behövdes det ett format där Z-axeln kunde urskiljas från bilden, utöver X och Y. Det konstaterades då att CogImage16Range var det sökta formatet som dessutom var det önskade formatet för många av verktygen i VisionPro. Förklaras i avsnitt 2.3.1.

Figur 4. Lådorna som användes som testobjekt i projektet

4.3 TCP/IP

I Visionsystemet användes en TCP/IP-förbindelse mellan PLC:n och Cognex Designer, detta valdes eftersom vid kommunikation till och från robot krävdes profinet kommunikation. Då profinetkort inte fanns tillgängligt för Cognex Designer behövdes en mellanhand för att kunna överföra koordinater från Cognex Designer till ABB-roboten, som mellanhand valdes att en PLC skulle användas.

Som kommunikationsalternativ valdes att Cognex agerade host och PLC agerade klient, detta till följd av att Cognex inte kunde spara connections och kunde därför inte komma ihåg vilka enheter den har pratat med. Detta ledde till kommunikation från Cognex till PLC blev en broadcast till alla klienter medan PLC skickar meddelanden specifikt till Cognex.

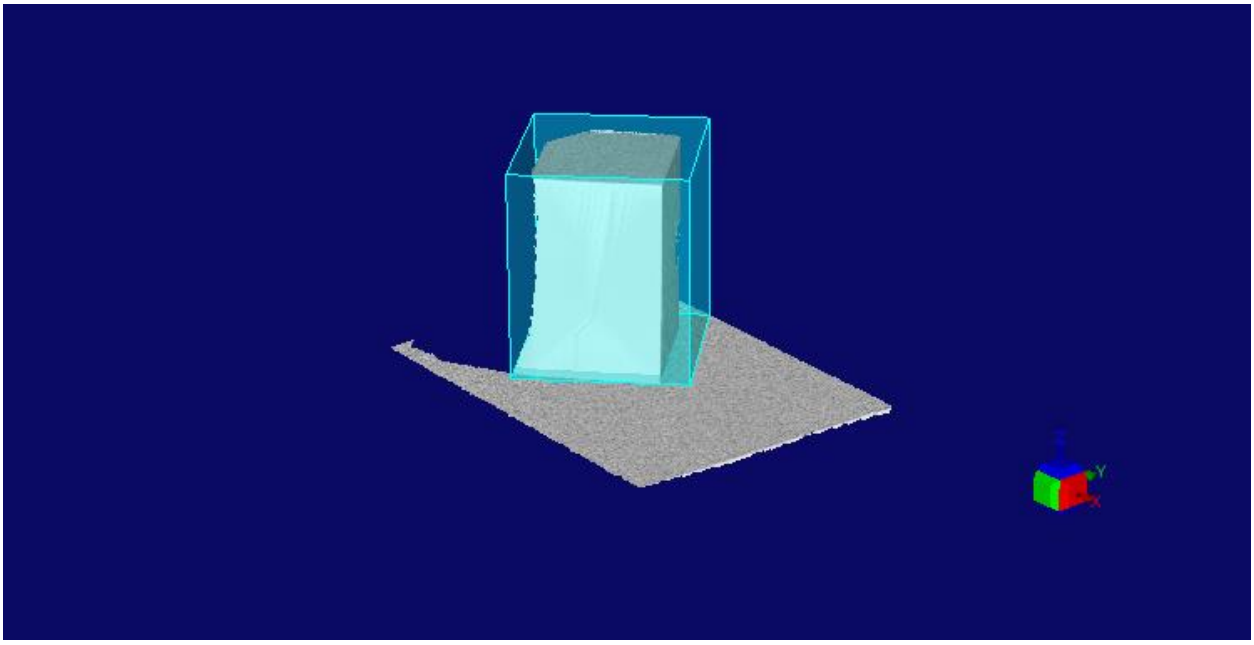
För mottagning av meddelande i PLC krävdes att längden på meddelandet var känt, eftersom storleken av meddelandet var tvunget att bli uppfyllt innan funktionsblocket

ansåg meddelandet som mottaget. Detta ledde till att Cognex programmerades till att skicka längden på data-meddelandet innan data-meddelandet skickades. Vid mottagande av längden var det känt att antalet tecken var tre, vilket berodde på att längd-meddelandet alltid hade ett värde mellan 100 och 200 (därmed skickas 3 tecken). När Cognex inte producerat ett resultat sändes ett meddelande som var 999 alternativt 998 vid omtagning av bild. Eftersom storleken av meddelandet alltid var tre kunde en fast storlek antas vid mottagning av längd-meddelandet. Mottagningen av koordinater och vinklar sker enligt följande, först tas koordinaterna emot som char:s i en char array, sedan sorteras koordinater och vinklar med hjälp av en sorts skiljetecken, i form av semikolon, som är inkluderade i meddelandet från Cognex Designer. Skiljetecknet definierar slut på varje separat parameter och en början på nästa parameter. Därefter görs varje parameter om till string:s för att sedan göras om till real:s (vilket är Tia Portals motsvarighet till Double:s). Anledningen till att alla parametrar skickas som ett meddelande är för att Robot Studios ska läsa av var objektet finns positionsvis och inte parametervis. För att undvika att parametrarna sedan skickas vid olika tillfällen till robot studios är det då lättare att ta emot hela positioner till PLC:n.

Ett annat problem som framkommit är att meddelanden mellan Cognex och PLC:n inte alltid anländer till mottagaren, vilket tros bero på att samma connection används vid alla kommunikationstillfällen. Detta har resulterat i att när denna länk stängs ned för att sedan startas upp igen sker ett kommunikationsfel vilket har lett till att timer används för att känna av timeouts. Om timeout sker skickas återigen senaste förfrågan från PLC:n till Cognex Designer.

