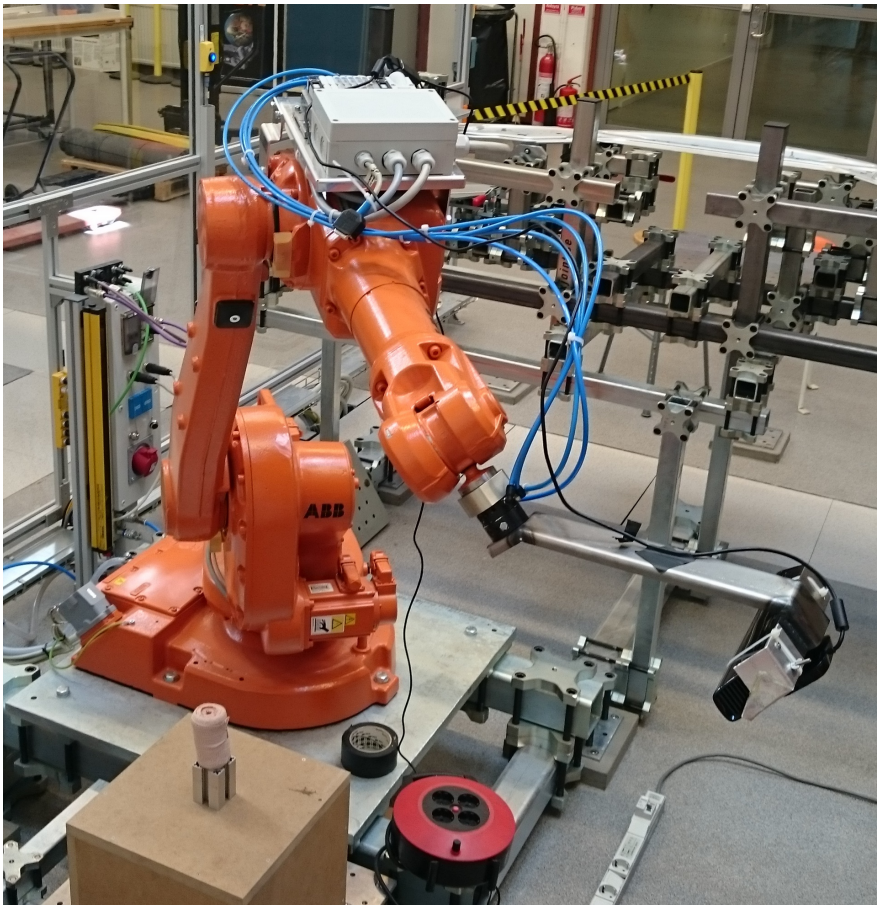
CHALMERS



Kopieringsfabriken

Automatisering av 3D-skanning och 3D-utskrift

Kandidatarbete på Instutionen för Produkt- och produktionsutveckling

Per Andersson, Matilda Halldén, Christoffer Hildebrand,
Carl Hornborg, Daniel Pantzar och Josef Östling

Instutionen för Produkt- och produktionsutveckling
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2015

Kopieringsfabriken
Automatisering av 3D-skanning och 3D-utskrift

PER ANDERSSON

MATILDA HALLDÉN

CHRISTOFFER HILDEBRAND

CARL HORNBORG

DANIEL PANTZAR

JOSEF ÖSTLING

Instutionen för Produkt- och produktionsutveckling.
Chalmers Tekniska Högskola 2015

Division of Product and Production Development
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Omslagsbild:
Carl Hornborg

Kandidatarbete 2015

Kopieringsfabriken

Automatisering av 3D-skanning och 3D-utskrift

Per Andersson, Matilda Halldén, Christoffer Hildebrand,
Carl Hornborg, Daniel Pantzar och Josef Östling

CHALMERS

Instutionen för Produkt- och produktionsutveckling
Chalmers Tekniska Högskola
Göteborg, Sverige 2015

Förord

Denna rapport är resultatet av ett kandidatarbete vid *Institutionen för Produkt- och produktionsutveckling (PPU)* vid Chalmers Tekniska Högskola. Kandidatarbetet genomfördes under våren 2015 av sex studenter i årskurs tre från programmet Automation och mekatronik.

Vi vill tacka vår handledare Per Nyqvist som har guidat oss genom projektet med god mejlkontakt och snabb respons. Vi vill även tacka Jon Andersson och Jonatan Berglund som bollade ideér med oss i början av projektet.

Tack till PPU för att vi har fått en plats att arbeta på, och för utrustningen vi har fått låna.

Till sist vill vi även tacka *Avdelningen för fackspråk och kommunikation* som har väglett oss genom rapportskrivandet.

Göteborg, 2015-05-19

Per Andersson, Matilda Halldén, Christoffer Hildebrand, Carl Hornborg, Daniel Pantzar, Josef Östling

Sammanfattning

I dagsläget är kopiering av fysiska objekt inget vem som helst kan utföra utan kunskap om de moment som ingår. Kopiering av objekt kan vara till stor hjälp när till exempel delar av maskiner går sönder och reservdelar behövs. Möjligheten att kopiera en del istället för att behöva beställa en ny skulle kunna spara mycket tid.

Projektets mål har varit att arbeta fram en helt automatisk kopieringsprocess av fysiska objekt. Utskriftens kvalitet har inte varit i fokus i detta projekt. Exempel på kopieringen finns i bilaga D.

Projektet har använt en ABB IRB-1600 industrirobot, en Microsoft Kinect och RepRapPro Ormerod 1 3D-skrivare för att göra kopieringen möjlig. Processen är indelad i styrning av robot, 3D-skanning, behandling av punktmoln, utskrift av 3D-modell och sammanlänkning av dessa delar.

Projektets resultat visar att det är möjligt att kopiera fysiska objekt automatiskt. För att ta projektet vidare och förbättra resultatet föreslås att fokus läggs på framtagningen och behandlingen av punktmoln. Detta eftersom ett punktmoln med för få punkter och för mycket brus är svårt att bearbeta och har varit ett problem under projektet.

Abstract

It is difficult today to copy physical objects without proper knowledge about the included elements. Being able to copy an object can for example be of great help when a part of a machine breaks, and a spare part is needed. Being able to copy a whole part instead of ordering one could save a lot of time.

The objective of this project has been to create a fully automatic process for copying physical objects. The quality of the printed object has not been focused on.

To make the copying possible, an ABB IRB-1600 industrial robot, a Microsoft Kinect and a RepRapPro Ormerod 1 3D-printer has been used. The process is divided in to controlling the robot, 3D-scanning, processing of point clouds, printing of 3D-models and linking the parts of the process together.

The results of the project shows that it is possible to automaticly copy physical objects. To further improve the project, it is recommended to focus on the acquiring and processing of point clouds. This is because a point cloud with few points and noise is hard to process and has been a problem during the project.

Ordlista

3D-skrivare	Skrivare för att skriva ut tredimensionella objekt
Batchfil [.bat]	En textfil som vid exekvering skriver sitt innehåll till kommandotolken
CAD	*Computer Aided Design* - Digitala ritningar och designer
FFF	*Fused Filament Fabrication* - Vanlig teknik för 3D-utskrift
G-kod	Lista med instruktioner till 3D-skrivaren
IR	Infraröd
IRC	*Internet Relay Chat*, chattserver med olika kanaler(rum)
Kommandotolk	Används för att köra MS-DOS-kommandon och andra datorkommandon i Windows konsol
Makerbot	Tillverkare av 3D-skrivare
MeshLab	Programvara för bearbetning och redigering av ostrukturerade triangulära 3D-maskor
Offline Programmering	Programmering av program i en virtuell miljö
Open source	Öppen källkod som innebär att ett programs kod finns tillgänglig och kan modifieras
Ormerod	Produktserie 3D-skrivare från RepRapPro
Punktmoln	Flera punkter i ett tredimensionellt kordinatsystem. De bildar ofta en modell av ett inskannat föremål i form av ett moln av punkter.
PYTHON	Programmeringspråk
RANSAC	Random Sample Consensus
RAPID-kod	Högnivåprogrammeringsspråk för att styra ABBs robotar
RobotStudio	Programvara från ABB för att virtuellt styra ABB-robotar
Skanndata	Data från skanning
Slicer	Program för att göra om 3D-modeller till g-kod
TCP	*Tool Center Point* - Punkten som all robotpositionering är relaterad till
Transformationsmatris	En matris som multipliceras med ett punktmoln för att molnet ska transformeras till ett annat kordinatsystem
xyz-fil	Textfil som innehåller tre vektorer med punkternas x-, y- och z-värde

Innehåll

1
Inledning

I dag är kopiering av fysiska objekt ingen enkel process, med flera moment som måste utföras. Objektet behöver skannas, punktmolnet behöver omvandlas till en 3D-modell som i sin tur behöver översättas till G-kod för 3D-skrivaren. Dessa moment kräver kunskap om processen och om efterbehandling av skanndatan. Om 3D-kopiering görs lika lätt att genomföra som kopiering av vanliga pappersdokument, skulle användarbasen växa. Det behöver därför undersökas till vilken grad det är möjligt att automatisera kopieringen, så att användaren behöver så lite tekniska förkunskaper som möjligt.

Möjligheterna med 3D-kopiering är många. Om en maskindel går sönder, kan företaget göra en kopia av en oskadad del istället för att behöva beställa och vänta på en ny del. Designmodeller kan formas av lera eller liknande, och sedan kopieras i plast eller metall.

1.1 Syfte

Syftet är att undersöka hur stor del av 3D-kopiering som går att automatisera. Detta för att göra 3D-kopiering mer tillgängligt.

1.2 Mål

Målet med projektet är att kopieringsprocessen ska ske automatiskt efter att användaren har startat kopieringen, utan att användaren ska behöva initiera de påföljande stegen. Det viktigaste är att få till en fungerande automatisering från början till slut. Sammanlänkningen mellan de olika stegen och automatiseringen av processen är det viktigaste i projektet.

1.3 Uppgift

Uppgiften är att framställa en automatisk kopieringsprocess med hjälp av en industrirobot. För att lösa detta har uppgiften delats in i mindre delar. Att lösa varje deluppgift perfekt finns det inte tillräckligt med tid för då projektet går ut på att få till en automatisk kopiering. Uppgiften ska lösas med hjälp av en 3D-skanner, en *IRB-1600* industrirobot och en 3D-skrivare. Roboten ska användas till för att hålla skannern och på så vis kunna ta fördel av robotens noggrannhet.

Kopieringsprocessen har delats upp i fem delar för att göra det tydligare vilka lösningar som behöver tas fram.

1.3.1 Styrning av robot

Den här deluppgiften går ut på att lösa hur robotarmen ska röra sig. Det finns olika alternativ, och det kan vara om roboten alltid ska utföra samma rörelsemönster eller om rörelsen ska anpassas efter skanningsobjektets storlek och geometri, alltså vara olika vid varje skanning. Beroende på hur roboten rör sig finns en risk för att krocka med objektet. Eventuella krockar kan undvikas genom att roboten får reda på storleken på objektet och därefter anpassar avståndet under skanningen. Om robotens rörelsemönster alltid är likadant kan krockar förmodligen lättare undvikas. Detta kan dock göra att skanningen blir sämre på grund av att avståndet till objektet blir för litet eller för stort, då objekten som ska skannas kan variera i storlek.

1.3.2 Skanning

Uppgiften här är att hitta en lösning på hur ett punktmoln kan genereras från ett objekt. Inom skanningen finns flera steg som måste genomföras. Skannern och roboten kan antingen startas oberoende av varandra, eller kommunicera under skanningen för att optimera processen. Därefter behöver skanndatan lagras, antingen på nätverket eller lokalt på datorn som styr kopieringsprocessen. Skannern eller roboten måste sedan meddela när skanningen är slutförd så att nästa steg kan påbörjas. Då inte skanndatans kvalitet kan kontrolleras ställer detta högre krav på själva skanningen. Om en kontroll ska kunna göras blir inte processen helt automatisk, men det kan vara nödvändigt att ha med för att inte skriva ut dåliga modeller.

Vilket underlag föremålet som skannas ska placeras på behöver också optimeras. Färger, mönster, geometrier och liknande skulle kunna fungera som referenspunkter, vilket kan underlätta för skannern. Vad som krävs av bakgrunden kan variera mellan olika skannrar.

Alla storlekar på fysiska objekt kommer inte att kunna skannas in, utan begränsas av robotens räckvidd. Räckvidden beror dels på robotarmens längd och från vilka avstånd 3D-skannern klarar att skanna.

1.3.3 Behandling av punktmoln

I den här deluppgiften ska punktmolnet omvandlas till en 3D-modell. Idag finns det många olika program som kan ta in punktmoln som sedan behandlas manuellt med inbyggda funktioner. I projektet måste punktmolnen behandlas på ett generellt sätt som ska passa för flera olika objekt. Metoder måste hittas för att automatiskt utföra behandlingen av punktmolnen. Behandlingen innebär olika typer av algoritmer så som beräkningar för att sy ihop flera punktmoln, få bort oönskat brus, beräkning av punkternas normaler samt rekonstruktion av ytan. Detta används sedan för att få fram en 3D-modell som kan skrivas ut. Att göra detta automatiskt i ett helt nytt program kräver mycket arbete och undersökningar. En variant är att kombinera de program som redan finns för att dra nytta av deras olika fördelar. En annan variant är att komma på en egen kombination av algoritmer och sedan applicera dessa automatiskt på punktmolnet med hjälp av MATLAB eller liknande program.

1.3.4 Utskrift av 3D-modell

Utskriftsdelen ska resultera i att 3D-modellen kan skrivas ut. Det här är ett område som inte kommer behandlas i första hand i detta arbete eftersom ett kandidatarbete på detta område har gjorts tidigare år [1]. I det redogjordes för hur man kan automatisera detta steg.

3D-skrivaren, RepRapPro Ormerod 1, begränsar utskriftens storlek till 200x200x200 mm så objekt som är större än dessa mått behöver i så fall förminskas innan utskriften görs. Därför kommer objekt som inte kan skrivas ut i förhållandet 1:1 att uteslutas.

Tillgång till en Makerbot 3D-skrivare fanns också, och den kan användas istället för Ormerodskrivaren om det anses vara mer lämpligt. Den Makerbot som finns tillgänglig har en utskriftsyta på 285x153x155 mm vilket är mindre än Ormerodskrivarens utskriftsyta.

1.3.5 Sammanlänkning

Den sista deluppgiften är att sammanlänka hela processen. För att hela processen ska vara automatisk krävs det att data skickas mellan de olika stegen utan inblandning från användaren. Datan ska skickas från skanningsprocessen till ett program som behandlar punktmolnet. När behandlingen av punktmolnet är klar ska datan skickas vidare för att skrivas ut.

1.4 Avgränsningar

Undersökningar om det är etiskt rätt att låta vem som helst kopiera fysiska objekt lätt och billigt, skulle kunna göras. Möjligheten att kopiera kan bidra till ökad piratkopiering, men också till att viktiga delar ersätts med plastkopior som eventuellt inte klarar samma slitage. Den här frågeställningen kommer dock inte undersökas, då det skulle ta mycket tid från huvuduppgiften i projektet.

