



AARHUS SCHOOL OF ENGINEERING

SUNDHEDSTEKNOLOGI OG INFORMATIONS- OG
KOMMUNIKATIONSTEKNOLOGI
BACHELORPROJEKT

AUTOMATISK ULTRALYDSSCANNER

Udviklingsdokumentation

Charlotte Søgaard Kristensen (201371015)

Mathias Siig Nørregaard (201270810)

Marie Kirkegaard (201370526)

Vejleder

Associate Professor

Michael Alrøe

Aarhus School of Engineering

2. december 2016

Indholdsfortegnelse

Indholdsfortegnelse	1
Kapitel 1 Versionshistorik	2
Kapitel 2 Indledning	3
Kapitel 3 Systemarkitektur	4
3.1 Systemoversigt	4
3.1.1 Domænemodel	4
3.1.2 Block definition diagram	5
3.1.3 Internal block diagram	5
3.2 Systemets interfaces	6
3.2.1 UR10	6
3.2.2 Kinect	7
3.3 Softwarearkitektur	7
Kapitel 4 Systemdesign	8
4.1 Pakkediagram	8
4.2 Klassediagram	9
4.2.1 GUI	9
4.2.2 ComputerVisionLibrary	10
4.2.3 ConversionLibrary	11
4.2.4 RoboLibrary	12
4.3 Sekvensdiagrammer	13
4.3.1 3D scan	13
4.3.2 Feed Path	14
4.3.3 Indlæsning af Robotarms positur	15
4.3.4 Ultralydsscan	15
4.4 Detajleret specifikation af klassediagrammer	16
4.5 Tilstandsdiagram	16
4.6 Beregninger	17
Kapitel 5 Udviklingsmiljø	19
Bilag	20

Versionshistorik

1

Version	Dato	Ansvarlig	Beskrivelse
1.0	2016-11-01	CSK	Første version af udviklingsdokument med systembeskrivelse, bdd og ibd
1.1	2016-11-03	CSK	Sekvensdiagrammer tilføjet
1.2	2016-11-04	CSK, MSK	Forklaringer til sekvens- og klassediagrammer

Tabel 1.1: Versionshistorik

Indledning 2

Dette dokument indeholder arkitektur og design for systemets hardware og software. Formålet med dokumentet er at give et overblik over, hvordan systemet Automatisk Ultralydsscanning er designet og udviklet. Forklaring på forkortelser findes i bilag sætningslist.

Systemarkitektur 3

Der er udarbejdet forskellige diagrammer på baggrund af de specificerede systemkrav. Diagrammerne har til formål at dele systemet op i realiserbare dele for at vise arkitekturen for og designet af systemet.

Arkitekturen beskriver den grundlæggende organisering af Automatisk Ultralydsscanner og opbygningen af dens tilhørende PC Applikation. Systemet er opbygget generisk, så man vil kunne udskifte komponenter som f.eks. robotarmen eller 3D kameraet med en anden type eller model. Udskiftning af komponenter vil dog resultere i en anderledes implementering. Der er i diagrammerne designet ud fra, at 3D kamera er af typen Microsoft Kinect 2.0 og Robotarmen er en UR10 robot.

3.1 Systemoversigt

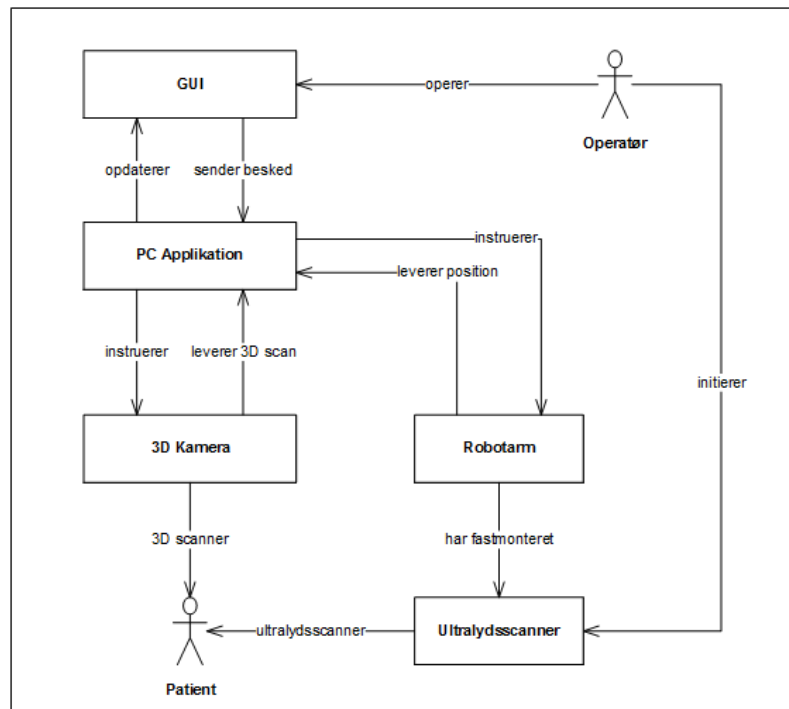
Systemet Automatisk Ultralydsscanner består af en PC applikation, en Ultralydsscanner, en Robotarm af typen Universal Robots UR10, et 3D kamera af typen Microsoft Kinect 2.0, samt et Access Point, af typen D-Link DAP-1160, til forbindelse mellem Robotarm og PC applikation. Der er fem aktører, Robotarm, Ultralydsscanner, 3D kamera, Operatør og Patient, som interagerer med PC applikation.