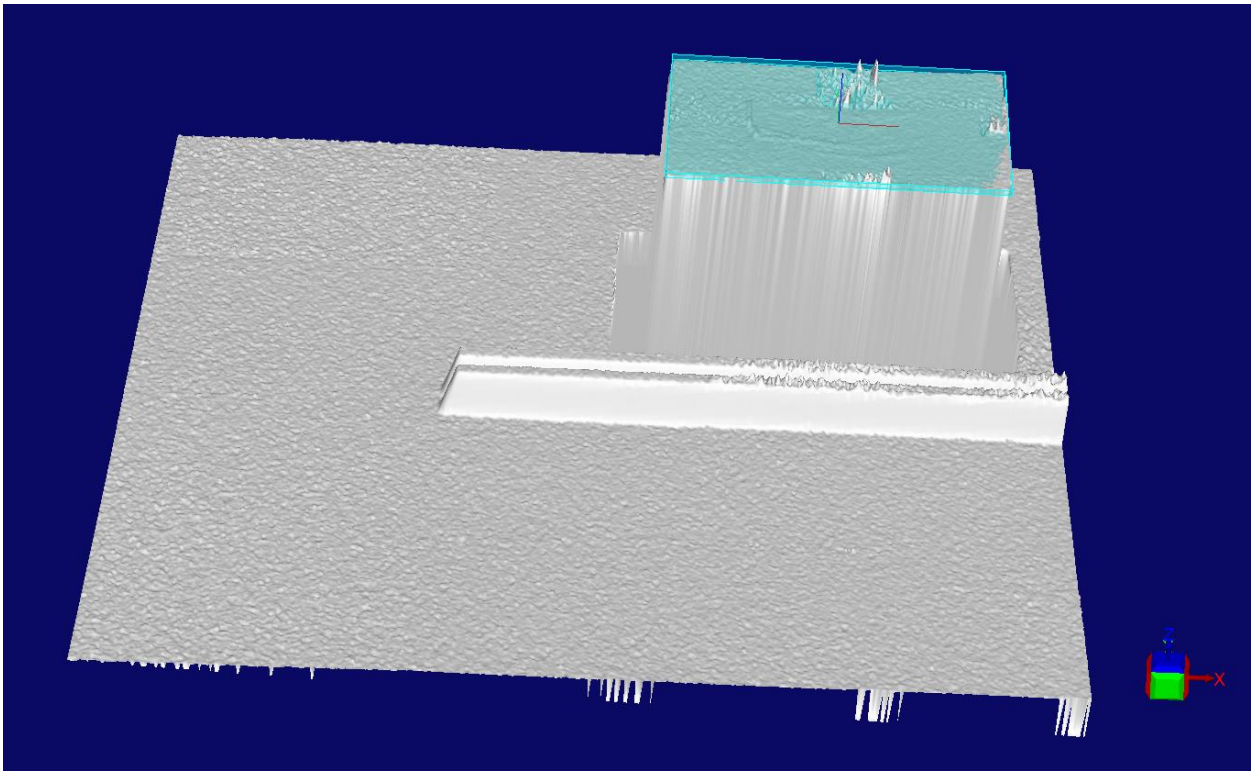
4.4 Bildbearbetning

Vid början av undersökningen så lades mycket betänketid på hur bildbearbetningen skulle gå till i Cognex Designer. Till en början så låg fokuset på att utveckla en algoritm som automatiskt kunde identifiera objekt placerade inom kamerans arbetsområde, vilket var första punkten i målspecifikationen. I Cognex Designers meny finns ett valbart block där programvaran VisionPro är integrerat, som heter VisionProToolBlock. I det blocket så bestämdes det att applikationens bildbearbetning skulle kretsa kring PatMax 3D Tool verktyget se avsnitt 2.3.1. Till en början så kapslades hela lådan in i träningsområdet, men att träna in flera sidor i samma mönster visades vara ett mindre bra beslut. Detta eftersom inträning av flera sidor gjorde det svårt för PatMax 3D att känna igen det intränade objektet när det var vridet eller då det på annat sätt visades andra sidor än mönstret som tränades in.



Figur 5: Bild från inträning av en hel låda där 3 av sidorna syntes

Det valda tillvägagångssättet blev istället att varje unik sida tränades in som mönster i varsitt PatMax 3D Tool enligt figur 6.



Figur 6: Inträning av ena sidan på lådan

Detta eliminerade felet som tillkom med träning av hela lådan. Nu kunde sidorna som syntes för kameran analyseras separat, istället för att analysera hela lådans som ett stort mönster med risk för att få oönskat skymda sidor och därmed ickematchat mönster.

Eftersom PatMax 3D använder sig av en identifierings-algoritm som gör att tränat mönster varken kan skalas upp eller ner på någon av de tre axlarna, så betyder det att PatMax 3D letar efter matchningar där måtten på respektive sida av lådan är densamma som på den ursprungliga träningen. Detta betyder att även om mönstret är rektangulärt, så måste även längden och bredden stämma överens. Därför valdes tre träningsmönstren som placerades i varsitt VisionProToolBlock enligt följande struktur, se figur 4.

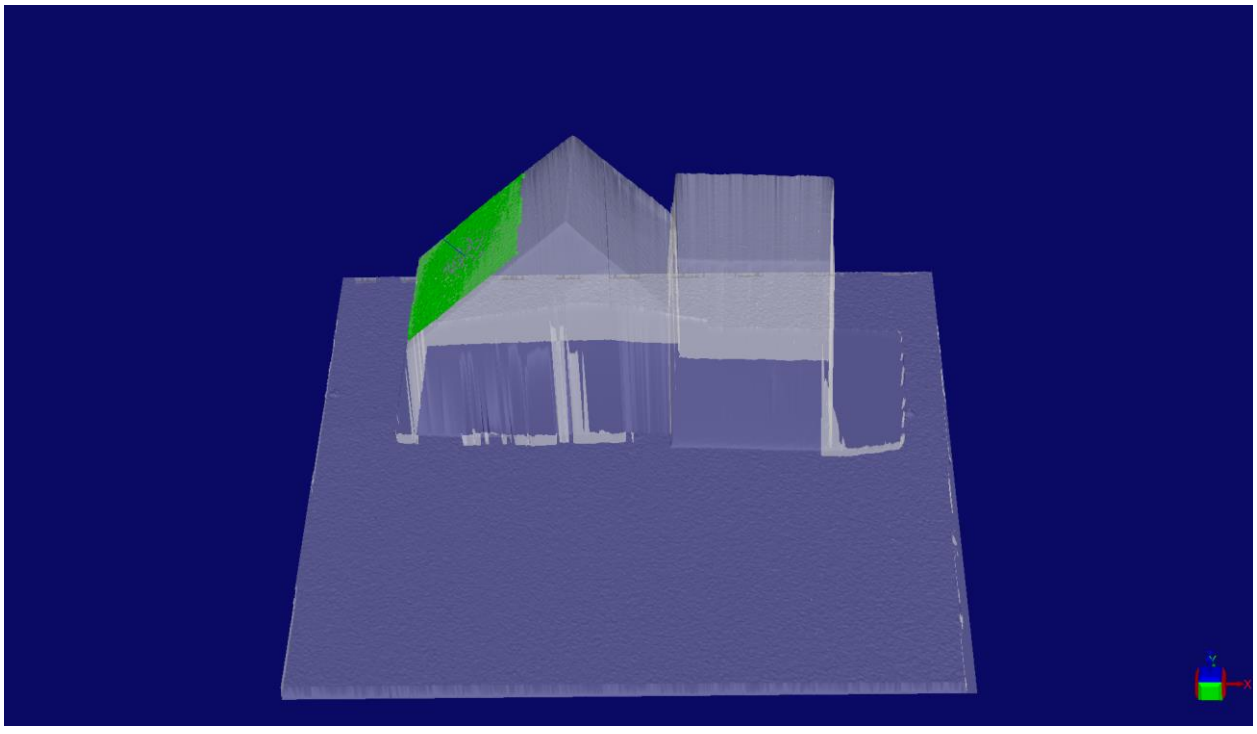
- Pattern1: Träning av toppdelen av lådan.
- Pattern2: Träning av den smala sidan av lådan.
- Pattern3: Träning av den breda sidan av lådan.

4.4.1 Basplanet

För att hela systemet skulle ha ett koordinatsystem att enas om så skapades ett basplan längs med arbetsytan. Fyra punkter sattes ut på den tagna bilden så arbetsytan lokaliserades och beräknades se avsnitt 2.3.3

4.4.2 Felaktigt hittade objekt

När verktyget körs så fås resultaten i en lista som kan visas i användargränssnittet. Varje matchat mönster har ett medföljande Score som avser hur många pixlar som överensstämmer med det tränade mönstret. Ett Score på 1.00 betyder att mönstret teoretiskt sätt är en fullständig match men det betyder inte att det är ett korrekt. För som scenariot visar nedan så kan Pattern1, som är toppen på lådan, få plats på de resterande sidorna på lådan med högt Score, fastän det är felaktigt, se figur 7.



Figur 7: felaktigt hittat mönster som utgörs av det gröna området.

Eftersom PatMax 3D Tool inte var konsekvent på att hitta rätt mönster, så utvecklades en sorteringsalgoritm för att sortera bort de icke existerande objekten. För att utvinna hörnpunkterna från det funna mönstret användes en enumerator (se avsnitt 7) för att iterera genom objektet i VisionPro som innehöll hörnpunkterna. Enumeratorn skapades med följande

kod:

enumerator = pat. Results[index]. GetBoundingBox(). GetVertices(). GetEnumerator();

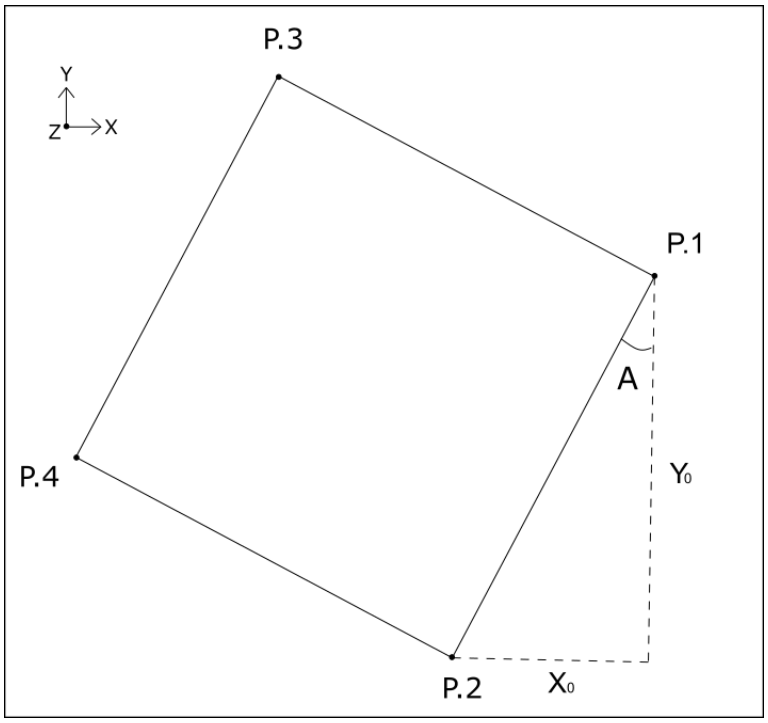
Enumeratorn användes sedan i funktionen getPoints() som returnerar en tvådimensionell double matris innehållande hörnpunkterna.