Kvaliteten på slutprodukten är inte fokus för projektet men kan behandlas i mån av tid.

Automatisering av utskriften har undersökts av tidigare kandidatarbeten, och är ett problem som inte behöver lika mycket uppmärksamhet som resten av kopieringsprocessen. Därför kommer fokus inte läggas på detta.

En avgränsning är att de objekt som ska kopieras ej får överstiga dimensionerna 200x200x200 mm. Föremålet kan inte vara större än att det får plats för att skrivas ut av 3D-skrivaren.

Valet av 3D-skanner ska i första hand göras bland de 3D-skannrar som finns tillgängliga på Chalmers.

Alla typer av objekt kommer inte att kunna kopieras då till exempel blanka ytor och spretiga geometrier blir för komplicerade för skannern. Ihåliga objekt kommer

inte heller kunna skrivas ut ihåliga, då skannern endast ser utsidan. Denna typ av objekt kräver mer unik efterbehandling, och inte den typen av generell behandling som är tänkt i detta projekt.

2
Teoretisk bakgrund

2.1 Robot

Under skanningdelen av projektet användes en industrirobot. Roboten, en ABB IRB 1600-8/1,45, har en positionsrepeterbarhet på ± 0,05 mm [2]. Roboten har en räckvidd på 1,45 meter och en maxlast på 8 kg. Den har sex axlar som kan röras oberoende av varandra. Verktyg kan fästas på roboten genom att skruva fast verktyget i en fästplatta som i sin tur fästs på roboten med hjälp av tryckluft.

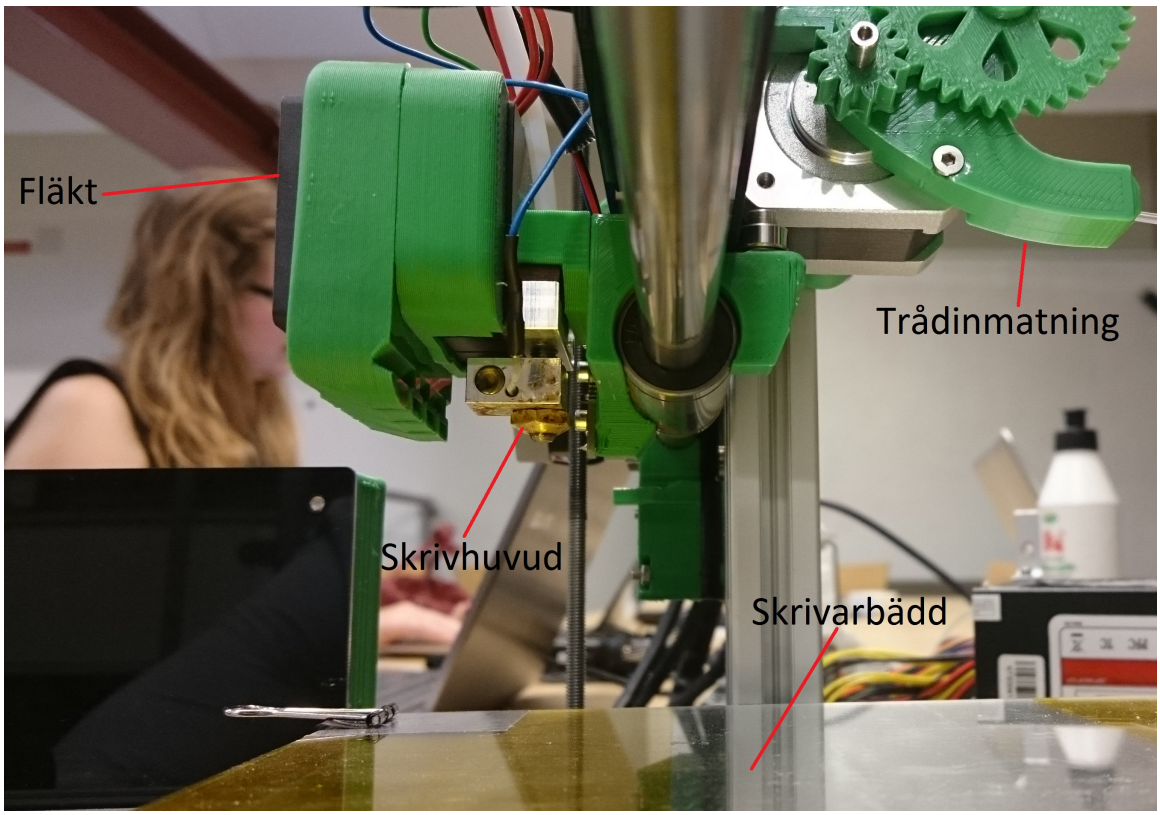
2.2 3D-skrivare

Under projektet användes 3D-skrivaren RepRapPro Ormerod 1, synlig i bild 2.1. Skrivaren är av typen FFF, Fused Filament Fabrication, och fungerar ungefär som en CNC-maskin, men tillför material istället för att ta bort [3]. För att skriva ut en 3D-modell behöver först 3D-modellen som skall skrivas ut, översättas till instruktioner som 3D-skrivaren förstår. Detta görs genom att 3D-modellen blir behandlad i en så kallad slicer. Slicern delar upp 3D-modellen i lager som skrivaren sedan kan skriva ut. Antalet lager beror på vilken lagertjocklek användaren väljer, samt hur högt objektet som ska skrivas ut är. När 3D-modellen är uppdelad i lager räknar sedan slicern ut vilka banor 3D-skrivarens skrivhuvud ska röra sig i för kunna bygga alla lager. Positionerna skrivhuvudet ska röra sig mellan översätts till instruktioner, G-kod, tillsammans med andra instruktioner användaren vill att skrivaren ska läsa. Instruktionerna kan till exempel ange lagertjocklek, temperatur på skrivarhuvudet och om skrivaren ska kalibrera sina axlar innan utskriften startas.

G-koden överförs sedan till 3D-skrivaren. De vanligaste sätten är via USB-kabel, via trådat/trådlöst nätverk eller via minneskort. När G-koden har skickats till 3D-skrivaren börjar skrivhuvudet och eventuell skrivarbädd värmas. När skrivhuvudet är varmt nog, matar 3D-skrivaren en tråd av termoplast, ofta PLA eller ABS, genom skrivhuvudet. Plasten smälter och blir sedan tryckt mot skrivarbädden, i det mönster slicern tidigare räknat ut. Därefter lyfts skrivhuvudet en lagertjocklek, och nästa lager skrivs ut. Så fortsätter det tills utskriften är färdig.

Samtidigt som skrivhuvudet matar ut smält plast, kyler en fläkt fäst på skrivhuvudet ner den plast som lagts ut. Detta gör att plasten stelnar snabbare vilket hindrar utskriften från att sjunka ihop. Kyls utskriften för fort stelnar och krymper de utskrivna lagerna innan tillräckligt många lager lagts ovanpå. Det kan leda till deformationer i utskriften eftersom krympningarna deformerar ytan skrivhuvudet skriver ut på. En uppvärmd bädd gör att utskriften inte släpper lika lätt från

bädden, vilket hindrar att utskriften rör på sig innan den är klar och därmed blir förstörd.



Figur 2.1: RepRapPro Ormerod 1

Andra välkända och väletablerade metoder för 3D-utskrift är SLM och SLA[4].SLM (Selective laser melting) skriver ut genom att en laserstråle bränner ett mönster i ett fint lager av metallpulver. När lagret är färdigt läggs ett nytt lager ovanpå som sedan nästa mönster bränns på. När alla lagers mönster har bränts, kan utskriften lyftas ur pulvret. SLA (Stereolithography) använder en laserstråle för att få flytande polymer att stelna. En skrivarbädd som kan flyttas i höjdled, placeras i en behållare av flytande polymer. En laserstråle belyser skrivarbädden och får lagermönstret att stelna. Därefter sjunker bädden en lagertjocklek och proceduren upprepas tills alla lager har skrivits ut.

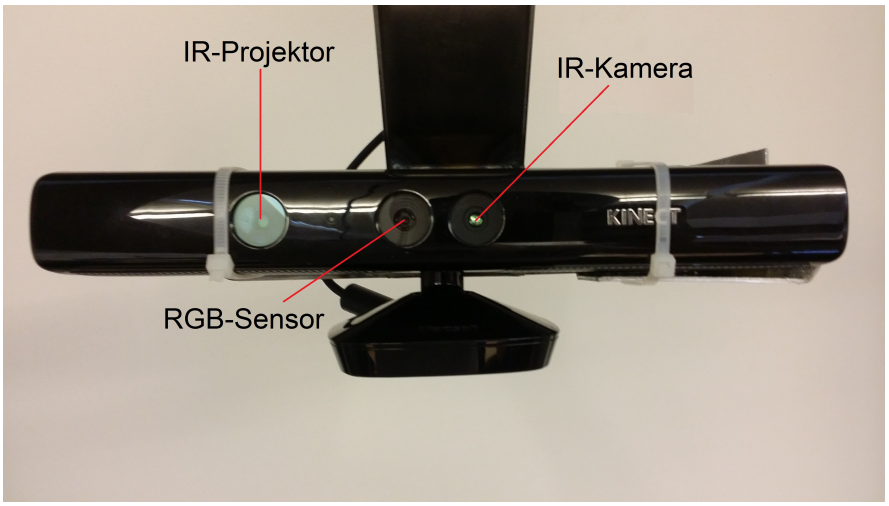
Vilka inställningar som är bäst för en specifik utskrift är svårt att veta i förväg. Därför krävs ofta flera utskrifter av samma modell med olika parameterval för att hitta de bästa inställningarna. Tjockleken på lagren är ofta mellan 0.1 och 0.3 mm. Ju tunnare lagren är, desto finare och mer exakt blir det utskrivna objektets geometrier. Tunna lager gör det lättare att skriva ut mer detaljerade objekt men det gör samtidigt att utskriften tar längre tid.

2.3 3D-skanner

En 3D-skanner använder en eller flera olika metoder för att skapa ett punktmoln av det skannern har framför sig. Hur datan för punktmolnet tas fram varierar beroende

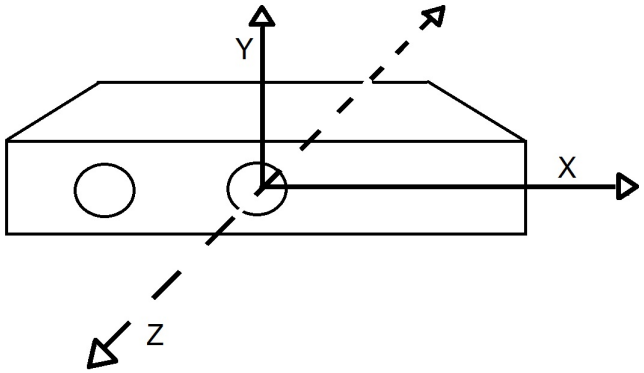
på vilken skanner som används. Ljus, laser och röntgen är de vanligaste metoderna.

Microsofts Kinect är en kamera som kan användas för 3D-skanning. Kameran kan ta bilder, både färg och djup. Kameran använder sig av IR-ljus för att kunna ta fram djup i en bild och upplösningen på djupbilden är 640x480 pixlar [5]. Detta ger ett punktmoln på cirka 300 000 punkter. Kameran har en IR-Projektor, en RGB-Sensor och en IR-Kamera enligt figur 2.2.



Figur 2.2: Microsoft Kinect

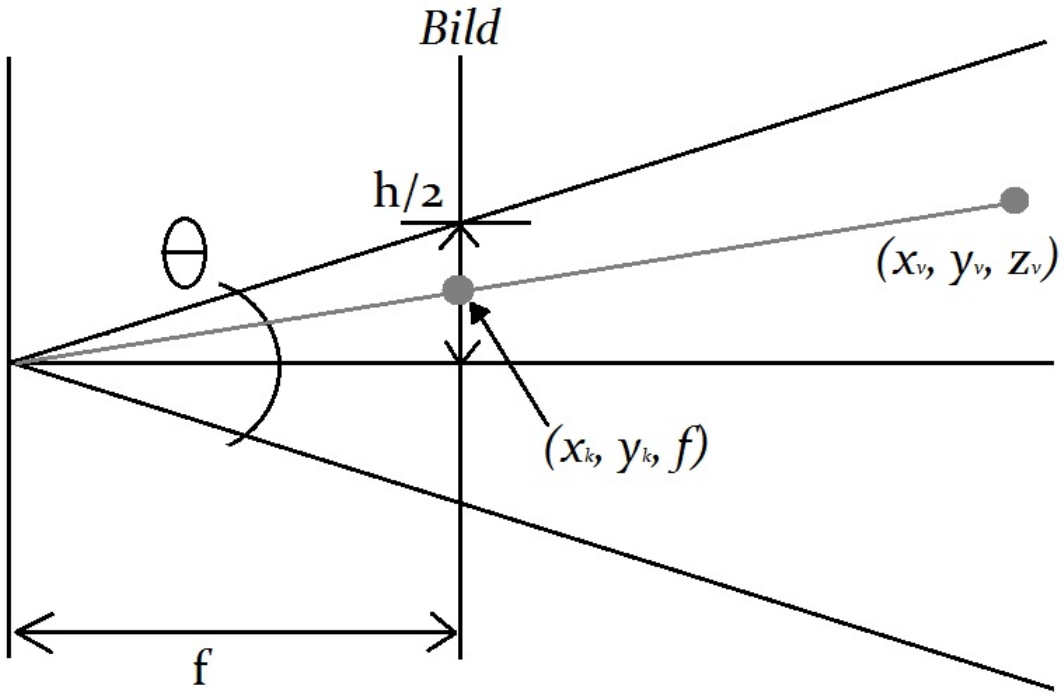
Origo är mitten på IR-Kamerans lins. Z-axeln pekar rakt ut mot objektet, y-axeln är vertikal och x-axeln är horisontell och bildar tillsammans ett koordinatsystem som ser ut som i figur 2.3. Koordinatsystemet är på detta sätt eftersom kameran oftast används för integration i dator- och tv-spel då en spegelvänd bild ska återges från kameran. Vid 3D-skanning vill en korrekt avbildning av verkligheten göras och då måste det tas hänsyn till speglingen i punktmolnsbehandlingen. Kameran uppfattar de fysiska objekten som finns inom området för sensorns räckvidd [5].



Figur 2.3: Kinects koordinatsystem

IR-Projektorn sänder ut en infraröd stråle som delas i flera strålar genom ett gitter för att skapa ett jämnt mönster av strålar på omgivningen [6]. Mönstret fångas upp av den infraröda kameran och jämförs mot ett referensmönster. Referensmönstret är lagrat i minnet på sensorn, och är ett plan som är taget på ett känt avstånd från sensorn. Detta ger ett punktmoln med 640x480 punkter, alltså punkter i alla kamerans pixlar.