Robotarm har en fastmonteret touch skærm, hvorpå Operatør manuelt kan flytte Robotarm, se Robotarms koordinator mm. Robotarm er forbundet via et ethernet kabel til et Access Point, så at PC Applikation kan kommunikere trådløst til den. 3D kamera er forbundet til en computer via USB 3.0. Ultralydsscanner er en separat enhed, som er fastgjort på Robotarm, men indgår i det fulde system Automatisk Ultralydsscanner. Ultralydsscanner skal manuelt tændes og slukkes af Operatør.

For Automatisk Ultralydsscanner er der udarbejdet forskellige diagrammer, som har til formål at dele systemet op i blokke og vise integration mellem blokkene, samt forbindelser mellem aktøre.

3.1.1 Domænemodel

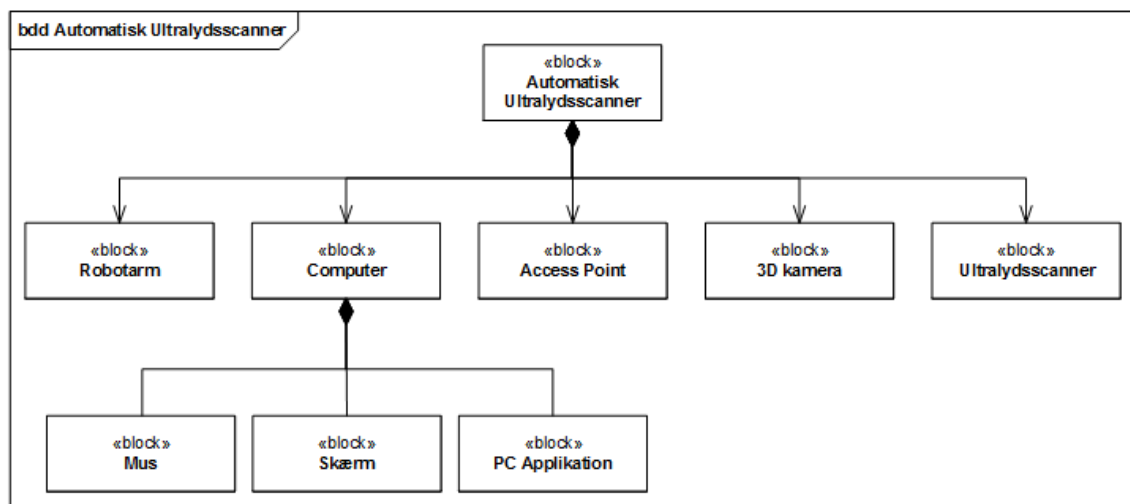
Domænemodellen på Figur 3.1 viser forbindelserne mellem de forskellige aktører i systemet.