curRes = getPoints(enumerator);

För att använda hörnpunkterna i beräkningar så var de först tvungna att sorteras, eftersom de inte låg i ordning när de hämtades. Punkterna sorteras med minskande X-värde och i andra hand minskande Y-värde. Detta betydde att hur objektet än var orienterat på ytan så var hörn 1 alltid högst upp till höger, hörn 2 längst ner till höger, hörn 3 längst upp till vänster och hörn 4 längst ner till vänster. Se figur 8

Förhållandena mellan dessa fyra hörn- eller vertexpunkternas koordinater används för att få ut projektionen kring de tre axlarna. Genom att använda de trigonometriska sambanden på trianglarna som bildas mellan hörnen i rymden så fås projektionen runt de tre axlarna, det vill säga orienteringen kring x-, y- och z-axeln.

Rotationen kring Z-axeln (vinkeln A) fås genom en jämförelse mellan punkt 1 och 2 i XY-planet se figur 8.



Figur 8. Låda orienterad runt Z-axeln

Denna rotation fås med följande beräkning:

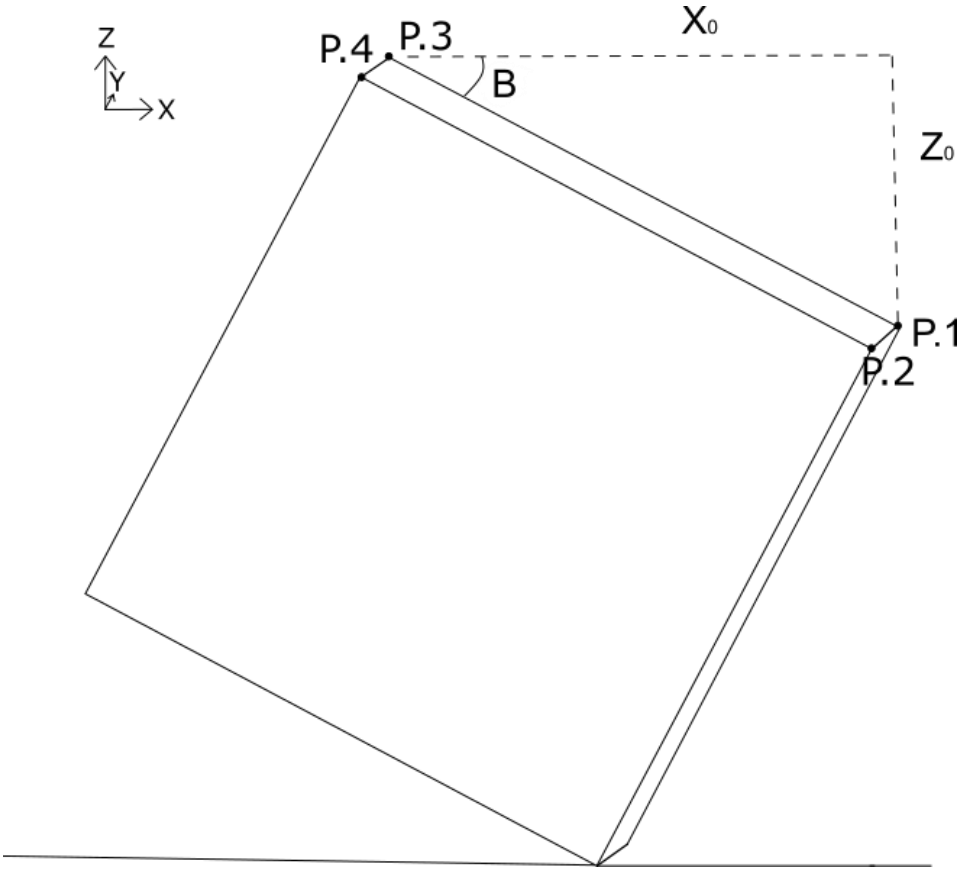
$$Y0 = | P.2(Y) - P.1(Y) |$$

$$X0 = | P.2(X) - P.1(X) |$$

$$A = \text{Arctan}(X0/Y0)$$

Har P.2 högre X-värde än P.1, det vill säga att objektet är roterat åt andra hållet, så blir A = -A.

Rotationen kring Y-axeln (vinkeln B) fås genom en jämförelse mellan punkt 1 och 3 i XZ-planet se figur 9.



Figur 9. Rotationen kring Y-axeln

Denna rotation fås med följande beräkning:

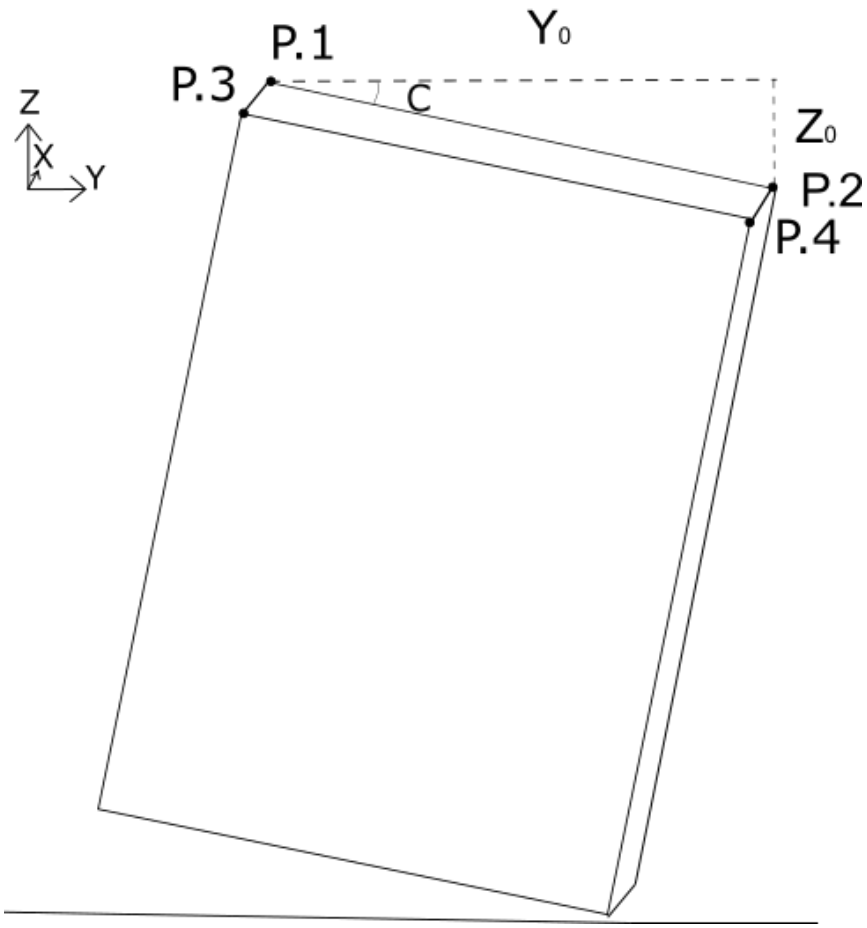
$$Z0 = | P.3(Z) - P.1(Z) |$$

$$X0 = | P.3(X) - P.1(X) |$$

$$B = \text{Arctan}(Z0/X0)$$

Har P.1 högre Z-värde än P.3, det vill säga att objektet är lutat åt andra hållet, så blir B = -B.

Rotationen kring X-axeln (vinkeln C) fås genom en jämförelse mellan punkt 1 och 2 i YZ-planet se figur 10.