Matematiska formeln för att beräkna fram x- och y-värdena på punkterna är enligt likheten av trianglar, se figur 2.4. ”k” står för kamerakoordinatsystemet och ”v” för världskoordinatsystemet. "h"är bildens bredd $h_x = 640$ respektive höjd $h_y = 480$ pixlar. Kamerans synfält är olika vertikalt och horisontellt. Vinkeln för synfältet är 57 grader i x-led och 43 grader i y-led [7]. z_v, y_k, och x_k är kända variabler ur punktmolnet. Detta ger två trianglar som kan användas för beräkning av de okända x- och y-värdena på punkterna i punktmolnet. Kamerans fokus är olika i x- och y-led eftersom bilden inte är kvadratisk. Det som ska tas fram är x_v och y_v. Med utgång från att (0,0) är uppe i bildens vänstra hörn kan beräkningar ske enligt 2.1 till 2.5:



Figur 2.4: Kameran sedd från sidan med y-axeln lodrätt och z-axeln vågrätt rakt ut ur IR-kamerans centrum

$$\tan\left(\frac{\theta}{2}\right) = \frac{\frac{h}{2}}{f} = \frac{h}{2f} \tag{2.1}$$

$$f_y = \frac{h_y}{2\tan\left(\frac{\theta_y}{2}\right)} = \frac{h_y}{2\tan\left(\frac{43}{2}\right)} \tag{2.2}$$

$$f_x = \frac{h_x}{2\tan\left(\frac{\theta_x}{2}\right)} = \frac{h_x}{2\tan\left(\frac{57}{2}\right)} \tag{2.3}$$

$$\frac{y_v}{z_v} = \frac{y_k}{f_y} \Rightarrow y_v = z_v\frac{y_k}{f_y} \tag{2.4}$$

$$\frac{x_v}{z_v} = \frac{x_k}{f_x} \Rightarrow x_v = z_v\frac{x_k}{f_x} \tag{2.5}$$

Dessa beräkningar utförs av MATLABfunktionen depthToPointCloud som introducerades i MATLAB 2014b.

Kinectkameran har begränsningar på vilket avstånd som den kan registrera punkter. Med standarinställningarna på Kinectkameran har kameran ett minsta avstånd på 0.8 m där den kan registrera punkter och ett maximalt avstånd på 4 m [8]. En inställning som kan utnyttjas för att kunna använda kameran på närmare avstånd är *NearMode*. I det läget kan kameran uppfatta punkter på ett minsta avstånd av 0.4 m.

Hur bra punktmoln som fås fram med den valda kameran beror till stor del på avståndet till det objekt som ska skannas [9]. Felet som uppstår ökar exponentiellt med avståndet. Det innebär att avståndet måste vara så litet som möjligt mellan kameran och objektet för att skannen ska bli så bra som möjligt. Ett avstånd på 0.5 m ger ett fel på cirka 5 mm med 95% säkerhet [9]. Punkternas position i punktmolnen kan alltså variera med 5 mm i x-,y- och z-led vilket gör att modellen inte kommer se ut precis objektet i verkligheten.

2.4 Mjukvara

Här presenteras de programvaror som används för att genomföra projektet.

2.4.1 RobotStudio

RobotStudio är ABBs simulations- och programmeringsmjukvara [10]. I RobotStudio kan en virtuell miljö som är en kopia av verkligheten byggas upp. På detta sätt kan programmet testas innan det körs i verkligheten. I denna miljö kan program göras och simuleras för att säkerställa att programmets rörelsebanor ser ut som det är tänkt och att inga kollisioner sker. Denna programmering görs på en PC som inte behöver vara ansluten till roboten. Detta sparar tid och är ett säkrare och snabbare sätt att programmera [11].

2.4.2 MATLAB

MATLAB är ett väldokumenterat program med stort användningsområde. Programmet har möjlighet att utföra komplicerade uträkningar och bygga upp program för att bland annat visualisera 3D-modeller [12]. MATLAB kan lösa ekvationssystem och har många verktyg för att göra matematiska program med bland annat iterativa metoder, linjär algebra och filtrering. MATLAB har också ett brett stöd för samverkan med andra programvaror och programspråk.

2.4.3 MeshLab

MeshLab är ett program för att bearbeta data från 3D-skanning [13]. I programmet finns det olika verktyg och filter för att bland annat redigera, behandla, inspektera och konvertera ett punktmoln som fås från 3D-skanningen. Meshlab är gratis, har öppen källkod och är relativt lättanvänt. I och med att det har en stor användarbas finns det gott om hjälp att få på forum och liknande. Den öppna källkoden gör det möjligt för användare att skapa egna modifikationer och hjälpmedel till MeshLab.

2.4.4 Slic3r

Slic3r är en slicer med öppen källkod som är lätt att använda, har en stor användarbas och är gratis [14]. I Slic3r kan skrivarhuvudets utskriftsbana simuleras och visualiseras. Detta för att ge användaren en bild av hur utskriften kommer gå till. Slic3r kan också uppskatta hur mycket material som kommer gå åt till utskriften, och hur hur lång tid den kommer ta. Slic3r har också inbyggt stöd för automatisering genom att vara styrbar med kommandon från kommandotolken.

2.4.5 Printrun

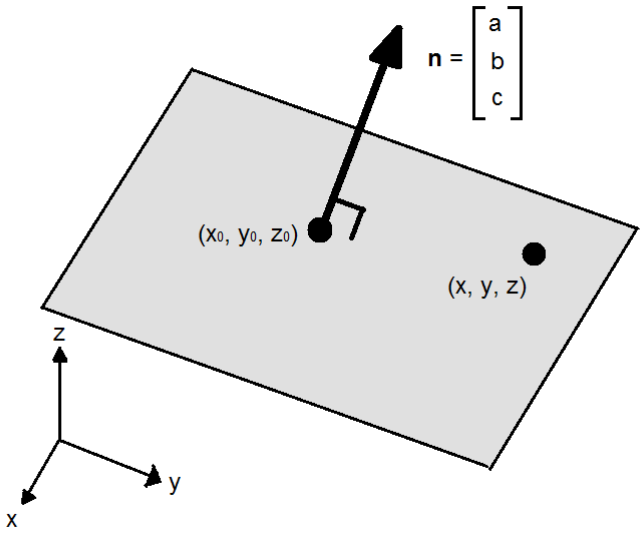
Printrun är ett programpaket [15] skapat av Kliment Yanev, och är gratis att använda. Printrun innehåller tre olika program; Printcore, Pronsole och Pronterface som erbjuder olika sätt att skicka G-kod till 3D-skrivare över USB. Från början fanns endast Printcore som är ett PYTHON-skript tänkt att köras genom kommandotolken. Efterföljaren Pronsole körs också genom kommandotolken, och har till skillnad från Printcore menyer och hjälpfunktioner. Pronterface är en utveckling av Pronsole och har ett grafisk användarsnitt som är mer användarvänligt.

2.5 Kalibrering av koordinatsystem

Varje punkt i ett punktmoln har ett x-,y- och z-värde i referens mot kamerans inbyggda koordinatsystem. En bild från kameran räcker inte för att skanna 360 grader av ett objekt, därför behövs det flera bilder från olika vinklar. För att punktmolnen från de olika bilderna ska kunna sammanfogas till en gemensam bild måste alla punktmoln finnas i ett gemensamt koordinatsystem. För att få fram ett gemensamt koordinatsystem måste det finnas en koppling mellan kamerans koordinatsystem och det tänkta gemensamma koordinatsystemet.

2.5.1 Planets egenskaper

Ett plan är uppspänt av två linjärt oberoende vektorer. Planets ekvation med en nollskild normalvektor n genom punkten (x_0, y_0, z_0) är $ax + by + cz + d = 0$ där $d = -ax_0 - by_0 - cz_0$ [18], se figur 2.5.



Figur 2.5: Planets egenskaper

Beräkning av variabeln d görs genom att ta $\bar{n}$ skalärt med punkten p enligt ekvation 2.6.

$$d = -\bar{n} * p \tag{2.6}$$

där är:

$$\bar{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

$$n_x = \frac{a}{\sqrt{a^2 + b^2 + c^2}}$$

$$n_y = \frac{b}{\sqrt{a^2 + b^2 + c^2}}$$

$$n_z = \frac{c}{\sqrt{a^2 + b^2 + c^2}}$$

och

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Normalisering av vektor utförs genom att ta vektorn och dividera med absolutbeloppet av vektorn enligt:

$$\hat{V} = \frac{V}{|V|} \tag{2.7}$$

Genom att lösa följande ekvationssystem hittas en gemensam punkt för tre plan:

$$\begin{cases} a_1x + b_1y + c_1z + d_1 = 0 \\ a_2x + b_2y + c_2z + d_2 = 0 \\ a_3x + b_3y + c_3z + d_3 = 0 \end{cases} \tag{2.8}$$

Ekvationssystemet löses genom att applicera minsta kvadratmetoden på $Ax = b$. Detta görs genom $x = A\backslash b = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$ där

$$A = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}, b = \begin{bmatrix} -d_1 \\ -d_2 \\ -d_3 \end{bmatrix}$$

Där den beräknade lösningen till ekvationen ger x-, y- och z-värdet till den gemensamma punkten för planen.

2.5.2 fitPlaneRANSAC

Funktionen fitPlaneRANSAC [16] används med några mindre modifikationer och returnerar en normalvektor och ett avstånd mellan två punkter. Inparametrarna till fitPlaneRANSAC är ett punktmoln, lösningens säkerhet i procent (konfidensintervall), maximala avståndet en punkt får befinna sig från det beräknade planet och den minimala distansen mellan de punkter som samplas fram.

Funktionen returnerar en normalvektor $n = (a, b, c)$ samt avståndet d från punkten $p_1 = (x_0, y_0, z_0)$, som normalvektorn går igenom, till punkten $p = (x, y, z)$. Den returnerar även hur många av punkterna i punktmolnet som ligger inom marginalen för det beräknade planet.

Funktionen slumpar först fram tre punkter, (p_1, p_2, p_3), ur planet som är tillräckligt långt ifrån varandra utifrån det angivna minimala avståndet [17]. Med dessa tre punkter beräknas normalvektorn till planet fram genom att ta kryssprodukten mellan $(p_1 - p_2)$ och $(p_1 - p_3)$, vilket ger normalvektorn $n = (a, b, c)$.

Funktionen beräknar avståndet D från en punkt $p = (x_0, y_0, z_0)$ till det beräknade planet $ax + by + cz + d = 0$ med ekvation 2.9

$$D = \frac{|a(x - x_0) + b(y - y_0) + c(z - z_0)|}{\sqrt{a^2 + b^2 + c^2}} = \tag{2.9}$$

$$= \frac{|ax + by + cz - ax_0 - by_0 - cz_0|}{\sqrt{a^2 + b^2 + c^2}} = \frac{|-d - ax_0 - by_0 - cz_0|}{\sqrt{a^2 + b^2 + c^2}}$$

Antalet punkter som är inom det maximala avståndet från planet hamnar i variabeln inliers. Iterationen utförs tills det önskade konfidensintervallet på lösningen uppnåtts. Ett högre konfidensintervall innebär att funktionen itereras fler gånger och då testar med flera olika punkter i kombination i samplingen. Ett lägre konfidensintervall ger färre iterationer men en snabbare beräkning. För varje iteration som ger fler inliers sparas de nya värdena för planets egenskaper. Efter modifikation ger funktionen även ut den punkten p_1 som används vid beräkning av d.

2.5.3 Transformationsmatris

För att transformera punkter från ett referenssystem, R, till ett nytt koordinatsystem, N, behövs en transformationsmatris. De tre första kolumnerna i ekvation 2.10 är det nya koordinatsystemets x-, y- och z-axel som enhetsvektorer och kallas tillsammans rotationsmatris. Den fjärde kolumnen är translationen av det nya systemets origo i referenssystemet. För att kombinera rotationsmatrisen och translationen i en matrismultiplikation läggs homogena koordinater till, vilket ger den fjärde raden. Hos robotar är den alltid [0 0 0 1] och nedan följer de matriser som används för beräkningar [19].

Transformationsmatrisen från det nya koordinatsystemet till referenssystemet fås genom:

$$^{R}T_{N} = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

För att transformera punkter till ett nytt koordinatsystem från referenssystemet måste matrisen inverteras enligt:

$$^{N}T_{R} = \begin{bmatrix} x_x & x_y & x_z & -\bar{p} * \bar{x} \\ y_x & y_y & y_z & -\bar{p} * \bar{y} \\ z_x & z_y & z_z & -\bar{p} * \bar{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

där

$$-\bar{p} * \bar{x} = \begin{bmatrix} -p_x x_x - p_y y_x - p_z z_x \end{bmatrix}$$

$$-\bar{p} * \bar{y} = \begin{bmatrix} -p_x y_x - p_y y_y - p_z y_x \end{bmatrix}$$

$$-\bar{p} * \bar{z} = \begin{bmatrix} -p_x z_x - p_y z_y - p_z z_z \end{bmatrix}$$

Inversen fås genom att ta transponeringen av rotationsmatrisen och projicera den negativa translationsvektorn på det nya systemets koordinataxlar genom skalärprodukt.

För att rotera en matris runt z-axeln förmultipliceras den med följande matris där θ är den önskade vridningsvinkeln.

$$Rot(z,\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

För att flytta ett system linjärt i koordinatsystemet kan transformationsmatrisen förmultipliceras med följande translationsmatris, där (p_x, p_y, p_z) är hur mycket förflyttning av origo som ska ske i x-, y- respektive z-led:

$$Trans(p_x, p_y, p_z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

2.5.4 Punktmolnsbearbetning

Brus i punktmoln är punkter som ligger utanför den önskade ytan. Punktmolnet kan även få en ojämn yta på grund av ojämn placering av punkterna. Detta är viktigt att behandla för att få ett punktmoln som sedan kan användas för utskrift. För att få ett punktmoln som är så exakt som möjligt kan metoder som till exempel att ta ett medelvärde av punkter inom ett område vilket ger en jämnare yta. I MATLAB finns funktionen pcdownsample. Funktionen tar ett punktmoln och delar in det i små kuber. Punkterna i dessa kuber blir till en punkt som är medelvärdet av dessa [20]. För att bli av med brus, som är punkter som ligger längre ifrån den önskade ytan, finns funktionen pcdenoise i MATLAB. Funktionen tar standardavvikelsen för närmaste punkterna hos alla punkter och använder det värdet som gräns för de punkter som räknas som brus [21]. Funktionen returnerar ett brusbehandlat punktmoln.

3
Metodutveckling & Genomförande

3.1 Skanning

Skanningen genomfördes med hjälp av en robot, en 3D-skanner samt en dator med ett antal programvaror.