Figur 3.1: Domænemodel for Automatisk Ultralydsscanner

3.1.2 Block definition diagram

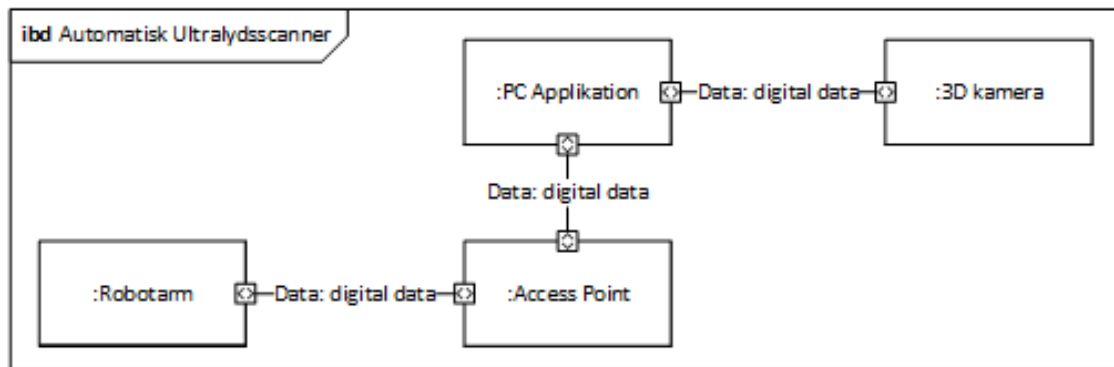
Block Definition Diagram Figur 3.2 viser systemet blokke.



Figur 3.2: BDD for Automatisk Ultralydsscanner

3.1.3 Internal block diagram

Internal Block Diagram Figur 3.3 viser systemets interne forbindelser og flow mellem blokkene. Bemærk at Ultralydsscanner ikke er inkluderet, da den ikke har forbindelse til de andre blokke udover at være fysisk monteret på Robotarm.



Figur 3.3: IBD for Automatisk Ultralydsscanner

3.2 Systemets interfaces

Systemet består af to interfaces, mellem PC Applikation og Kinect og mellem PC Applikation og UR10. Kommunikationen mellem PC Applikation og Kinect sker over modbus-protokollen, hvorimod kommunikationen mellem PC Applikation og Kinect er gennem Kinect's API. Se afsnittene nedenunder for en detaljeret beskrivelse af disse interfaces.

3.2.1 UR10

Overførsel af data til UR10 sker gennem Transmission Control Protocol/IP-protokollen (TCP/IP). Til afsendelse af positur-værdier fra PC Applikation anvendes modbus-protokollen. Modbus-protokollen sørger for at skrive binære værdier på registre på UR10-controlleren. UR10 kører på URScripts, hvis den skal styres automatisk. UR10 har et script der læser værdierne på registrene og instruerer UR10s Tool Center Point (TCP) til at flytte sig til en positur med en given acceleration og hastighed.

TCP/IP

PC Applikation skriver til UR10 over en TCP/IP forbindelse på en IP. Der anvendes to porte; port 502 for kommunikation over modbus-protokollen, og port 30002 for indlæsning af nuværende værdier. Kommunikationen over port 502 er både read og write, hvor port 30002 kun er read. Bemærk, at forskellen ligger i at de værdier der bliver overført på port 502 kun er til styring af UR10 på script-niveau, som fx den ønskede positur, hastighed og acceleration. Modsat, på port 30002, indhentes de nuværende tilstandsværdier som UR10 har, som fx posituren den reelt har. For at give et eksempel på hvordan dette foregår sekvensmæssigt: PC Applikation sender en positur over port 502. Værdierne i denne positur indskrives på UR10s registre. URSkriptet aflæser disse værdier og instruerer UR10 i at flytte sit TCP til denne positur. Efter noget tid vil den have nået denne positur. Der vil løbende kunne aflæses om UR10 har nået posituren på port 30002.

Port 502

URScript Skriptet der kører på UR10 aflæser i alt 3 værdier: Acceleration, hastighed og positur. Positur består af værdierne X, Y og Z for position og RX, RY og RZ for rotation. Positur-værdierne aflæses på registrene 135-140 og indsættes som properties i et 'pose'

objekt. Accelerationsværdien, hastighedsværdien samt pose-objektet gives som parameter til funktionen *move1*. UR10s TCP flyttes lineært i funktionen *move1*. Tiden på acceleration og deceleration styres omvendt proportionalt af accelerationsværdien, mens den maksimale hastighed styres af hastighedsværdien.