Figur 10. Rotationen kring X-axeln

Denna rotation fås med följande beräkning

$$Z0 = | P.2(Z) - P.1(Z) |$$

$$Y0 = | P.2(Y) - P.1(Y) |$$

$$C = \text{Arctan}(Z0/Y0)$$

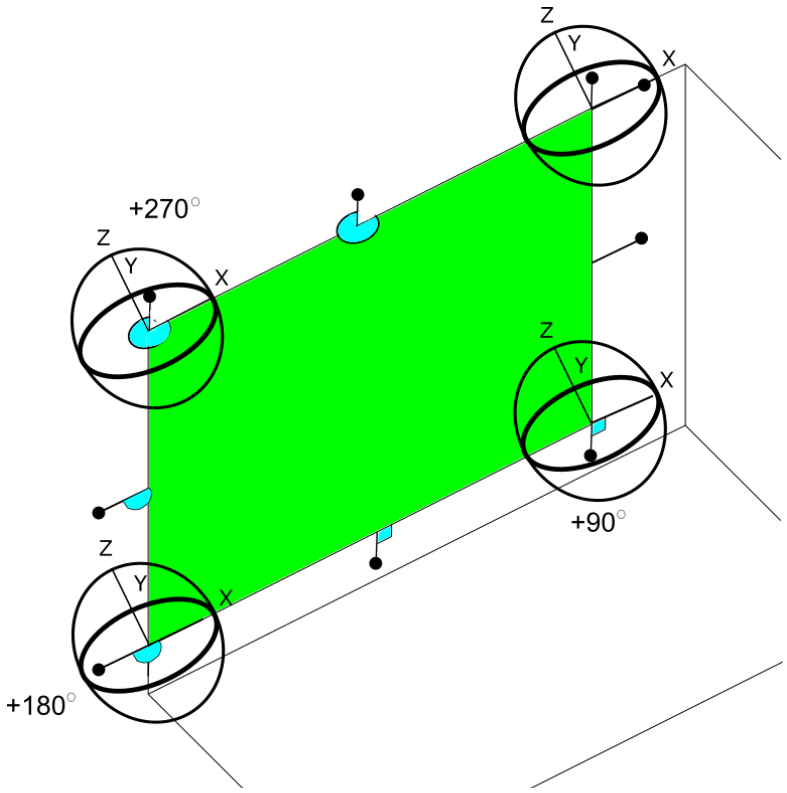
Har P.2 högre Z-värde än P.1, det vill säga att objektet är lutat åt andra hållet, så blir C = -C.

Vinklarna som beräknades används sedan för att hitta felaktiga matchningar genom att beräkna ett steg ut från framtagna hörnpunkter i riktning med mönstrets yta. När ett steg har gjorts till en ny beräknad punkt så undersöks det om det finns en datapunkt i punktmolnet som går att analysera, vid just den platsen. Finns där en datapunkt så har inte PatMax 3D lyckats matcha mönstret perfekt, och det anses därmed vara en felaktig matchning. För beräkning av steget ut

så används teorin om rymdpolära koordinater, se avsnitt 2.8. En punkt i rymden kan beskrivas enligt formeln vid avsnitt 2.8 som i detta fall blir:

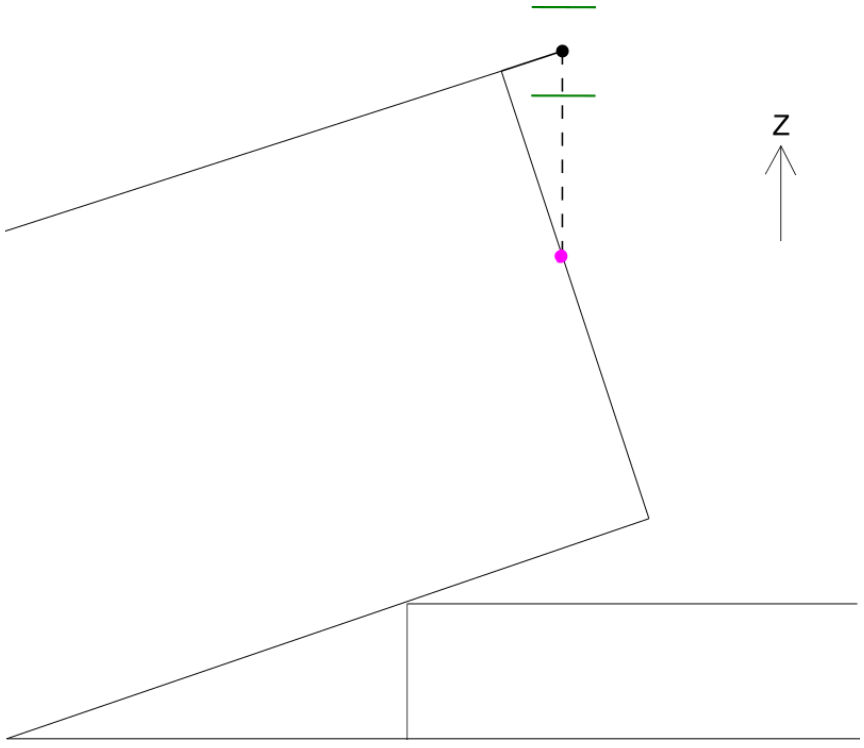
$$x = startX + r\sin(B) * \cos(A)$$
$$y = startY + r\sin(B) * \sin(A)$$
$$z = startZ + r\cos(B)$$

Där φ enligt figur 3 är B och A är θ. För att utesluta att ett mönster är felaktigt så går funktionen ut på att iterera ett steg ut genom ett valt antal punkter på vardera sida av mönstret. StartX, startY och startZ är därför punkterna som steget beräknas från eftersom det inte utgår från origo. Längs med första sidan, mellan P.1 och P.2, se figur 8, så är steget i riktning med X-led och påverkas endast av vinklarna B och A. På nästa sida av det rektangulära mönstret, mellan P.2 och P.4 enligt figur 8, så adderas A med 90 grader för att steget ska gå vinkelrätt mot sidan. Samtidigt så påverkas inte längre steget av vinkel B längre då steget är i riktning med Y-axeln, utan ersätts då med C i formeln. Vid nästa sida så adderas 90 grader igen, och eftersom det är P.1 och P.2 sidans motsats så används vinkel -B istället. På samma tankesätt så byts det tecken till -C på sista sidan mellan P.3 och P.1. Figur 11 illustrerar detta.



Figur 11: Steg ut från en felaktig yta där svart prick motsvarar den beräknade rymdpolära koordinaten och rotationen adderas 90 grader för varje sida.