3.1.1 Plattform för skanning

För att genomföra skanningen placeras objektet på en plattform. Plattformens utformning ska möjliggöra framtagningen av matriser för att kunna sammanlänka bilderna. Plattformen tillverkades som en trälåda där alla vinklar var 90 grader. Lådan fästes sedan på en ställning av metall som kunde skruvas fast i en perforerad metallplatta. Lådan kan på så sätt tas bort och ställas tillbaka på samma position gång på gång. Om lådan vid en inskanning skulle vara på en annan höjd än vad som är kalibrerat skulle oönskade delar kunna komma med i utskriften vilket inte är önskvärt.

Figur 3.1: Plattform med distans

För att undvika att brus från lådan kommer med i punktmolnet placeras objektet på en liten upphöjning som visas i figur 3.1. På så sätt kommer objektet en liten bit ovanför lådan och det blir lättare att klippa bort punkter som inte tillhör objektet.

3.1.2 Verktyg

En begränsning är att skannern är tvungen att vara minst 0,5 m från objektet. Detta för att objektets yta ska kunna uppfattas av skannern. Det kravet i samband med att robotens räckvidd endast är 1,45 meter gjorde att ett verktyg var tvunget att tillverkas.
Verktygets syfte är att göra robotarmens räckvidd längre så objektet som skannas in blir mer lättåtkomligt från alla vinklar. Verktyget är vinklat så att det är lätt för roboten att komma åt att skanna horisontellt vid objektets nederkant. Verktyget gör också att roboten inte kommer med på bilderna som tas.

Figur 3.2: Verktyget med Kinectkameran fäst på roboten

CAD-modellen av verktyget har använts vid simuleringar av programmen i RobotStudio. Detta gjordes för att säkerställa att inga krockar sker när programmet körs i robotcellen.

Verktyget tillverkades av 2 mm tjock plåt och klipptes ut till rätt storlek. Därefter borrades hål för att göra det möjligt att fästa den med robotens monteringshuvud. När hålen var borrade sågades en kil ut för att få till rätt vinkel. Plåten bockades först längs långsidan till 90 grader. Sedan bockades plåten längs kortsidan så att vinkeln blev 45 grader i båda ändar. Kilen som pressades samman svetsades för

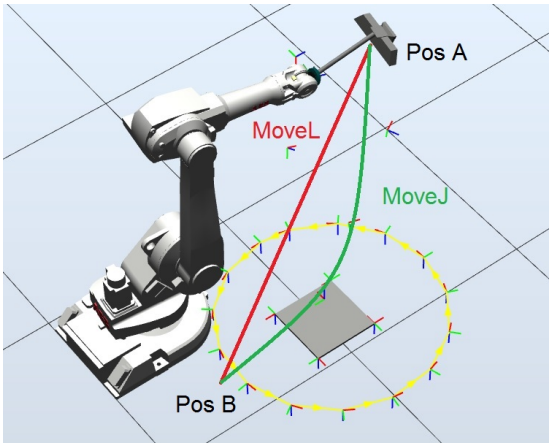
ökad stabilitet. En plåtbit svetsades fast på ena änden för att Kinectkameran enkelt skulle kunna fästas på verktyget.
En CAD-ritning av ett verktyg som fästs på roboten gjordes i Catia. Modellen av verktyget användes i RobotStudio för att användas vid simulationer.

3.1.3 Robot

Tre personer i gruppen fick en utbildning om roboten. I utbildningen gjordes ett enkelt program och säkerhetsföreskrifter för hur roboten ska användas gicks igenom.

ABB-roboten *IRB 1600* användes för att skanna in objekt. Med en Microsoft Kinect monterad på verktyget gjordes olika program för att skanna in på bästa sätt. I början av projektet var tanken att skannern ska filma i en svepande rörelse runt objektet. Då ingen metod för att göra detta hittades utvecklades istället ett program som tar bilder vid förutbestämda positioner med hjälp av MATLAB. Roboten kör alltså till varje position i tur och ordning för att skannern ska kunna ta bilder.

Kommandot MoveL [22] används för att röra roboten linjärt från en punkt till en annan. Problem uppstår när roboten inte kan röra sig linjärt mellan vissa positioner. Om roboten till exempel ska röra sig från punkt A till punkt B som i figur 3.3, kan det hända att roboten själv står i vägen för att röra sig i en linjär bana. För att undvika detta problem användes kommandot MoveJ [23] som gjorde att roboten kunde röra sig utan att problemet uppstod. MoveJ användes vid körning från en punkt till en annan då banan roboten rör sig i, inte har någon betydelse. När MoveJ används är alla axlar framme vid slutpositionen vid samma tidpunkt.



Figur 3.3: Simulering av skannerns positioner

3.1.4 Skanner

Ett antal olika 3D-skannrar undersöktes för att välja en som passar till projektet. Det finns många olika krav som skannrarna är tvungna att uppfylla vilket krävde att flera tester genomfördes. Om skannern har medföljande mjukvara, behöver mjukvaran utvärderas. Mjukvaran behöver vara kompatibel med övriga delar i systemet. Till en del skannrar fanns det inte mjukvara med öppen källkod och ibland innebar det att det blev svårt att få dessa att fungera med det övriga systemet. En annan egenskap som jämfördes var på vilket avstånd skannern kunde få ett bra resultat. Vissa skannrar får bättre resultat på längre avstånd och vissa på kortare avstånd till objektet. Ett viktigt krav är att skannern ska kunna fästas på roboten. Fästet tillsammans med skannerns vikt får inte överskrida robotens gränser. Skannern är också tvungen att kunna föras runt ett objekt med hjälp av roboten utan att kollidera med något under skanncykeln. Detta ställer krav på skannerns fysiska dimensioner. Den skanner som uppfyllde kraven och fanns tillgänglig valdes, alltså Microsoft Kinect.

3.1.5 Mjukvara

Det program som användes för att styra Microsoft Kinecten var MATLAB. Kod skrevs i ett skript för att skannern skulle ta flera bilder, bilda punktmoln av dem och sammanfoga dem till ett enda punktmoln. Då ett nytt tilläggspaket till MATLAB användes krävdes den senaste versionen av programmet vilken var R2015a.

3.2 Kalibrering av koordinatsystem

För att kunna göra en fullständig skanning och samla alla punktmoln i samma koordinatsystem genomfördes först en kalibrering av koordinatsystemet.

3.2.1 Kalibreringsposition

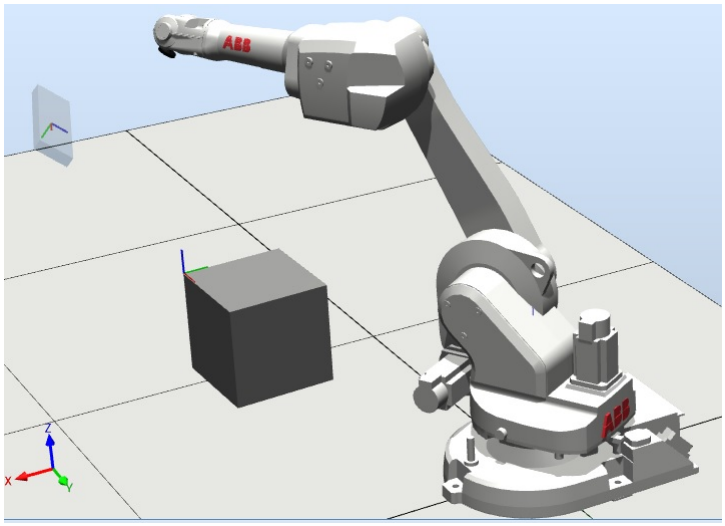
För att ta fram kalibreringpositionen kördes skannern först tätt intill lådans ena sida och robotens axlar justerades för att skannerns axlar skulle bli parallella med lådans. Skannerns mitt kördes sedan till ett av lådans hörn. Därifrån backades skannern från hörnet så att avståndet mellan lådan och skannern blev 70 cm. Denna position sparades undan för att användas i RobotStudio. I RobotStudio sattes ett TCP(Tool Center Point) i lådans hörn. Detta TCP vreds sedan 45 grader runt y-och z-axeln för att skannerns riktning skulle vara mot lådans hörn enligt figur 3.4.

3.2.2 Fotopositioner

För att ta fram åtta positioner för fotografering användes kalibreringspositionen som position för foto ett. De sju andra positionerna togs fram genom att flytta TCP i x- och y-led så att avståndet till lådans mitt är samma i alla positioner. TCP vreds sedan 45 grader runt z-axeln för varje ny position.

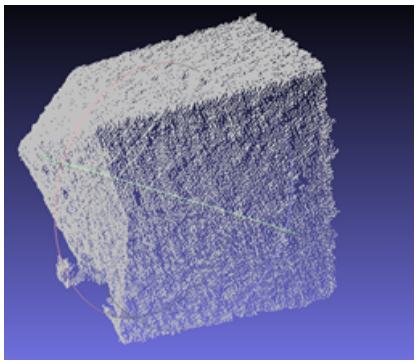
3.2.3 Transformationsmatris

Microsoft Kinect monterades på roboten och sattes i kalibreringspositionen. Sedan kopplades skannern ihop med MATLAB för att registrera ett punktmoln. Ur bilden



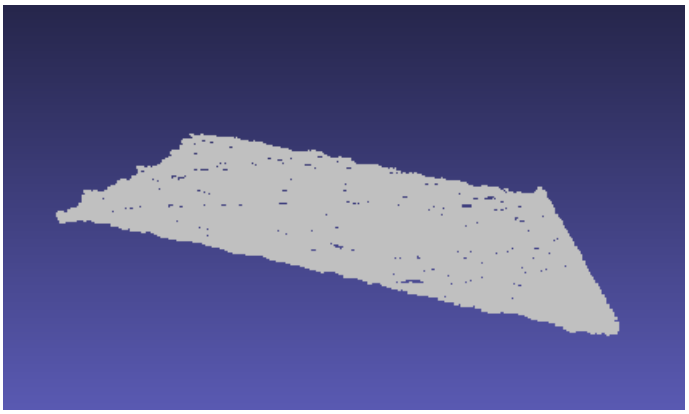
Figur 3.4: Framtagning av nytt koordinatsystem

klipptes punkterna på lådan ut manuellt ur punktmolnet och sparades i en xyz-fil, se figur 3.5.



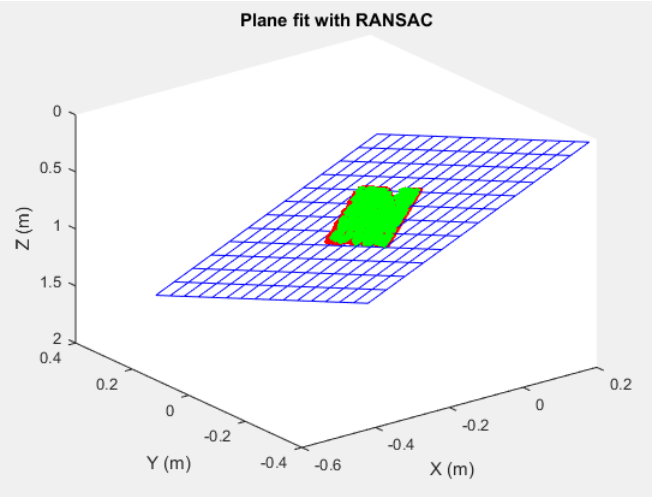
Figur 3.5: Låda urklippt ur punktmoln

Filen öppnades sedan i MeshLab och de tre planen på lådan skars ut, ett plan i taget, och sparades undan i separata xyz-filer, se figur 3.6.



Figur 3.6: En av lådans sidor urklippt ur punktmoln

I MATLAB öppnades en fil i taget och användes som input till funktionen fitPlaneRANSAC. Konfidensintervallet valdes till 99%, gränserna för punkter som tillhör planet valdes till 1 mm och minsta avståndet mellan de samplade punkterna valdes till 10 cm. Funktionen illustrerar det beräknade planet, se figur 3.7. Funktionen returnerar sedan värden för planets ekvation och hur många punkter som ligger i planet. Dessa värden sparas undan för vidare användning.



Figur 3.7: Planefit med RANSAC

Funktionen fitPlaneRANSAC användes på de tre planen från punktmolnet vilket gav tre stycken normaler och värden på planens avstånd d och punkter p_1. Normalen från övre planet gav z-axeln, normalen från vänstra planet gav x-axeln och normalen från det högra planet gav y-axeln för referensplanet. Normalens riktning kan vara åt två håll på planet. En kontroll av värdena på normalen utfördes för att se om den var riktad enligt det önskade referenssystemet. Om normalen var felvänd multiplicerades normalen med -1 för att vända den rätt.

Det framtagna koordinatsystemet var inte helt ortogonalt därför utfördes en justering. Normalen för det planet med flest punkter i, finns i det plan som är närmast det verkliga planet. De framtagna normalerna normaliserades enligt ekvation 2.13. Normalen ur det planet med flest punkter användes i kryssprodukt med normalen från det plan med näst flest punkter, vilket gav den tredje axelns normal. För att få alla axlar ortogonala mot varandra utfördes även en kryssprodukt mellan det bästa planets normal och den beräknade normalen vilket gör att alla tre axlar nu är ortogonala. Då normalerna räknats fram med hjälp av kryssprodukten, användes ekvation 2.6 för att räkna ut nya d för de två planens ekvationer. De tre planens ekvationer ställdes sedan upp enligt ekvation 2.8 och löstes för att få fram punkten där planen korsar varandra. Lösningen på ekvationssystemet samt de tre normalerna användes för att få fram en transformationsmatris enligt 2.10. Transformationsmatrisen inverteras enligt 2.11 för att transformera punktmolnet från skannerns koordinatsystem, R, till bordets koordinatsystem, N.

Transformationsmatrisen translaterades så origo placerades i mitten av lådan. För

detta flyttades x- och y-axeln 145 millimeter genom multiplikaton med 2.13. Transformationsmatrisen roterades 45 grader runt z-axeln genom att multiplicera med 2.12. Detta ger transformationsmatrisen för bild två. Matriserna sparas undan och detta upprepades för att få fram transformationsmatriserna för de övriga sju bilderna.

3.3 Behandling av punktmoln

Punktmolnen som tas fram under skanningen behöver sammanfogas, behandlas och filtreras för att till slut skrivas ut innan kopieringen är färdig.

3.3.1 Sammanfogning av punktmoln

De framtagna transformationsmatriserna multiplicerades med respektive punktmoln i MATLAB. Det avgränsade området skars ut ur punktmolnen. I z-led skars alla punkter ut inom 0-200 mm, i x- och y-led alla punkter mellan -145 mm och 145 mm. Alla moln plottades sedan i samma figur. Detta ger ett punktmoln där endast det skannade objektet finns med. För att se hur bra sammanfogningen blev skannades en träbit med kända mått in. Träbiten var placerad med ena punkten i origo och sidorna längs med x- och y-axeln. En kopia av träbiten tillverkades även i CAD med samma hörn i origo och samma sidor längs x- och y-axeln. En jämförelse mellan orginal (CAD) och kopia (skann) gjordes sedan i MeshLab.

3.3.2 Punktmoln till 3D-modell

Eftersom punktmolnet innehåller brus och ojämnheter filtrerades det med funktionen NoiseReduction i MATLAB. NoiseReduction använder pcdownsample och pcdenoise vilket resulterade i ett jämnare och brusreducerat punktmoln.