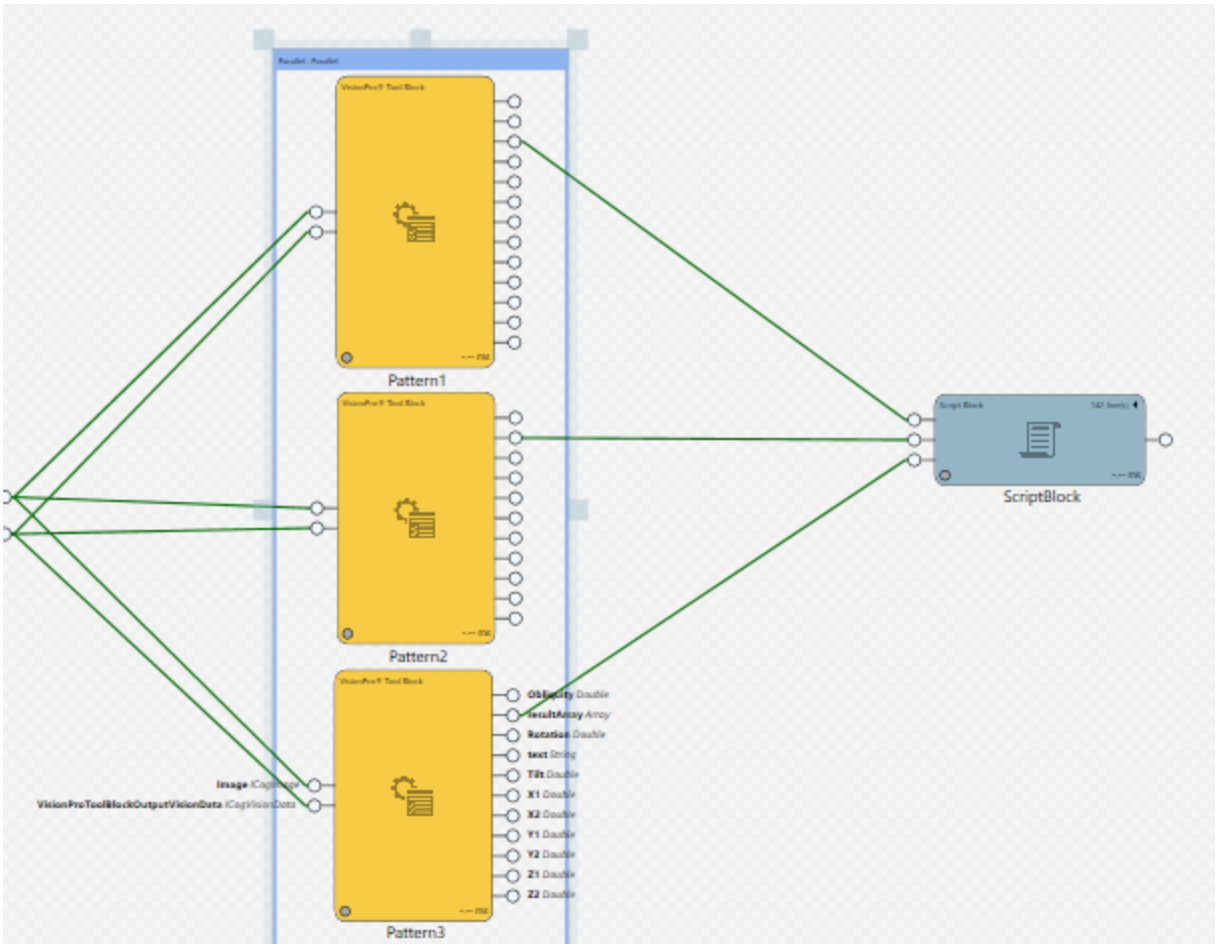
Efter varje enskilt utfört steg så används VisionPro verktyget Cross Section Tool för att avgöra om det finns en giltig datapunkt i den aktuella punkten, se avsnitt 2.3.2. I programmeringsmiljön så ställdes körningsparameterna in för verktyget så att ett litet tvärsnitt gjordes vid den beräknade punktens X och Y koordinat. På den skapade profilkonturen, som endast består ett fåtal datapunkter, så används operatorn ExtractPoint för att få ut den genomsnittliga punkten från profilen. Sedan jämförs Z-värdet från den extraherade punkten med punkten som beräknades med rymdpolära koordinater, se figur 8. Denna jämförelse görs på så sätt att om Z-värdet från den extraherade punkten infaller inom intervallet av den rymdpolärakoordinatens Z-värde + 10 och rymdpolärakoordinatens Z-värde - 10 så anses punkten och koordinaten matcha, figur 12 illustrerar detta. Vid matchning så är det konstaterat att det är felaktigt mönster och inga fler steg görs. Har alla sidor undersökts så är det perfekt matchat mönster.



Figur 12: Figuren visar ett godkänt steg ut från mönstret där den svarta pricken är den beräknade rymdpolära punkten där de gröna linjerna utgör Z-intervallet. Den rosa punkten är vad tvärsnittverktyget Cross Section Tool extraherar från bilden.

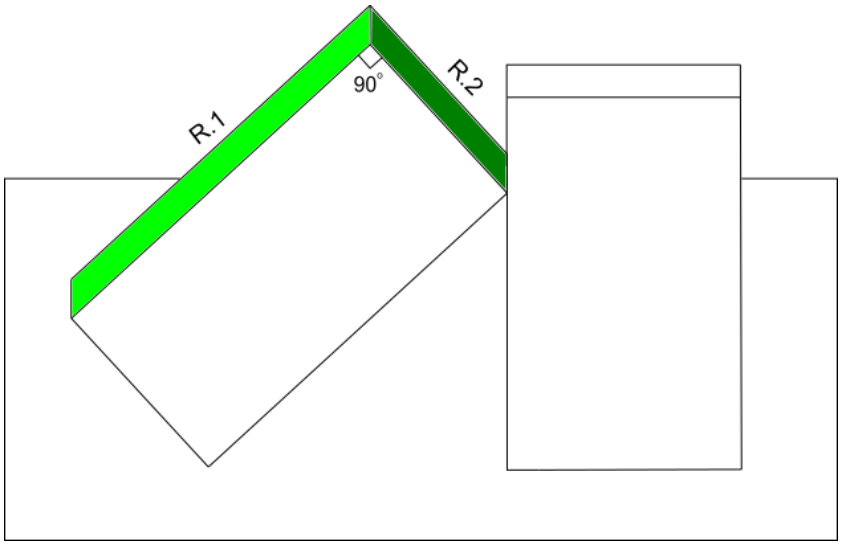
4.4.3 Utsortering av mönster

När lådorna i bilden är lutade så är flera sidor av lådan exponerade för kameran. Problemet som uppstår då är att flera korrekta matchningar fås på samma låda. Om detta hade förbisetts och data hade skickats vidare från vardera mönster så hade det lett till att roboten hade försökt plocka samma låda flera gånger. För att förhindra detta så programmerades ett ScriptBlock med en utsortering för att urskilja om fler mönster tillhörde samma låda. Utsorteringens funktion gick ut på att jämföra vinklarna mellan de tre olika mönsterna Pattern1, Pattern2 och Pattern3. För att sorteringen skulle vara möjlig så var resultaten tvungen att anlända i ScriptBlocket samtidigt, vilket de inte gjorde från början till följd av att VisionProTool blocken exekverades sekventiellt i tasken. Åtgärden för detta var att använda ett så kallat Parallel Region och innesluta Pattern1, Pattern2 och Pattern3 se figur 9. Parallell Region medför en parallell exekvering av de block som komponenten omsluter. Genom att på så vis använda ytterligare kärnor i CPU:n så minskar även exekveringstiden av programmet.



Figur 13: Bild på den implementerade subtask:en med ett omslutande Parallell Region.

Eftersom lådorna i detta fall har formen av ett rätblock konstaterades att de tre inträna de mönstrens lutning, från samma låda, alltid skiljes 90 grader emellan se figur 14. Följaktligen jämfördes lutningen av mönstrens mellan varandra (R1 och R2 i figur 14), om lutningen skilde sig exakt 90 grader sorterades mönster från samma låda bort så att endast ett av dem plockas av programmet.



Figur 14. Ett scenario där utsortering behövs, där R.1 respektive R.2 är mönster från separata PatMax 3D verktyg på samma låda.

4.5 Profinet och robot

Utbytet av data mellan PLC och ABB-roboten skedde som tidigare nämnts via kommunikationsprotokollet profinet, då I/O-utbytet mellan dessa skedde kontinuerligt fanns en styrsignal från PLC till ABB-roboten som sa när den skulle plocka ett objekt. När roboten sedan hade utfört sin uppgift och lämnat objektet på avsedd plats skickades en svarssignal så att PLC:n visste att roboten var klar och redo för en ny uppgift.