För att skanndatan ska kunna skrivas ut, måste hålrummen mellan punkterna i punktmolnen fyllas i. För detta ändamål valdes MeshLab. MeshLab har öppen källkod vilket gör att användare kan skapa egna modifikationer och tillägg. Ett sådant tillägg är *MultiMesh Scripting Tool* [24] som kan användas för att behandla ett punktmoln utan att användaren behöver starta varje delmoment manuellt. Den medföljande batchfilen kör ett skript där de operationer som ska utföras på punktmolnen står listade, tillsammans med sina parametrar.

MultiMesh Scripting Tool kommer med en lista på filter och operationer med passande parametrar enligt skaparen av skriptet. Då det dröjde innan projektet genererade egna punktmoln, testades filterskriptet på nedladdade punktmoln. Detta för att förbättra kunskapen om programmet innan projektet kommit tillräckligt långt. När projektet hade tillgång till egna punktmoln testades skriptet på nytt.

Flera av de egna punktmolnen hade för få punkter och/eller för mycket brus. Detta gjorde det svårt att hitta kombinationer av filter och parametrar som leder till bra modeller oavsett punktmoln. När filter valdes och testades efter ett visst punktmoln, gjorde det att filterskriptet ofta endast fungerade bra med just det punktmolnet, och sämre med andra. Efter många tester valdes till slut en kombination av filter och parametrar som gav ett godkänt resultat.

De filter och operationer som används är:

- **Borttagning av bord och distans**
  För att ta bort bordet och distansen ur punktmolnet, tas alla punkter med en z-position under distansens högsta punkt bort. Detta flyttar origo till objektets nya botten.
- **Compute normals for point sets**
  För att MeshLab ska veta vad som är in- och utsida hos punktmolnet, måste man beräkna punkternas normaler. Filtret tar som inparameter antalet grannar till varje punkt som skall användas för att räkna ut dess normal.
- **Poisson Disk Sampling**
  Skapar ett nytt lager av samplade punkter från det aktuella punktmolnet. Punkterna genereras enligt en Poission disk fördelning, med hjälp av algoritmen som beskrivs i *Efficient and flexible sampling with blue noise properties of triangular meshes.* [25].
- **Poisson Surface Reconstruction**
  Skapar ytor mellan punkterna vilket skapar 3D-modellen som sedan ska skrivas ut.
- **Borttagning av punkter med Z-koordinat < 0**
  Efter flera tester visade det sig att MeshLab ibland bygger ut 3D-modellen i negativ z-led. Varför detta sker är inte helt klart, men korrigeras genom att ta bort de punkter med z-värde mindre än 0.

Filternas inställningar finns i bilaga B.

3.4 3D-modell till G-kod

```
G21 ; set units to millimeters
M107
M190 S65 ; wait for bed temperature to be reached
M104 S200 ; set temperature
G28 ; home all axes
G1 Z5 F5000 ; lift nozzle

M109 S200 ; wait for temperature to be reached
G90 ; use absolute coordinates
G92 E0
M82 ; use absolute distances for extrusion
G1 F1800.000 E-1.00000
G92 E0
G1 Z0.350 F7800.000
G1 X94.818 Y96.794 F7800.000
```

Figur 3.8: Exempel på använd G-kod

För att 3D-skrivaren ska kunna skriva ut en 3D-modell, behöver modellen översättas till G-kod. Till detta används *Slic3r* genom att en av tre möjliga batch-filer startas. De olika batch-filerna innehåller sökvägen till Slic3r samt olika utskriftsinställningar, och väljs beroende av vilken typ av utskrift användaren vill göra. De inställningar användaren kan välja syns i figur 3.9
Utöver inställningarna innehåller batchfilerna också en sökväg till den 3D-modell som ska översättas till G-kod.

Val	Lagertjocklek [mm]	Fyllnadsgrad [%]
Snabb	0.3	10
Fin	0.1	10
Stark	0.3	90

Figur 3.9: Valbara utskriftskvaliteter

3.5 Utskrift

Utskriften startas genom att starta en batchfil som innehåller sökvägen till Printcore samt parametrar. De parametrar Printcore behöver är: G-koden, vilken USB-port på datorn som skrivaren är ansluten till, samt överföringshastigheten för G-koden.

3.6 Automatisering

För att de olika delmomenten ska kunna startas och samverka, behövs ett huvudprogram som har koll på hela kopieringsprocessen. Då skanningen och skapandet av punktmolnet görs i MATLAB, letades det efter ett sätt att utföra övriga moment i samma program. Översättning till G-kod och utskrift etc. går att utföra med hjälp av batchfiler, och det blev därför smidigt att låta MATLAB starta de olika batchfilerna när de behövs. För att göra koden överskådlig gjordes ett huvudprogram i MATLAB som innehåller flera olika skript och funktioner. Ett flödesschema för programmet finns i avsnitt 4.4.

4
Resultat

4.1 Kopieringsprocessen

Huvudprogrammet är skrivet i MATLAB och körs från en dator som är kopplad till roboten via nätverkskabel. För att roboten ska kunna utföra de kommandon huvudprogrammet ger den, körs en separat RAPID-kod hos roboten innan huvudprogrammet startas. När huvudprogrammet startas skickas en fil till ABB-robotens hårddisk vilket gör att roboten kör till start-positionen. Roboten letar efter filerna på sin hårddisk, och rör sig inte innan filerna finns där.

Första bilden tas efter en tidsfördröjning för att säkerställa att roboten är i position när bilden tas. MATLAB väntar sedan på att bilden har sparats och skickar därefter en ny fil till roboten. Roboten kör då till nästa position och proceduren upprepas. När samtliga bilder är tagna sammanfogas de till ett punktmoln som sparas. Därefter bearbetas punktmolnet i MeshLab och sparas som en 3D-modell.

När bearbetningen är färdig visas 3D-modellen i ett fönster. För att komma vidare behöver modellen godkännas av användaren. Godkänns modellen skrivs den ut.

4.2 Skanning

Här redogörs resultaten från kalibreringen samt punktmolnens precision.

4.2.1 Kalibrering av koordinatsystem

Efter att funktionen fitPlaneRANSAC räknat fram de tre planens ekvationer blev antalet punkter i planen:

inliers X = 16277
inliers Y = 16176
inliers Z = 24399

Funktionen fitPlaneRANSAC gav variablerna (a,b,c,d) för planen enligt:

$$X = \begin{bmatrix} 0.711099459301486 & -0.509798975760091 & 0.487143828562192 & -0.362442152411613 \end{bmatrix}$$

$$Y = \begin{bmatrix} -0.722177924586288 & -0.491965491555549 & 0.487009051743804 & -0.345297772244581 \end{bmatrix}$$

$$Z = \begin{bmatrix} 0.010283750134262 & -0.693324597361843 & -0.720565160705277 & 0.492805181499026 \end{bmatrix}$$

Punkterna p som användes vid beräkning av planets ekvation är:

$$p_x = \begin{bmatrix} -0.0300 & 0.1089 & 0.9020 \end{bmatrix}$$
$$p_y = \begin{bmatrix} 0.1224 & -0.0769 & 0.8130 \end{bmatrix}$$
$$p_z = \begin{bmatrix} 0.0117 & -0.2783 & 0.9520 \end{bmatrix}$$

En kryssprodukt av den normaliserade normalen från planet Z och normaliserade normalen från planet X gav normalen för Y. Kryssprodukten av framräknade normalen Y och Z gav en justerad normal för X.

Transformationsaxlarna är då:

$$\bar{x} = \begin{bmatrix} 0.710016 & -0.70411 & 0.010284 \end{bmatrix}$$
$$\bar{y} = \begin{bmatrix} -0.50234 & -0.51668 & -0.69332 \end{bmatrix}$$
$$\bar{z} = \begin{bmatrix} 0.493486 & 0.487101 & -0.72056 \end{bmatrix}$$

Lösningen på de tre planens ekvation gav origo i:

$$O = \begin{bmatrix} -369.1190 \\ -349.5630 \\ 492.8052 \end{bmatrix}$$

En kontroll av hur ortogonalt koordinatsystemet som räknades fram görs genom att ta skalärprodukt mellan systemets axlar. Resultatet är att systemet är ortogonalt då alla skalärprodukter är väldigt nära 0:

$$\bar{x} * \bar{y} = 1.4480 * 10^{-8}$$
$$\bar{x} * \bar{z} = 3.1626 * 10^{-8}$$
$$\bar{y} * \bar{z} = 5.5573 * 10^{-6}$$

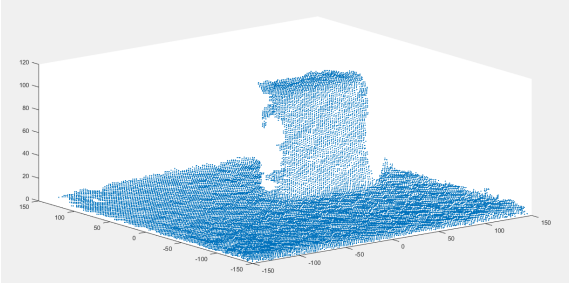
Efter flytt av origo till mitten av bordet med translationsmatrisen blev transformationsmatrisen:

$$T = \begin{bmatrix} 0,710016 & -0,50234 & 0,493486 & -514,119 \\ -0,70411 & -0,51668 & 0,487101 & -494,563 \\ 0,010284 & -0,69332 & -0,72056 & 492,8052 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

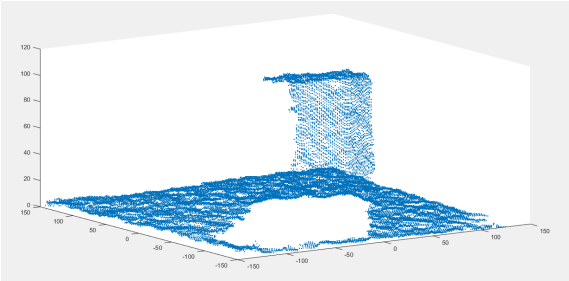
Transformationsmatrisen T används som T_1 och sedan roterades matrisen 45 grader runt z-axeln för att få nästa bilds transformationsmatris, efter upprepning fås alla åtta transformationsmatriser, se bilaga A.

4.2.2 Sammanfogning av punktmoln

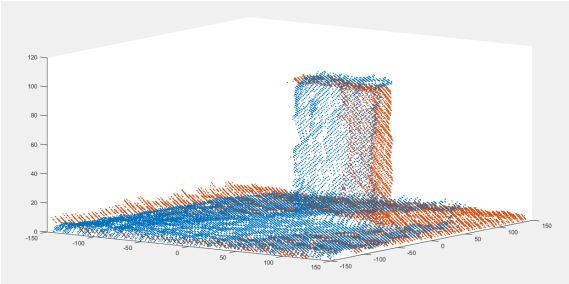
Här visas punktmolnet från bild 1 i figur 4.1 och bild 5 i figur 4.2, samt en sammansättning av dessa efter utförd transformering av punkterna i figur 4.3. Bild 1 och 5 valdes här för att visa två bilder som är vridna 180 grader från varandra runt z-axeln.



Figur 4.1: Skannpunkterna från bild 1



Figur 4.2: Skannpunkterna från bild 5

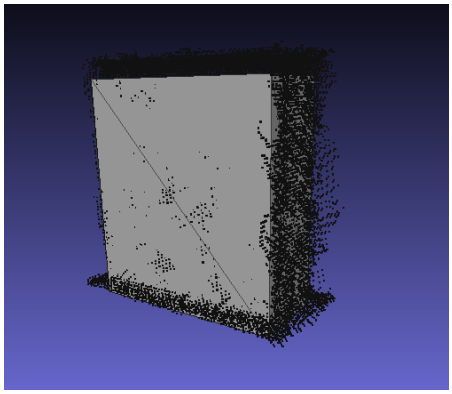


Figur 4.3: Skannpunkterna från bild 1 (blå färg) sammanfogat med bild 5 (röd färg)

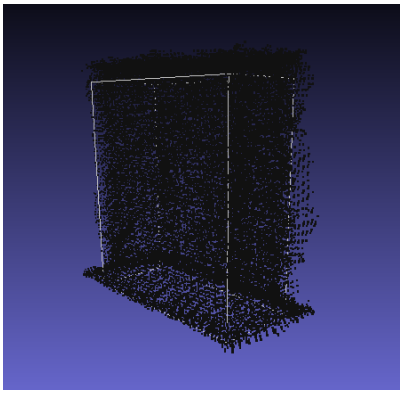
Det observeras en ojämnhet på objektets yta samt att det finns brus i form av punkter som ligger längre ifrån objektets yta.

4.2.3 Noggrannhet på punktmoln

För att visa resultatet jämförs det inskannade molnet med en identisk CAD-model av objektet. Efter inskanning innehåller punktmolnet mycket brus, se figur 4.4. Det sammansatta molnet består av 33295 punkter.

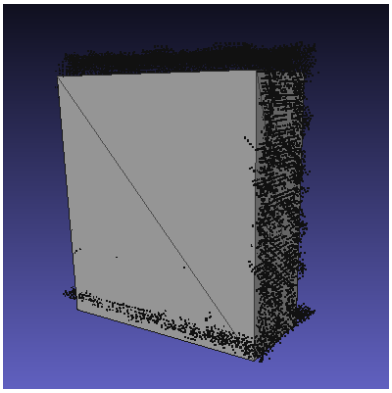


(a) Solid CAD-modell och obehandlat punktmoln

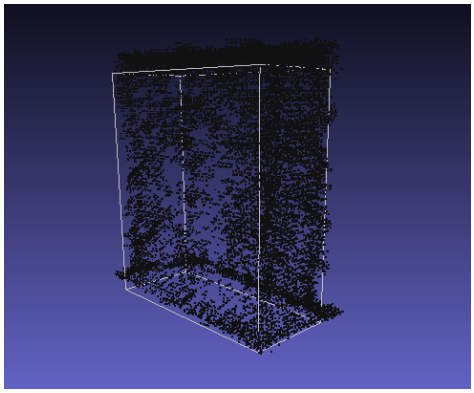


(b) Ihålig CAD-modell och obehandlat punktmoln

Figur 4.4



(a) Solid CAD-modell och behandlat punktmoln



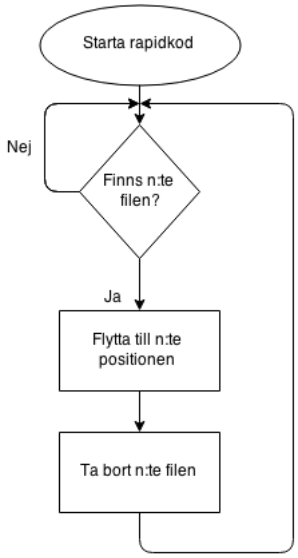
(b) Ihålig CAD-modell och behandlat punktmoln

Figur 4.5

Efter en punktmolnsbehandling med funktionen NoiseReduction innehåller det sammansatta punktmolnet istället 21556 punkter. Punkter som är borttagna är brus och ojämnheter på objektets yta, se figur 4.5. Det noteras att det fortfarande finns ojämnheter och brus kvar i punktmolnet.

En förskjutning av punktmolnet iaktas och mäts till att vara ca 4.4 mm i positivt z-led och ca 2.2 mm i negativt x-led. På punktmolnets sida har reduceringen av punkter gjort att vissa områden blivit nästan helt tomma på punkter.

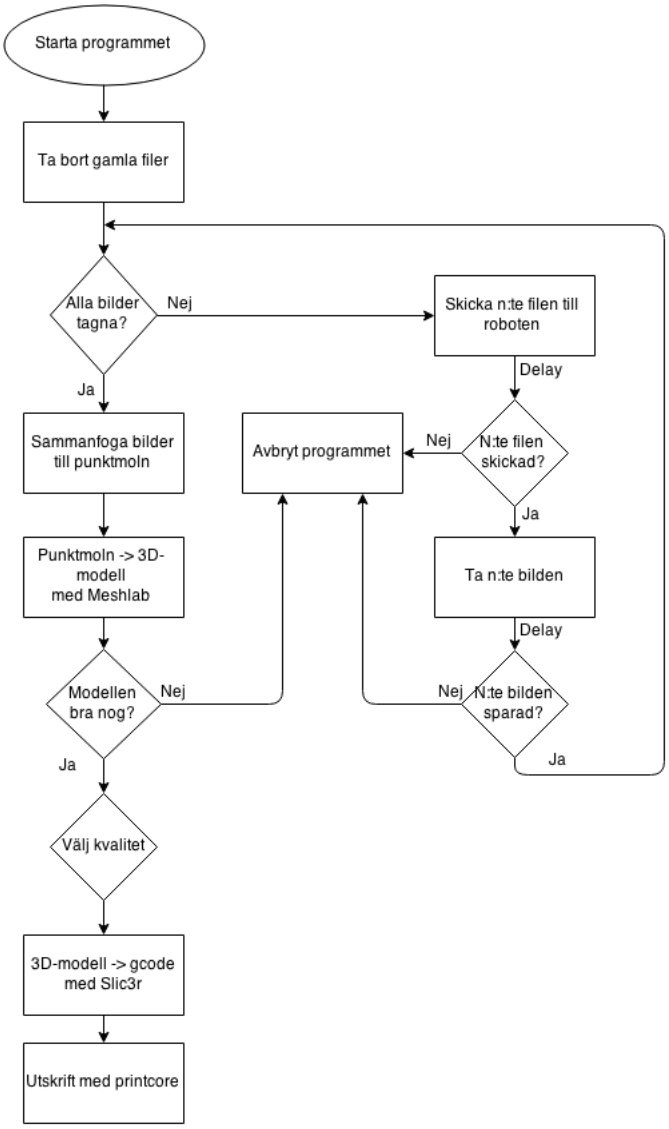
4.3 RAPID-kod



Figur 4.6: Flödeschema av RAPID-kod

I figur 4.6 visas ett flödeschema av den RAPID-kod som används. I programmets första steg kontrolleras det om det ligger några filer på robotens hårddisk med de filnamnen som ska användas i programmet. Om filerna ligger där kommer de att raderas för att inte störa fortsättningen av programmet. Efter att det steget är gjort kontrollerar programmet om första filen ligger på en specifik plats på robotens hårddisk. Om filen ligger där den ska, flyttar roboten kameran till den första positionen. Samma sekvens sker när roboten ska köra till de övriga positionerna och håller på tills att alla positioner är nådda. Filerna som skickas till hårddisken måste skickas i en förutbestämd ordning för att programmet ska fortsätta som planerat.

4.4 MATLAB-kod



Figur 4.7: Flödesschema av MATLAB-kod

I figur 4.7 visas ett flödesschema av den MATLAB-kod som används. När programmet startas tas filer från tidigare skanningar bort, för att de inte ska användas igen. MATLAB ansluter sedan till roboten via FTP, vilket gör att programmet kan skicka styrfilerna till roboten. Inskanningen börjar genom att styrfilen för kamerans första position, skickas till robotens hårddisk. Första bilden tas efter en fördröjning för att säkerställa att roboten har hunnit ta sig till sin position, detta eftersom det inte finns någon kommunikation från roboten till programmet. MATLAB väntar sedan på att bilden har sparats undan innan nästa styrfil skickas till roboten. Roboten kör då till nästa position och proceduren upprepas. När samtliga bilder är tagna sammanfogas bilderna till ett punktmoln som sparas. Därefter behandlas och filtreras punktmolnet i MeshLab.

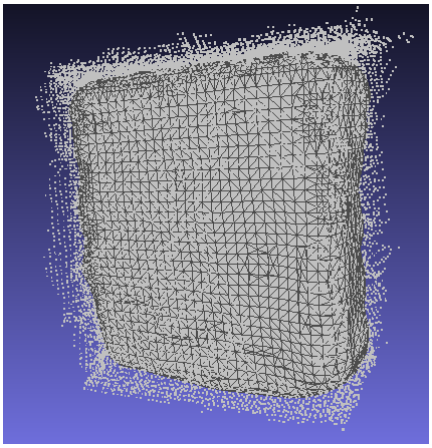
När bearbetningen är färdig i MeshLab visas en bild av resultatet. Användaren får

då välja mellan att skriva ut modellen eller att avsluta programmet. Om modellen godkänns av användaren får utskriftskvalitet väljas. När detta är gjort översätter Slic3r 3D-modellen till G-kod och skickar G-koden till skrivaren.

4.5 Behandling av skanndata

De flesta punktmoln som tagits fram är bra nog för att, med individuell behandling i MeshLab, resultera i en 3D-modell som går att skriva ut. Dock krävs det en hel del rensning av brus som kräver mer avancerade filterskript än vad det funnits tid för att ta fram. Rensningen krävs för att kunna automatisera behandlingen av skanndatan.

Filterskriptet som togs fram i 3.3.2 användes på de punktmoln som tagits fram, och i samtliga fall gick det att se på 3D-modellen vad som skannats. För att få en uppfattning av hur bra 3D-modellerna blir, sattes de ihop med motsvarande punktmoln i MeshLab. 3D-modellerna blev ofta något mindre än de skannade objekten, vilket kan bero på kraftig filtrering. I figur 4.8 visas ett punktmoln av ett rätblock, tillsammans med motsvarande 3D-modell.



Figur 4.8: Punktmoln och 3D-modell

4.6 Utskrift

Utskriften görs automatiskt och startas genom en batchfil via huvudprogrammet i MATLAB. Exempel på utskrifter från inskannade punktmoln finns i figur 4.9

Figur 4.9: Utskrivna kopior av en elastisk linda och ett rätblock

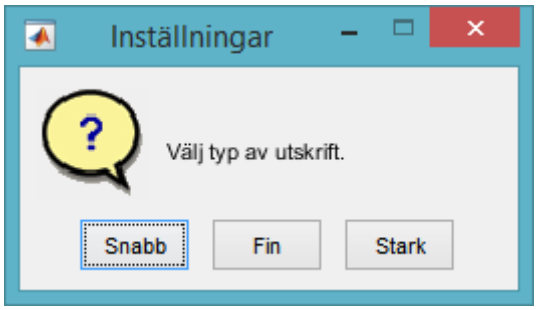
Efter en mätning av det utskrivna rätblocket noteras att differens från originalet inte överskred 5 mm.

4.7 Gränssnitt och användarvänlighet

Att två separata program måste köras samtidigt gör kopieringen något mindre användarvänlig. Genom att sätta roboten i ”autoläge” och starta MATLAB-programmet kan användaren lämna datorn och roboten, i väntan på att skanningen ska bli färdig. Först när 3D-modellen är klar behövs användarens uppmärksamhet igen. Användaren får då se hur 3D-modellen blev, och kan välja att fortsätta med kopieringen eller att avsluta programmet genom att välja alternativ i dialogrutan som visas i figur 4.10.

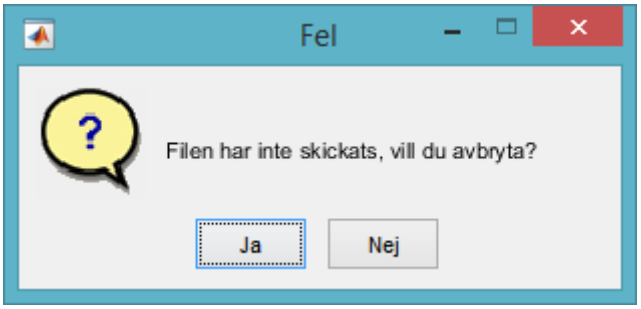
Figur 4.10: Val efter förhandsvisning av 3D-modellen

Om användaren väljer att fortsätta kopieringen, får den välja mellan olika utskriftskvaliteter som visas i figur 4.11. Detta val anpassar utskriften efter användarens behov.



Figur 4.11: Val av utskriftskvalitet

Om något fel inträffar under programmets körning dyker ett felmeddelande upp. Beroende på vad som gått fel finns olika meddelanden där användaren informeras om vad som hänt och därefter kan den välja att ändå fortsätta programmet eller avbryta det. Felen som kan uppstå är att en bild som tas inte sparas, att en fil inte skickats till roboten eller att en gammal fil inte tagits bort. Ett exempel visas i figur 4.12.



Figur 4.12: Felmeddelande

5
Diskussion

5.1 Felfaktorer

En faktor som bidrar till att fel kan uppstå är robustheten i verktyget. När roboten körs rör den sig snabbt och stannar hastigt. Det syns tydligt att skannern vibrerar när roboten stannar vid positionerna. Detta beror till stor del på att verktyget inte är tillräckligt styvt. Programmet som roboten kör är dessutom ryckigt då det stannar vid alla positioner. Problemet hade kunnat lösas om programmet ändrades så den saktar in och inte gör ett lika plötsligt stopp som det är i nuläget. När den hela tiden stannar riskerar skannern att komma ur position eftersom det inte gjordes ett stabilare fäste till skannern på verktyget. Skannern sitter fast med plastband och har statiska stöd som gör att den skall vara lätt att placera på samma plats om den skulle flyttas. Denna lösning var inte helt pålitlig och det gjordes inte någon förändring av fästanordningen då den ansågs fungera tillräckligt bra. Detta kan ha påverkat slutresultatet om punktmolnen blir sämre på grund av kamerafästet.

Linsen på Microsoft Kinect sitter inte mitt på skannern utan lite till vänster. Eftersom verktyget är gjort så att själva skannern är centrerad är bilderna alltså skjutna lite åt vänster. Det kan påverka hur punktmolnet blir och göra att kalibreringen av molnet inte blir optimalt.

5.2 Utbildning om robot och RobotStudio

Det var ingen i gruppen som hade någon erfarenhet av RobotStudio vid projektets start, därför lades mycket tid ner på att lära sig de grundläggande funktionerna i programmet. Det finns internetforum för RobotStudio där diskussionerna var till stor hjälp. Ibland var det dock för hög nivå på diskussionerna eftersom många på forumet arbetar med programmet och har en större förståelse för det. Om gruppen hade fått mer utbildning i RobotStudio hade programmeringen inte tagit lika mycket tid i uppstartsfasen. Mer tid kunde istället ha lagts på övriga delar av projektet.

Som det var tänkt i början av projektet skulle programmet se olika ut beroende på objektets form och storlek. Om detta avancerade robotprogram skulle kunna ha gjorts hade en mer ingående utbildning krävts.

5.3 Förbättringspotential

Det finns områden i projektet där ytterligare arbete skulle kunna leda till ett bättre slutresultat.

5.3.1 Fler bilder - bättre punktmoln

Det slutliga punktmolnet sammanfogas av åtta bilder från olika vinklar. Testkörningarna genererade ca 120 punkter/cm^2. För att resultatet ska bli av bättre kvalitet behövs fler punkter eller mindre brus i punktmolnet. Att få mindre brus är svårt att uppnå. Lite av bruset kan samplas bort men att automatiskt sampla bort allt är svårt. Om fler bilder hade tagits hade fler punkter fåtts, då hade det antagligen gått att sampla hårdare och på så sätt få bort mer brus. Om fler bilder ska tas finns olika sätt att göra det på. Antingen kan flera bilder tas vid varje nuvarande position eller så kan antalet bildtagningspositioner utökas. Vilket av dessa alternativ som ger bäst resultat är ännu osäkert. Att ta fler bilder vid varje position hade lett till att samma transformationsmatriser som används i nuläget hade kunnat användas igen. Istället så hade då flera bilder från samma position givit ett medelvärde för varje punktmoln, vilket hade givit en mer noggrann yta som sedan hade kunnat användas för vidare arbete. Detta hade sparat mycket tid gentemot att ta fram nya bildtagningspositioner. Bilder från flera vinklar hade gjort att objekt med överhäng, som legokorset och skålen i bilaga D, inte får skuggor där skannern inte kan se.

5.3.2 Skanning av blanka ytor

För att objektet som skannas in ska bli bra är det viktigt att objektet inte har blanka ytor. Objekt tillverkade av blanka material som metall, glas eller plast ger ofta inget bra punktmoln. Anledningen är att det blir störningar som gör att punktmolnet inte blir användbart [26]. För att kunna skanna blanka objekt, kan en beläggning läggas på objektet som gör ytan matt. Som beläggning kan till exempel talkpuder användas [27]. Alla ytor skulle kunna ge bättre punktmoln med en matt beläggning.

5.3.3 Behandling av punktmoln

Ett fortsatt arbete med att utveckla en funktion som kan behandla olika punktmolns brus och yta mer generellt bör göras. Detta leder till att allt efterarbete samt kalibrering blir lättare och att slutresultatet blir bättre.

5.4 Val av skanner

Från början var tanken att använda en 3D-skanner från DotProduct som i en jämn rörelse skulle svepa runt objektet för att få ett punktmoln av objektet. Flera aspekter gör att denna skanner inte är möjlig att använda. Bland annat är skannerns programvara låst till Android vilket gör det svårt att styra den utifrån. Det hade varit möjligt genom att utveckla en app, men det i sig hade varit ett stort projekt. Det skulle komma ett program till skannern som skulle göra den lättare att använda i projektet, det programmet dröjde dock och har fortfarande inte kommit ut. Ytterligare en aspekt var att punktmolnen från skannern inte var tillräckligt noggranna. Anledningen till det är att skannern egentligen är tänkt att skanna av omgivningar.

Innan Microsoft Kinect valdes, testades en Xbox Kinect. Den fungerade dock inte tillsammans med MATLAB på grund av att det är en annan hårdvara i skannern för att fungera med ett Microsoft Xbox 360. Det var svårt att hitta en skanner vars programvara går att använda utan att användaren behöver kontrollera den manuellt.

Att undersöka olika skannrar på internet är svårt då test ej kan utföras på dem samt att information om skannrarna ofta är svåråtkomlig. Det upptäcktes tidigt att det finns ett stort utbud av skannrar, men många av dem är för dyra för projektet. Noggrannheten hos de dyrare skannrarna skulle säkerligen varit högre vilket hade givit ett bättre slutresultat.