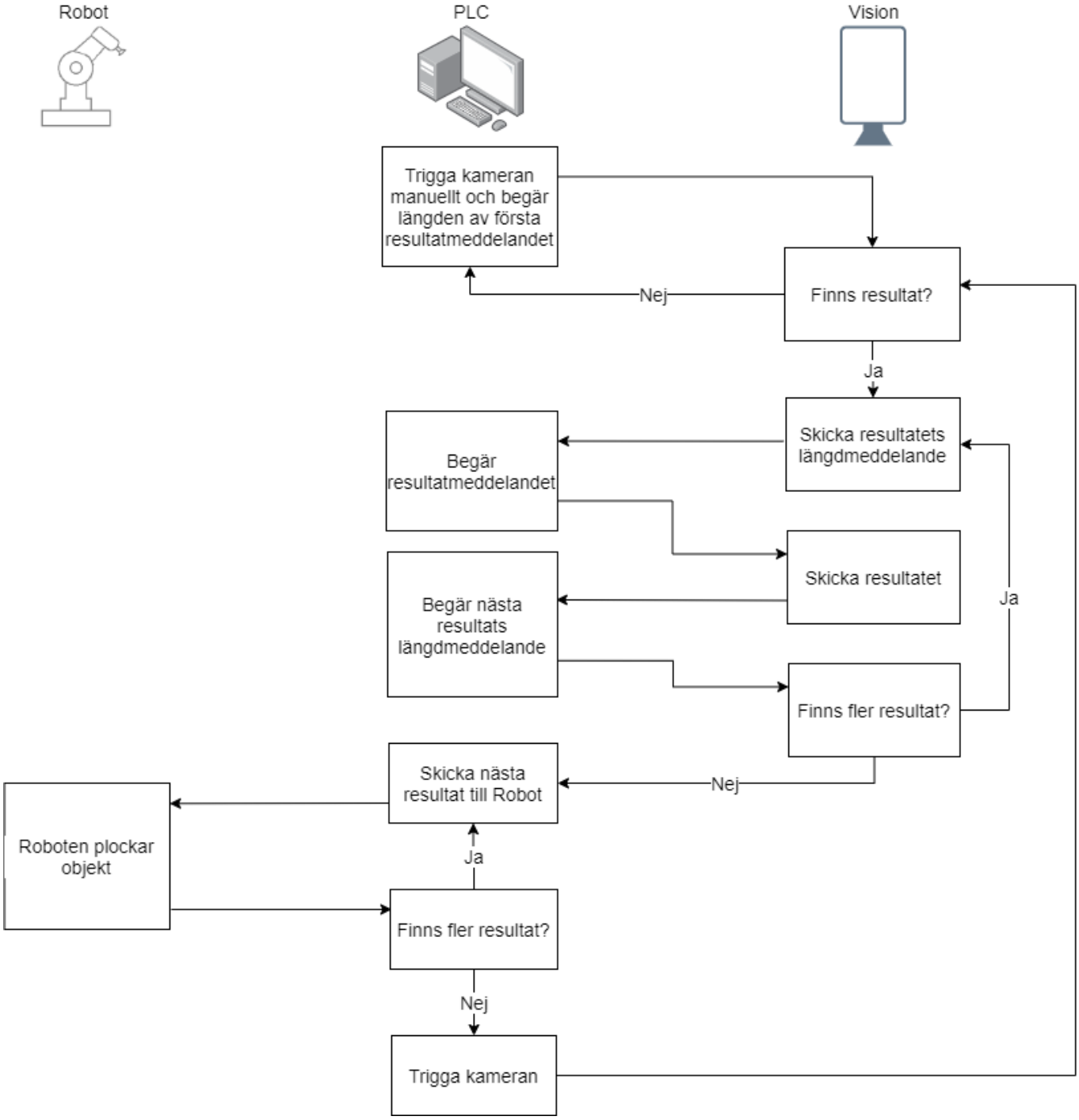
När Outputen från PLC blev till input till roboten kördes en algoritm för att ändra Inputbitarna från Big Endian till Little Endian. Detta gjordes för att få rätt värden för roboten att plocka då PLC räknade med Big Endian och roboten med Little Endian.

En viktig del för projektet var att synkronisera koordinatsystemet från Cognex Designer

med ABB-robotens koordinatsystem. Synkroniseringen av koordinatsystemen gjordes med hjälp av Cognex mjukvara 3D Hand-Eye Calibration. Kalibreringen av koordinatsystemen gjordes enligt följande, först fick ABB-roboten hålla ett föremål framför kameran, varpå kameran tog en bild av föremålet som manuellt tränades in. Därpå kördes roboten manuellt till nya positioner som fotades av kameran, för varje position skrevs de koordinater som roboten var lokaliserad vid in i en textfil enligt formatet. Textfilen användes sedan av 3D Hand-Eye Calibration för att skapa en XML-fil för att kunna anpassa kamerans koordinatsystem efter robotens. Denna XML-fil validerades genom att köra roboten manuellt till de angivna punkterna i mjukvaruprogrammet. Kalibreringen gjordes om flera gånger då valideringen i 3D Hand-Eye Calibration inte blev godkänd av programmet. Detta visade sig bero på att föremålet som använts vid kalibreringen måste vara tredimensionellt för att transformationen av positionerna ska blir korrekt beräknade. Detta löstes genom att istället för att försöka använda den vanliga testlådan, så användes en kalibreringsplatta som vid nästan alla positioner visade alla sidor och korrekta transformationer kunde göras av programmet.

5 Resultat

Det slutgiltiga resultatet av det automatiserade 3D-visionsystemet bestod av tre delar, och dess kommunikation med varandra är uppbyggt enligt följande figur.



Figur 15: flödesdiagram representerande det färdiga systemet

Det automatiserade visionsystemet sätts igång genom att manuellt skicka ett meddelande till Cognex Designer via en TCP/IP kommunikation, med instruktioner om att exekvera Tasken vilket då även triggar kameran. Om det inte finns något tillgängligt resultat efter en bild har tagits så väntar PLC:n på att bli manuellt påsatt igen, eftersom det då går att konstatera att det inte finns några objekt i bilden. Utöver att exekvera programmet så görs det i samma instruktion en begäran att få tillbaka längden på den sträng från det första resultatet, som innehåller all data som mittpunktskoordinater, lutningar i de olika axlarna och mått på lådan. När Tia Portal har tagit emot värdet på strängens längd så förbereds genast det nya mottagsblocket genom att ställa in så att strängens längd är längden på det meddelande som blocket förväntar sig få vid nästa mottagning, som då är själva strängen med data. När all information har extraherats från strängen så skickar PLC:n begäran om att få längd på strängen från nästa resultat. Denna sekvens loopas igenom tills alla resultat från Cognex Designer är skickade. När Cognex Designer har skickat sitt sista resultat och PLC:n ännu en gång begär längd på strängen så svarar programmet med att det är tomt, och då integreras nästa del av kommunikationen.

PLC:n kommunicerar med IRB1200 roboten tack vare PROFINET IO vilket medför att utbyte av data sker snabbt mellan parterna. Genom att ha mappa vilka ingångar och utgångar som används så tar RobotStudio emot och sparar den nödvändiga datan i I/O registret, som sedan används i RAPID-programmeringen för robotens rörelsemönster. Datan som PLC:n skickar vidare består av följande (Se avsnitt 4.2 för detaljer):

- Mittpunktskoordinater på aktuell yta.
- Lutning på ytan runt x-, y- och z-axeln.
- Objektets mått, det vill säga dess bredd, längd och höjd. Detta för att underlätta vid placering av upplockat objekt.
- Avlämningskoordinater.

PLC:n skickar resultaten en och en och inväntar signalbyte på en av utgångarna i RobotStudios. När en boolean i RobotStudio har ändrats från låg till hög så har roboten utfört den programmerade rörelsen. Innan nästa skickas från PLC:n så adderas avlämningkoordinatens Y-värde med en kombination av det föregående objekts bredd i Y-led

och nuvarande objekts bredd i Y-led delat med två. Detta görs för att få ut korrekt avlämningsposition så att två objekt läggs tätt intill varandra. När roboten har hämtat och lämnat alla objekt så skickar PLC:n återigen instruktion till Cognex Designer för att trigga kameran och Tasken igen. Det just beskrivna händelseförloppet loopas sen om igen.

6 Slutsats

I detta avsnitt återges först en redogörelse av problemformuleringen, en reflektion kring framtida utvecklingsmöjligheter samt kring etiska aspekter.

6.1 Reflektion av problemformulering

1. *Hur används programvaran Cognex Designer för att få ut rätt parametrar från bilderna som blir tagna av 3D-kameran?*
   Det finns flertalet sätt att göra detta på beroende på vilket/vilka verktyg som används, se möjliga alternativ i sektion 6.2.
   I nuvarande version hämtas parametrar från bounding box på det hittade objektet.

2. *Hur överförs koordinater och annan information till PLC styrsystemet och hur kommunicerar det i sin tur med roboten?*
   PLC:n kommunicerar med Cognex Designer via TCP/IP protokollet och via profinet protokollet med ABB-roboten, för mer detaljer se sektion 4.2.2 angående TCP/IP eller 4.2.4 angående profinet.

3. *Hur hanteras objekt som har inte har samma höjd över hela föremålet?*
   Utifrån hörnpunkterna på ytan så beräknas vinklar i rymden ut som sedan används för att vinkla robotens plockverktyg vinkelrätt mot lådans mittpunkt där den sedan plockar.

4. *Hur urskiljs objekt som är staplade exakt på varandra?*
   Dessa urskiljs inte med en bild, här hittas det översta objektet först och plockas därefter tas en ny bild för att se om det finns ett objekt under. Om det är så att det finns ett objekt under plockas även detta objekt se sektion 5 för triggersekvens.

5. *Hur hanteras objekt som ligger delvis på varandra?*
   Detta fall hanteras på samma sätt som ovanstående problemformulering 4.

6. *Hur urskiljs objekt som ligger tätt intill varandra, så det inte uppfattas som ett stort objekt?*

   Detta problem har ej lösts med den nuvarande lösningen, PatMax 3D Tool kunde inte urskilja objekten åt när den letade mönster. Detta tros vara till följd av att kamerans upplösning inte var tillräckligt bra.

6.2 Utvecklingsmöjligheter

Det finns stort utrymme för utveckling när det gäller den nuvarande lösningen. Systemet är som konstaterat begränsat till intränade rätblock. Om det istället skulle utgöras av exempelvis cylinderformade objekt så hade en ny lösning behövts implementeras. Med det sagt så finns det i huvudsak två alternativ angående hur projektet bör fortskrida vid fortsatt arbete med Cognex Designer. Det första alternativet är att fortsätta använda PatMax 3D Tool och det andra alternativet är att göra en egen mjukvara till att känna igen objekt med hjälp av Cross Section Tool.

Vid vidare användning av PatMax 3D Tool är systemet fortsatt beroende av inträning av objekt men å andra sidan finns det redan ett system som gått att bygga vidare på och modifiera. Om valet faller på att använda Cross Section Tool som huvudverktyg finns det möjlighet att utveckla ett system som är oberoende från inträning av mönster kopplade till särskilda objekt. Det vill säga att mönsterigenkänning inte längre är en del i systemet. Det andra alternativet har möjlighet till att vara mer flexibelt än det första alternativet men kräver avsevärt mycket mer arbete och det finns dessutom inga garantier för att det fungerar att använda Cross Section Tool med samma säkerhet som PatMax 3D Tool.

6.2.1 3DPatMax Tool

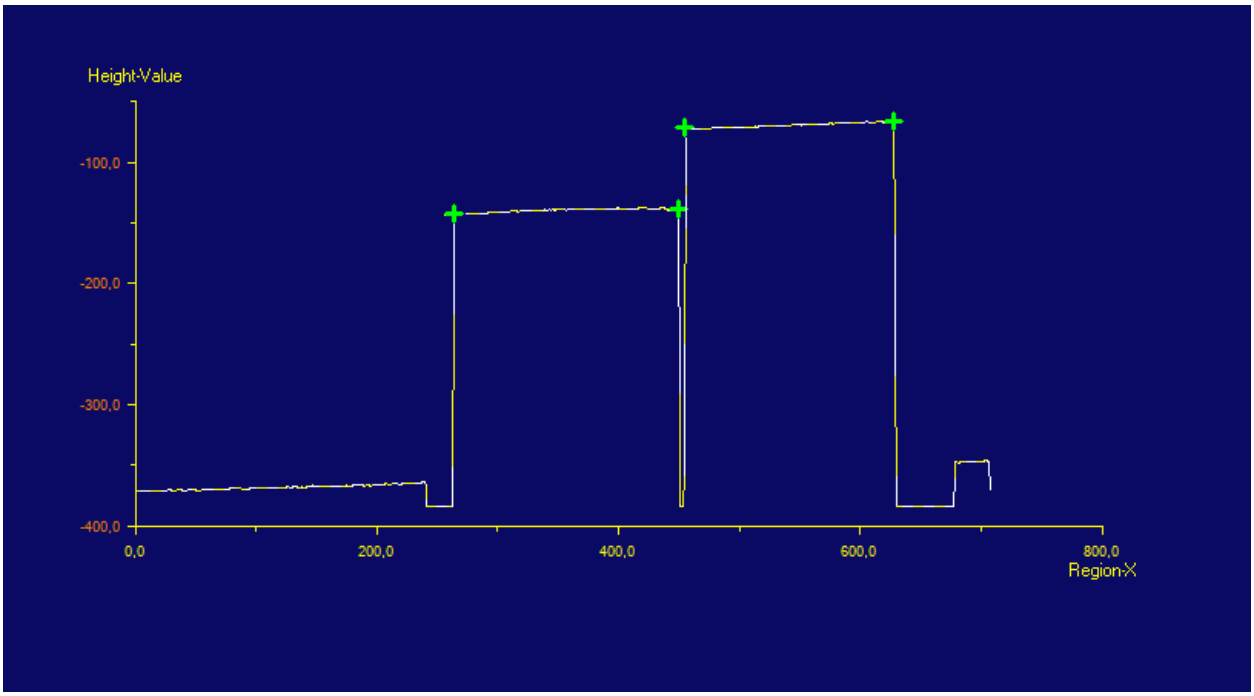
En möjlighet som tidigt fanns i åtanke var att kombinera 2D-verktyg med PatMax 3D verktyget. Nackdelen med PatMax 3D Tool är att det inträna mönstrets storlek varken kan skalas upp eller ned, utan det förblir ett fixt mönster som endast kan översättas till andra platser i koordinatsystemet med ändrad vinkelorientering. Ett alternativ är att inkorporera ett 2D träningsverktyg som har egenskapen att få ett 2D-mönster som kan skalas upp och ner på både X-axeln och Y-axeln. På detta sätt kan kravet på lådans mått förbises så länge det

fyrkantiga mönstret upprätthålls. Väsentlig 2D-data kan sedan extraheras från den funna ytan som bredd och längd. Eftersom det är möjligt att träna in objekt under runtime så kan 2D-ytans egenskaper nyttjas till träning av ett PatMax 3D Tool, där X och Y kan begränsats av den aktuella 2D-ytan. Detta kan resulterat i ett lite mer flexibelt system där objektets mått inte har någon betydelse då 2D verktyget gör det möjligt för skalning av det inträngade mönstret. Att träna in objekt tar dock relativt lång tid om det görs manuellt, träning under runtime kan därför påverka exekveringstiden negativt.

6.2.2 Cross Section Tool

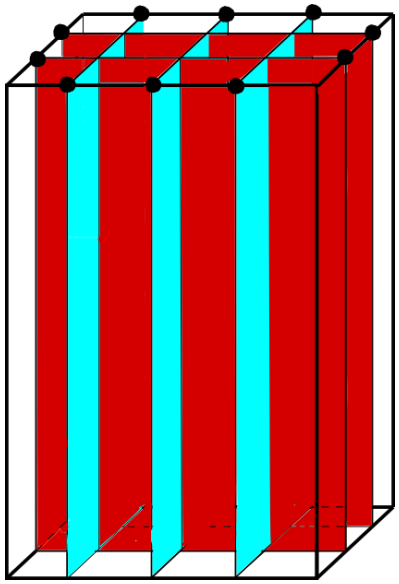
Den alternativa lösningen som nämns är att använda Cross Section Tool som huvudverktyg istället för PatMax 3D Tool för bildbeartningen. Teoretiskt sätt så går det att analysera objektets 2D-profil, och med hjälp av tillhörande operatorer få ut relevant data som beskriver objektet. Detta begränsas dock till objekt med tydliga kanter, vilket denna lösning förhåller sig till.

Funktionaliteten går ut på att göra ett smalt tvärsnitt i riktning med x-axeln som sträcker sig över hela bilden, med en början längst ner i bilden där Y är som lägst, och ständigt gör nya tvärsnitt med förhöjt Y-värde tills hela bilden har granskats. Efter att alla horisontella tvärsnitt är gjorda så utförs samma iteration igen fast vertikalt längs y-axeln. Hittar verktyget något i den tagna bilden vid det aktuella y- eller x-värdet så är tanken att profilen analyseras. Med hjälp av CogIPOneImageTool som sätter alla saknade pixlarna till det minsta Z-värdet i bilden, så fås en 2D profil likt detta.