5.5 Användning av robot

I projektets förutsättningar ingår det att en robot ska användas för att genomföra kopieringen. Skannern skulle då fästas på roboten för att skanna från alla håll runt objektet. Om en robot inte hade behövt användas skulle skannern kunna vara stationär. Då hade ett alternativ varit att låta objektet rotera framför skannern. Andra förändringar som hade kunnat ske om roboten hade uteslutits hade de komplicerade uträkningarna i MATLAB inte behövt göras. Eftersom roboten används kan dess höga noggrannhet utnyttjas.

Många färdiga programvaror är uppbyggda på ett sätt som är beroende på att skannern står still och objektet roterar, alltså i princip tvärt emot hur det här projektet gör.

5.6 Val av skrivare

Den skrivare som användes var RepRapPro Ormerod 1 och var en av de skrivare som tillhandahölls projektet. Skrivaren var bra på det sätt att den var kompatibel med ett antal olika program samt att många inställningar kunde göras på den. Kvaliteten på skrivaren var dock inte den bästa då skrivaren är en billigare modell. Skrivaren är införskaffad i delar och därav ej monterad i fabrik vilket gör att små fel lätt kan ha uppstått under monteringen vilket i sin tur kan ha påverkat slutresultatet.

5.7 Skannerns avstånd till objektet

Enligt teorin i 2.3 ska avståndet från skannern till objektet vara minst 40 cm för att en bild ska kunna genereras. Det fungerade dock inte i praktiken. Objekt som var närmre än 60-65 cm uppfattades inte av skannern. Därför utökades avståndet till lådan till 70 cm för att vara säker på att alla punkter kommer med. Anledning kan vara att skannern har utsatts för något slag som ändrat dess kalibrering eller liknande.

5.8 Kalibrering

Den första metoden som användes gick ut på att ta fram lådans fyra hörn ur ett punktmoln för att sedan mäta ut samma punkter med robotens precision. På så sätt fås ett förhållande mellan skannern och robotens referenssystem. Resultatet av processen visas när alla transformationers origo, (fjärde kolumnen), flyttas till samma punkt genom translation. X-, y- och z-värdena ska vara samma för alla matriser. De största differenserna är i x-led på 7 mm mellan O_1 och O_2, i y-led 19 mm mellan O_2 och O_3 samt 3 mm i z-led mellan O_2 och O_3. Detta leder till att punktmolnens sammansättning får ett fel upp till 19 mm i origo. En slutsats

drogs att skannern har för dålig noggrannhet för den här metoden och att robotens precision istället ska utnyttjas.

$$O_1 = \begin{bmatrix} -369.1190 \\ -349.5630 \\ 492.8052 \end{bmatrix}, O_2 = \begin{bmatrix} -376.6194 \\ -338.3049 \\ 493.2814 \end{bmatrix}$$

$$O_3 = \begin{bmatrix} -365.4898 \\ -357.3889 \\ 490.3613 \end{bmatrix}, O_4 = \begin{bmatrix} -369.8330 \\ -343.7183 \\ 492.1543 \end{bmatrix}$$

Även i den valda metoden är skannerns noggrannhet ett stort problem som gör att arbetet med kalibrering blir svårare. Robotens noggrannhet är mycket större än skannerns och kan inte utnyttjas till fullo på grund av skannerns låga kvalitet. De resterande transformationsmatriserna beräknas utifrån den framtagna matrisen för kalibreringen. Detta gör att alla punktmoln transformeras med samma precision. Är koordinatsystemet lite förskjutet så blir den även det för alla andra transformationsmatriser.

För att få ett bättre kalibrerat system måste testutskrifter göras för att se om objektet blir förskjutet i x-, y- eller z-led eller om objektet blir skevt. Förskjutningen kan kompenseras genom att manuellt ändra läget på origo. Det görs genom att ändra på värdena på translationen av origo i fjärde kolumnen i transformationsmatrisen. Blir koordinatsystemet skevt så kan man även ändra på rotationsdelen i matrisen. Detta borde inte behövas då systemet enligt resultaten är ortogonalt. När detta är kalibrerat i första matrisen beräknas sedan de resterande matriserna fram och får samma nya kalibrering. Kalibreringen funkar i alla olika positioner eftersom skannern alltid tar bild på samma sätt.

Det har varit problem att få ut bra utskrifter på grund av bristande kvalitet på punktmolnen. Detta har gjort att en manuell justering av origo samt rotationsdelen inte varit möjlig att genomföra.

Fler punkter i planet skulle kunna göra att kalibreringen blivit mer noggrann. Tagning av flera bilder ur kalibreringspositionen skulle kunna användas för att ge ett medelvärde av punkterna för bättre kalibrering.

Det finns brister i fitPlaneRANSAC då inte alla kombinationer av punkter itereras igenom, vilket påverkar tillförlitligheten för planets ekvation. Kraven som är ställda vid kalkylering är att avståndet från planet till punkterna får vara max 1 mm och att avståndet mellan punkterna i planet ska vara 100 mm. Tid skulle kunna läggas på att optimera metoden i fitPlaneRANSAC för att få ett bättre beräknat plan. Till exempel genom att se om ett bättre plan hade hittats med andra värden på säkerhet och avstånd för punkter i planet samt avståndet mellan de punkter som beräkningen utgår ifrån. Test att iterera igenom alla punkter i planet har gjorts. Problem uppstod då konstiga kombinationer av beräkningspunkter hittades eftersom funktionen vill ha det plan som har flest inliers. Detta kan bero på att planet var för brusigt.

Koordinaterna för punkterna i punktmolnet räknades först ut för hand enligt teorin. Sedan upptäcktes stöd för detta i MATLAB-funktionen depthToPointCloud. En jämförelse har inte gjorts av de manuellt beräknade punkterna och resultatet från funktionen. Ett antagande har gjorts att MATLAB jobbar med standardvärdena gällande skannerns fokus och synvinklar. En annan kalibrering av skannern och modifiering av funktionen hade kunnat leda till bättre punktmoln.

5.9 Punktmoln

För att punktmolnet ska vara användbart behöver det behandlas. Funktionen NoiseReduction som användes är bäst anpassad för objekt med plana ytor. Finns det komplexa delar på objektet kommer dessa förmodligen ses som ojämnheter och filteras bort.

En fönsterfiltreringsfunktion har gjorts och tester har påbörjats. Funktionen går ut på att punktmolnet delas upp i olika så kallade fönster i z-led. Första fönstret är hela objektet i ett bestämt intervall i z-led. Alla z-värden som är inom intervallet och motsvarande x- och y-värden i punktmolnet sparas undan. Sedan skärs en kub ut och punkterna däri sparas undan. Är antalet punkter i kuben över en bestämd gräns tas medelvärdet fram av alla punkter och en ny punkt bildas och sparas undan. Kuben flyttas över hela fönstret med ett bestämt intervall och samma process utförs tills x och y når lådans ände. Processen upprepas i ett nytt fönster efter att fönstret har flyttats ett bestämt intervall uppåt i z-led. Detta gör att de punkterna som tagits fram från föregående fönster kan tas med igen i det nya fönstrets kuber. Resultatet blir en jämnare yta eftersom medelvärdet av andra medelvärden tas hos punkterna. Detta utförs till z har gått från 0-200 mm.

Efter tester visar det sig att antalet punkter i molnet inte är tillräckligt för att få en jämn yta. Brister finns antagligen i funktionen då fler tester och undersökningar inte hunnits med.

Punktmolnet är förskjutet 2.2 mm i x-led jämfört med CAD-modellen. Det kan bero på att något gick fel vid inmätningen av origo på lådan. Står objektet lite fel blir proportionen för objektet rätt vid utskrift, men inte i jämförelse med CAD-modellen, då den utgår från att ena hörnet är placerat i origo. Punktmolnet är förskjutet i z-led ungefär 4.4 mm. Detta kan bero på att origo är för högt upp i z-led i transformationsmatrisen. Det leder till att brus och delar av lådan kan komma med efter att punktmolnet har beskurits i z-led.

Fler punkter i punktmolnet hade även gjort att funktionen NoiseReduction skulle kunna returnera ett bättre punktmoln med säkrare medelvärdesberäkningar vid kalibreringen. Molnet hade sedan kunnat användas i fitPlaneRANSAC för att få ett mer noggrant beräknat plan. NoiseReduction användes inte vid kalibreringen då det inte verkade vara något problem med brus och ojämnheter. I efterhand hade en behandling av kalibreringsmolnet varit bra för att få ett bättre slutresultat. NoiseReduction har gjort att vissa delar av punktmolnet har områden där det är väldigt få punkter. Detta gör att en fortsatt bearbetning av molnet för att få en utskrivsbar

modell nästan är omöjlig. Funktionen hade kunnat göra punktmolnet finare, men då krävs det att det finns fler punkter i molnet från skanningen.

6
Slutsats och rekommendationer

Kopiering av 3-dimensionella objekt med hjälp av en industrirobot är en komplex process och är något som bör undersökas vidare. Projektet har bevisat att en automatisk process av kopiering med hjälp av robot, 3D-skanner och 3D-skrivare är möjlig. Genom användning av Microsoft Kinect, roboten IRB-1600 och 3D-skrivaren *Ormerod* från *RepRapPro* utfördes kopieringen i detta projekt. Då flertalet olika programvaror behövde användas var det viktigt att de var kompatibla med varandra för att de olika delarna skulle fungera ihop.

Att använda en industrirobot för att skanna in ett objekt med hjälp av en svepande rörelse skapar en del problem. Många befintliga skanningsprocesser går ut på att skanningsobjektet roterar och skannern står still. Kinecten som användes fungerade på det vis att den rörde sig runt objektet och tog 8 bilder som sedan sammanfogades. Det gjorde att det krävdes att positionerna där bilderna tagits sparades. I det här fallet kunde positionerna fås fram i och med att roboten har ett eget kordinatsystem. Om ett liknande kopieringsystem används utan en industrirobot, skulle koordinaterna vara svårare att ta fram. Det kan lösas genom att låta skannern vara på samma ställe under hela skanningen.

Skannern som användes var tillräckligt bra för att uppfylla projektets krav. Om kravet på kvaliteten vore högre hade en bättre skanner behövts. Punkterna i punktmolnet som bildades med hjälp av Microsoft Kinect kunde variera upp till 4.5 mm jämfört med det verkliga objektet. Skanningen är gjord på kortaste möjliga avstånd från skanningsobjektet och hade avståndet ökat skulle felet bli större.

Efter att ett punktmoln har framställts behövs ett antal redigeringar för att skapa en utskriftsbar fil. Största arbetet ligger i att behandla grunddatan från skannern så att punktmolnet blir användbart. Det upptäcktes att det behövs olika redigeringar beroende på vilket objekt som skannas in. Om en vidarutveckling av kopieringsprocessen görs kan en metod för identifiering av objektets geometri vara nödvändig. Detta för att passande filtrering ska användas till olika objekt.

Ormerodskrivaren uppfyllde de krav som ställdes på den. Den klarade att ta emot data och skriva ut objekt som tidigare skannats in så att objekten kunde identifieras. Eftersom 3D-modellerna inte är exakta kopior av det verkliga objektet blir därför inte utsrkriften en bra kopia utseendemässigt.

Kvaliteten på den slutgiltiga utskriften var inte fokus för projektet utan målet var att få till en fungerande automatisk kopieringsprocess. Det uppnåddes dock ett resultat

som var inom skannerns angivna noggrannhet. Eftersom processen som skapades fungerar anses målet med projektet vara uppnått.

Litteraturförteckning

[1] Karlsson, M., Persson, J., Ringius, H., Rostedt, M. 2014. *Kandidatarbete: 3D-skrivarsystem*. Chalmers Tekniska Högskola

[2] ABB *IRB 1600*. http://www.abb.com/product/seitp327/7e3f68d2bb3a6f93c125727a00568dd2.aspx Hämtad 2015-04-28

[3] Lipson, H., och Kurman, M. 2013. *Fabricated: The New World of 3D Printing*. Indianapolis: John Wiley & Sons, Inc.

[4] Jinming, Z. *Nationalencyklopedin, friformsframställning*. http://www.ne.se/uppslagsverk/encyklopedi/lång/friformsframställning Hämtad 2015-03-20

[5] MATLAB *Math Works, Microsoft Kinect for Windows Support from Image Acquisition Toolbox*. http://se.mathworks.com/hardware-support/kinect-windows.html?refresh=true Hämtad 2015-04-21

[6] Khoshelham, K. *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*.http://www.mdpi.com/1424-8220/12/2/1437/htm Hämtad 2015-05-14

[7] Microsoft *Microsoft, Kinect for Windows Sensor Components and Specifications* https://msdn.microsoft.com/en-us/library/jj131033.aspx Hämtad 2015-04-21

[8] Eisler, C. *Microsoft, Near Mode: What it is (and isn't)* http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx Hämtad 2015-05-10

[9] Liu, Y *ROS: kinect accuracy*. http://wiki.ros.org/openni_kinect/kinect_accuracy Hämtad 2015-04-21

[10] ABB *ABB*. http://new.abb.com/products/robotics/robotstudio Hämtad 2015-04-21

[11] Delfoi *Offline-programmering*. http://www.delfoi.com/web/web_swe/solutions/kortaserier/sv_SE/kortaserier/ Hämtad 2015-04-30

[12] MATLAB *MATLAB, Features*. http://se.mathworks.com/products/matlab/features.html/ Hämtad 2015-04-24

[13] MeshLab *MeshLab*. http://meshlab.sourceforge.net Hämtad 2015-04-30

[14] Ranellucci, A. *Slic3r: G-code generator for 3D printers*. http://slic3r.org/about Hämtad 2015-03-20

[15] Yanev, K. *Printrun: Pure Python 3d printing host software* http://www.pronterface.com/] Hämtad 2015-05-09

[16] Staberg, P. fitPlaneRansac *find dominant plane in pointcloud with RANSAC* MatrNr:1327767. Kontakt via mail. 2015-03-25

[17] MathWorks. *Using RANSAC for estimating geometric transforms in computer vision* http://se.mathworks.com/discovery/ransac.html?refresh=true Hämtad 2015-05-20

[18] WolframMathWorld. *Plane* http://mathworld.wolfram.com/Plane.html Hämtad 2015-04-28

[19] Nyqvist, P. *Kinematics: "The relationships between the positions, velocities and accelerations of the links of a manipulator"* Hämtad 2015-04-02

[20] MathWorks. *pcdownsample* http://se.mathworks.com/help/vision/ref/pcdownsample.html Hämtad 2015-05-02

[21] MathWorks. *pcdenoise* http://se.mathworks.com/help/vision/ref/pcdenoise.html Hämtad 2015-05-02

[22] ABB Developcenter *MoveL - Moves the robot linearly* http://developercenter.robotstudio.com/BlobProxy/manuals/RapidIFDTechRefManual/doc101.html Hämtad 2015-05-12

[23] ABB Developcenter *MoveJ - Moves the robot by joint movement* http://developercenter.robotstudio.com/BlobProxy/manuals/RapidIFDTechRefManual/doc98.html Hämtad 2015-05-12

[24] Hazelden, A. 2014. *Meshlabserver Automation with the MultiMeshScripting Tool*. http://www.andrewhazelden.com/blog/2014/06/ Hämtad 2015-03-20

[25] Corsini, M., Cignoni, P., och Scopigno, R. 2012. *Efficient and flexible sampling with blue noise properties of triangular meshes.* Visualization and Computer Graphics, IEEE Transactions on , 18(6), 914-924.