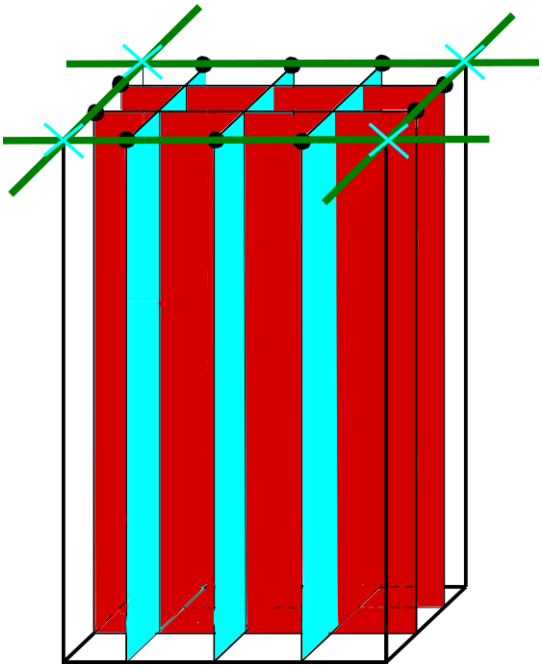
Figur 16. 2D-profil på två närliggande lådor med hörnorna markerade av ExtractCorner

Till följd av de tydliga kanterna som bildas på objektet av verktyget så kan ExtractCorner operatorns algoritm känna av hörnpunkterna av profilen, som i sin tur är punkter på sidan av objektet. Ställs det in så att punkterna indexeras från exempelvis vänster till höger så är det känt att index 0 och index 1 tillhör samma objekt, om de har samma Z-värde, och kan därför skiljas åt vid beräkningar. Har tvärsnitt utförts som beskrivet så skulle resultatet av punkter sett som i figur 17, i ett 3D-perspektiv.



Figur 17: Iteration av tvärsnitt på låda

Med dessa punkter går det sedan att beskriva hur objektet är utformat. I fallet av en rektangulär låda, som har varit testobjekt för projektet, så är det möjligt att utifrån punkterna skapa räta linjer. Därefter kan skärningspunkterna för de fyra linjerna bli beräknade, där skärningspunkterna motsvarar hörnen på objektet (Se figur 18). Efter detta steg så kan samma beräkningar göras som vid PatMax 3D Tool lösningen för att få ut centrumkoordinater och lutningar.



Figur 18: Linjer utformade från punkter med skärning i hörnpunkterna

6.2.2 Jämförelser

Med den upplösning som 3D-A5120 kameran hade så visade det sig att 3DPatMax Tool var inkonsekvent med att hitta rätt mönster. Särskilt när två objekt låg tätt intill varandra, då verktyget såg det som ett sammanfogat stort objekt. Den positiva aspekten med den alternativa lösningen är att Cross Section Tool skapar en mer detaljerad återgivning av objektet. I scenariot med två objekt tätt intill varandra så syns det i profilen att det är två objekt till följd av dalen som bildas. Det behövs inte heller träning av objekten med Cross Section Tool, vilket medför att det inte finns några restriktioner på objektets mått. Detta leder i sin tur till att den inte är lika känslig för deformationer.

Exekveringstiden är en viktig aspekt för många vision-applikationer, och eftersom det är en oprövad lösning så är exekveringstiden till stor del okänd. Körningstiden för Cross Section verktyget en gång är cirka 100 ms, men hur många iterationer som krävs för att granska hela bilden är inte testat nog för att få fram en exakt siffra. Jämförelsevis så är 3DPatMaxTool beroende av hur många datapunkter den har att analysera, det vill säga hur många objekt som är närvarande i bilden. En bild med tre objekt tar cirka 5000 ms för de tre PatMax-verktygen att hitta mönsterna. Detta kan betyda att den alternativa lösningen är för ineffektiv för att användas i en verklig situation, beroende på hur många gånger verktyget behöver köras.

6.3 Reflektion över etiska aspekter

För att lösningen ska ha någon samhällsnytta så behövs optimering och utveckling av nuvarande lösning. Exekveringstiden är för tillfället för hög för att användas i ett automatiserat system.

Om programmet skulle vidareutvecklas skulle det kunna bidra till att effektivisera produktionen för vissa automatiserade företag. Samtidigt skulle det kunna spara in pengar för företaget eftersom företaget hade kunnat ersätta arbetande personal med maskiner och kameror. Vid effektiviserad automation till följd av vision hade eventuella företag kunnat spara in pengar till följd av reducerad mänsklig arbetskraft samt generera mer pengar då robotar inte har behov som att vila eller äta. Ett företag som genererar mycket pengar och går med vinst kommer, om marknaden tillåter, vilja expandera vilket i sin tur kommer resultera i nya arbetsmöjligheter i form av exempelvis försäljning, human resources, underhåll av maskineri och driftpersonal.

Nackdelarna med att robotar, tillsammans med vision, ersatt mänsklig arbetskraft är att den personal som förlorat sina arbetsuppgifter kommer att behöva nischa om sig eller återutbilda sig. Att individer som arbetat för företaget tidigare blir tvungna att hitta nytt jobb eller gå i utbildning hade kunnat leda till att påverka inte enbart individen men också dessa eventuella familj, särskilt om denne varit ensamstående förälder. Alltså skulle detta program kunna resultera i fler jobb men det skulle även kunna bidra till att de som inte har vissa utbildningar blir av med sina nuvarande jobb eftersom maskinerna tar över deras arbetsuppgifter. Exempel på fall då robotar skulle kunna ersätta människors arbetsuppgifter är när det finns två lastpallar som är lastade till hälften med produkter. I detta fall hade det kunnat vara önskvärt att lägga ihop produkterna på en pall för bättre utnyttjande av lagerutrymme eller för bättre packning av lastbilar vid transport. I detta fall hade vision system kunnat ersätta personal då 2D-vision inte hanterar höjd och hur lastpallar är packade. Om programmet skulle vidareutvecklas skulle det kunna bidra till att effektivisera produktionen för vissa automatiserade företag. Samtidigt skulle det kunna spara in pengar för företaget eftersom företaget hade kunnat ersätta arbetande personal med maskiner och kameror.

7 Terminologi

CogImage16Range - Ett 3D-bildformat i Cognex Designer

Bounding Box - rektangulär volym inneslutande det inträna mönstret

Datameddelande - Meddelandet som skickar resultatsträngen som innehåller all plockdata.

Enumerator – Ett objekt som kan iterera genom en samling av icke-generiska typer.

Felaktigt Mönster - Funna ytor som PatMax3D Tool tror är korrekt utifrån inträna mönster, men som är felaktigt matchade.

Längdmeddelande - Ett meddelande som anger hur många tecken resultatsträngen innehåller.

Position - En punkt i rymden.

Scripting - Programmering i form av text

Task - Ett område i Cognex Designer där projektdata och tools skapas och hanteras.

Tool - Ett verktyg inkluderat i Cognex Designers bibliotek

8 Källförteckning

[1] ABB automationsföretag. *IRC5 Industriell Robot Controller.* 2019–06 https://search.abb.com/library/Download.aspx?DocumentID=ROB0295EN&LanguageCode=en&DocumentPartId=&Action=Launch

[2] Bolton och William. 2009. Programmerbara Logic Controllers. 5:e upplagan. https://www-sciencedirect-com.ludwig.lub.lu.se/book/9781856177511/programmable-logik-controllers

[3] Cognex Corporation. 2018. *VisionPro snabbreferens 9.6.* https://support.cognex.com/docs/vpro_960/EN/VisionProQuickReference.pdf

[4] Cognex Corporation. *Cognex Designer Release Notes 4.3.1*. 2020-02-03. https://support.cognex.com/docs/designer_431/SV/Designer_Manual_RN.pdf

[5] Cognex Corporation. *3D-A5000-SERIEN OMRÅDE SCAN 3D-KAMERA SPECIFIKATIONER.* https://www.cognex.com/products/machine-vision/3d-area-scan-cameras/3d-a5000-series-area-scan/specifications (2020-08-28)

[6] Cognex Corporation. 2020-01-31. *Visionpro Dokumentation.*

[7] James F. Kurose & Keith W. Ross, Datornätverk*, En top-down-strategi*, 2017, 7. Uppl, Pearson Utbildning Begränsad, Edinburgh Gate.

[8] J. Månsson & P. Nordbeck, *Flerdimensionell Analys*, 2018, Uppl 1:6, Studentlitteratur AB, Lund.

[9] Microsoft Corporation. *Dynamisk-Link-bibliotek.* 2018-05-31. https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-libraries