[26] Zaimovic-Uzunovic, N., och Lemes, S. 2010. *Influences of surface parameters on laser 3D scanning*. In 10th International Symposium on Measurement and Quality Control (ISMQC 2010) September (pp. 5-9).

[27] Millstein, B. 2013 *Digitizer Update 8 - Digitizing Tips* http://www.makerbot.com/blog/2013/07/19/digitizer-update-8-digitizing-tips/ Hämtad 2015-05-13

A

Transformationsmatriser

$$T_1 = \begin{bmatrix} 0,710016 & -0,50234 & 0,493486 & -514,119 \\ -0,70411 & -0,51668 & 0,487101 & -494,563 \\ 0,010284 & -0,69332 & -0,72056 & 492,8052 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 0,999938 & 0,010139 & 0,004515 & -13,828 \\ 0,004176 & -0,72056 & 0,69338 & -713,245 \\ 0,010283 & -0,69331 & -0,720558 & 492,805181 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} 0,704110 & 0,5166820 & -0,487100 & 494,562865 \\ 0,710015 & -0,502340 & 0,4934858 & -514,118580 \\ 0,010283 & -0,693318 & -0,720558 & 492,805181 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} -0,004175 & 0,720560 & -0,693379 & 713,245489 \\ 0,9999384 & 0,010139 & 0,004514 & -13,827978 \\ 0,010283 & -0,693318 & -0,720558 & 492,805181 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5 = \begin{bmatrix} -0,710015 & 0,502343 & -0,493485 & 514,11858 \\ 0,704110 & 0,516682 & -0,487100 & 494,56286 \\ 0,010283 & -0,693318 & -0,720558 & 492,805181 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6 = \begin{bmatrix} -0,999938 & -0,010139 & -0,004514 & 13,827978 \\ -0,004175 & 0,720560 & -0,693379 & 713,245489 \\ 0,010283 & -0,693318 & -0,720558 & 492,805181 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_7 = \begin{bmatrix} -0,004175 & 0,720560 & -0,693379 & 713,245489 \\ 0,9999384 & 0,010139 & 0,004514 & -13,827978 \\ 0,010283 & -0,693318 & -0,720558 & 492,805181 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_8 = \begin{bmatrix} -0,704110 & -0,516682 & 0,487100 & -494,562865 \\ -0,710015 & 0,5023435 & -0,493485 & 514,118580 \\ 0,010283 & -0,693318 & -0,720558 & 492,805181 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B
Filter

Compute normals for point sets

- Neighbour num = 20
- Smooth Iteration = 0
- Flip normals w.r.t viewpoint = false
- Viewpoint Pos. = 0,0,0

Poisson Disk Sampling

- Number of samples = 15000
- MonterCarlo OverSampling = 20
- Approximate Geodesic Distance = false
- Base Mesh Subsampling = true
- Refine Existing Samples = false
- Best Sample Heuristic = true

Poisson Surface Reconstruction

- Octree Depth = 9
- Solver Divide = 8
- Samples per Node = 5
- Surface offsetting = 1

C

Förutsättningar

C.1 Köra Pythonskript i Windows

För att kunna köra Pythonskript i Windows behöver följande steg utföras först. Stegen är hämtade från http://stackoverflow.com/a/21373411.

1. Ladda ner Python 2.7.X för aktuell Windows version från: `https://www.python.org/download/releases`
2. Installera till C:\Python27
3. Öppna Kontrollpanelen och navigera till: System and Security\System\Advanced system settings
4. Klicka på 'Environment Variables'
5. Under 'System Variables', klicka på variabeln 'Path' och sedan 'Edit...'
6. Utan att ta bort befintligt text; Lägg till 'C:\Python27;' (inklusive semikolon) till början av 'Variable value' och klicka på OK
7. Klicka OK i 'Environement Variables' fönstret
8. Starta om datorn

C.2 Köra Pythonskript i MATLAB

Guide hämtad från `http://stackoverflow.com/a/1709660`.

1. Gör en kopia av C:\... \MATLAB\R2015\toolbox\matlab\general\perl.m i samma mapp och döp till python.m
2. Ersätt 'perl' med 'python' på alla ställen i filen
3. Spara

C.3 Använd Microsoft Kinect i MATLAB

Guide hämtad från `http://se.mathworks.com/help/imaq/installing-the-kinect-for-windows-html?refresh=true`.

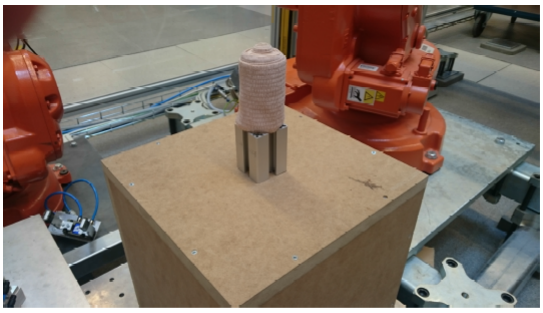
1. I MATLAB: skriv 'supportPackageInstaller'

2. Välj: Install from Internet
3. Välj: Kinect for Windows Runtime
4. Fullfölj installationen

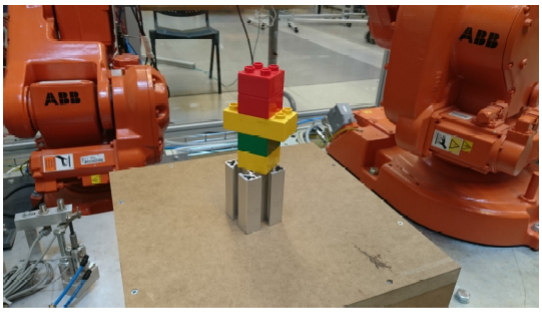
D

Exempel på kopieringar

Flera testkörningar av kopieringsprogrammet har gjorts. I figur D.1 visas fyra av de objekt som skannats.

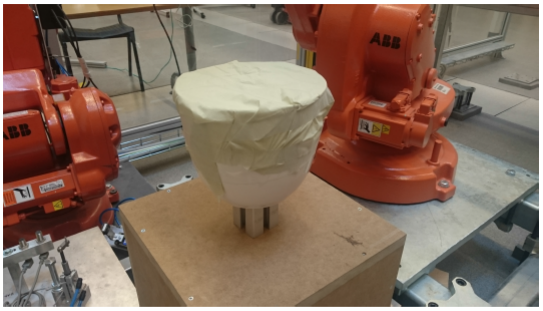


(a) Gasbinda



(b) Legokors

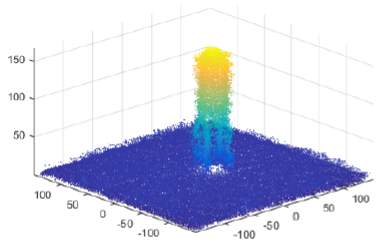
(c) Plånbok



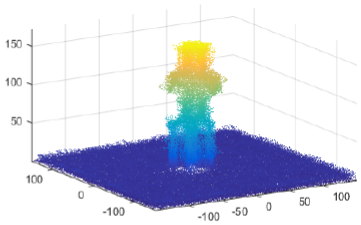
(d) Skål

Figur D.1: Skannade objekt

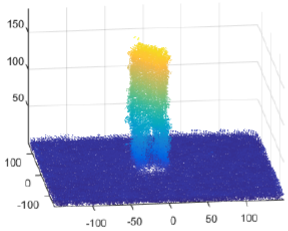
För att förbättra chanserna för en bra skann, valdes objekt med enkla geometrier. Skålen som användes hade en för blank yta, och täcktes därför med papper.
I figur D.2 visas de punktmoln som genererades av skanningarna.



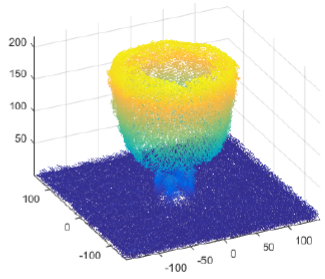
(a) Gasbinda



(b) Legokors



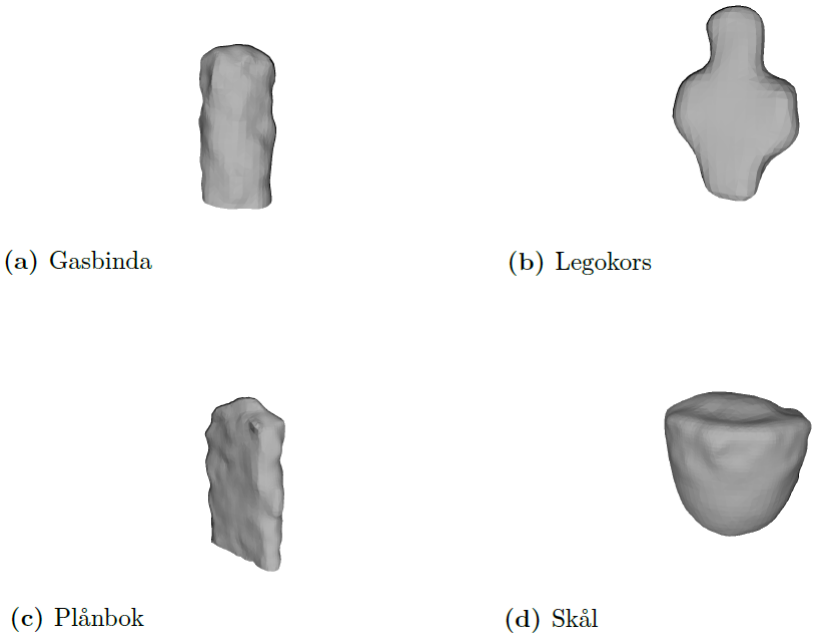
(c) Plånbok



(d) Skål

Figur D.2: Punktmoln

I figur D.3 visas de 3D-modeller som punktmolnsbehandlingen resulterade i. Eftersom skannern endast tar kort ovanifrån objektet, tolkas den tvärliggande delen av legokorset på fel sätt. Detta kan troligtvis undvikas genom låta skannern ta bilder både snett under- och ovanifrån.



(a) Gasbinda

(b) Legokors

(c) Plånbok

(d) Skål

Figur D.3: 3D-modeller av skannade objekt

Kopieringen fungerade automatiskt som tänkt. Kvaliteten på 3D-modellerna är bra nog för att det ska gå att se vilket objekt 3D-modellet skapats av.

E

Bidragsrapport

Nedan listas samtliga gruppmedlemmar tillsammans med vilka olika delar de har bidragit med. Delar som gruppen bidragit lika mycket med, står inte med. Numreringen av delar i rapporten syftar till samtliga underliggande delar. Det vill säga att 1.3 innefattar 1.3.1 - 1.3.5.

Per Andersson

- Huvudansvar för skanning och behandling av punktmoln i MATLAB
- Huvudansvarig för transformationsmatriserna
- Huvudansvarig för drift av 3D-skrivaren
- Delförfattare av 1
- Huvudförfattare av 2.3
- Huvudförfattare av 2.5
- Delansvar för robotprogrammering
- Delförfattare av 3.2.1
- Delförfattare av 3.2.3
- Huvudförfattare av 3.3.1
- Huvudförfattare av 4.2
- Huvudförfattare av 5.3.2
- Huvudförfattare av 5.8
- Huvudförfattare av 5.9
- Delförfattare av 6

Matilda Halldén

- Delansvar för utskrift
- Delansvar för sammanlänkning av mjukvaran i MATLAB
- Delansvar för huvudprogrammet i MATLAB

- Delförfattare av 1
- Delförfattare av 2.2
- Delförfattare av 2.3
- Delförfattare av 3.3
- Delförfattare av 4.6
- Delförfattare av 4.7
- Delförfattare av 5.1
- Delförfattare av 6

Christoffer Hildebrand

- Huvudansvar för CAD-modeller
- Huvudansvar för tillverkning av verktyg
- Huvudansvar för tillverkning av skanningsbord
- Delansvar för robotprogrammering
- Delförfattare av 1
- Huvudförfattare av 2.1
- Huvudförfattare av 2.4.1
- Huvudförfattare av 3.1.1
- Huvudförfattare av 3.1.2
- Huvudförfattare av 3.1.3
- Huvudförfattare av 5.1
- Huvudförfattare av 5.2
- Delförfattare av 6

Carl Hornborg

- Huvudansvar för huvudprogrammet i MATLAB
- Huvudansvar för behandling och filtrering i MeshLab
- Huvudansvar för sammanlänkning av mjukvara
- Delansvar för utskrift
- Delansvar för RAPID-kod

- Delförfattare av 1
- Delförfattare av 2.2
- Delförfattare av 2.4.3
- Delförfattare av 2.4.4
- Huvudförfattare av 2.4.5
- Huvudförfattare av 3.3.2
- Huvudförfattare av 3.4 - 3.6
- Delförfattare av 4.1
- Delförfattare av 4.3
- Huvudförfattare av 4.4 - 4.7
- Delförfattare av 5.3.1 - 5.3.2
- Delförfattare av 6

Daniel Pantzar

- Huvudansvar för robotprogrammering.
- Huvudansvar för RAPID-kod
- Huvudansvar för tillverkning av verktyg
- Huvudansvar för tillverkning av låda
- Delansvar för huvudprogrammet i MATLAB
- Delförfattare av 1
- Delförfattare av 2.1
- Delförfattare av 2.4.2 - 2.4.4
- Delförfattare av 3.1.1 - 3.1.2
- Huvudförfattare av 3.2.1 - 3.2.2
- Delförfattare av 4.1
- Delförfattare av 4.3
- Huvudförfattare av 5.3.1
- Delförfattare av 5.3.2
- Delförfattare av 5.4

- Huvudförfattare av 5.5
- Huvudförfattare av 5.7
- Delförfattare av 6

Josef Östling

- Huvudansvar för undersökning och val av kamera
- Huvudansvar för undersökning och val av programvara för skanning
- Delansvar för skanning och behandling av punktmoln i MATLAB
- Delansvar för tillverkning av verktyg
- Delförfattare av 1
- Delförfattare av 2.3
- Huvudförfattare av 2.4.2
- Huvudförfattare av 3.1.4-3.1.5
- Huvudförfattare av 4.1
- Delförfattare av 4.7
- Huvudförfattare av 5.4
- Delförfattare av 5.5
- Huvudförfattare av 5.6
- Huvudförfattare av 6