Port 30002

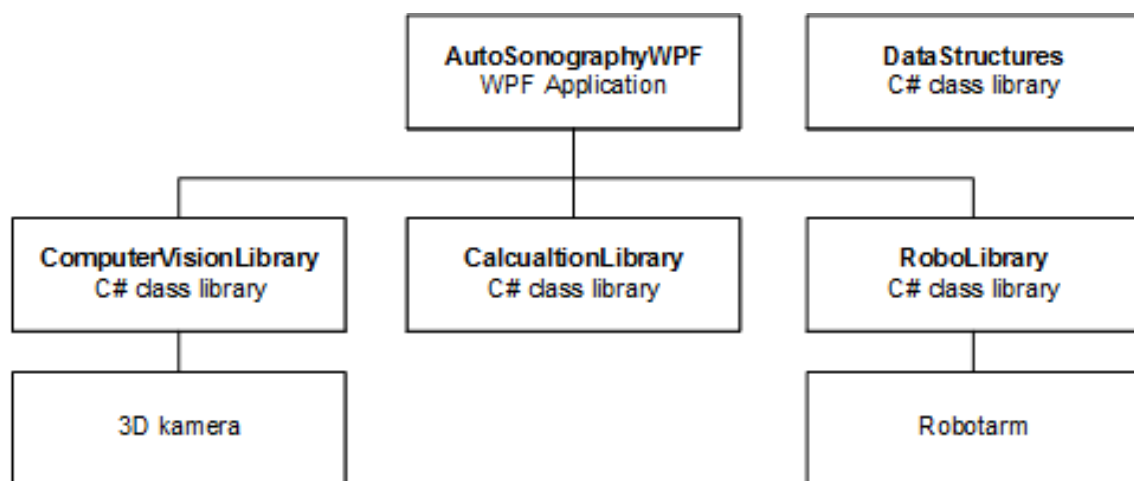
URScript

3.2.2 Kinect

Skriv tekst her om Kinect API'et udviklet af Microsoft

3.3 Softwarearkitektur

PC Applikationen er opdelt af forskellige moduller for at øge samhørigheden og nedsætte koblingen, hvilket er med til at sikre overskuelighed og gøre PC applikation lettere at vedligeholde. Modulerne er inddelt efter ansvarsområder angående præsentation til bruger, Robotarm, 3Dkamera, udregning af robotens sti og datastrukturerne. Figur 3.4 viser referencen mellem modullerne.



Figur 3.4: PC Applikationens opdeling af moduler

Lag	Beskrivelse af ansvar
AutoSonographyWPF	Håndterer Operatørs interaktion med PC Applikation, hvor Operatør kan vælge 3D scan- eller ultralydsscan.
ComputerVisionLibrary	Afgrænser 3D scanning fra 3D kamera.
CalculationLibrary	Sørger for at konvertere 3D scanningen givet fra 3D Kameron rum til Robot Arms rum.
RoboLibrary	Muliggør at kommunikere med Robotarm.
DataStructures	Indeholder diverse data transfer objekter (DTO), der bruges gemmen hele applikationen til at sende objekter mellem de forskellige moduler.

Tabel 3.1: Modulopdeling og ansvar

Systemdesign 4

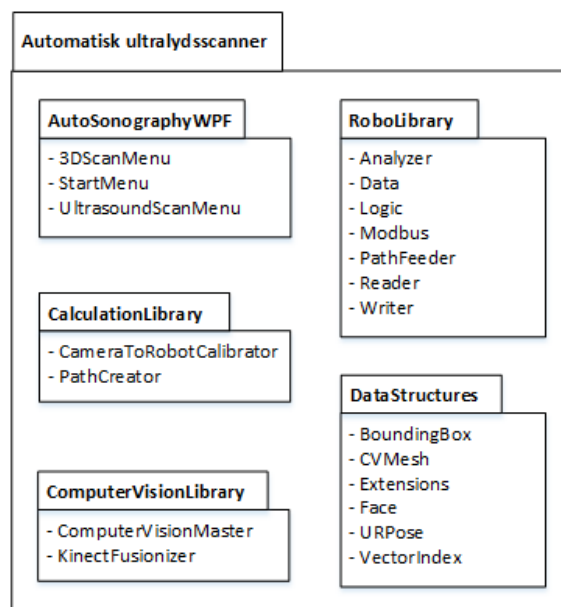
4.1 Pakkediagram

Der er på baggrund af softwarearkitekturen lavet et pakkediagram, som kan ses på nedenstående Figur 4.1. Pakkediagrammet er en opdeling af systemets klasser i forskellige packages. I dette projekt er der valgt tre pakker.

AutoSonographyWPF inderholder alt, der hører til Grafisk brugergrænseflade (GUI), herunder de tre menuer, 3DscanMenu, StartMenu og UltrasoundMenu.

ComputerVisionLibrary indeholder klasser, der har at gøre med 3D billedet og konvetering til et mesh, herunder ComputerVisionMaster, KinectsFusionizer og PLYExporter.

RoboLibray pakken indeholder klasser som robotarmen skal benytte for at kunne flytte sig, herunder Analyzer, Data, Logic, Modbus, PathCreator, PathFeeder, Reader, RoboMaster, URPose og Writer.



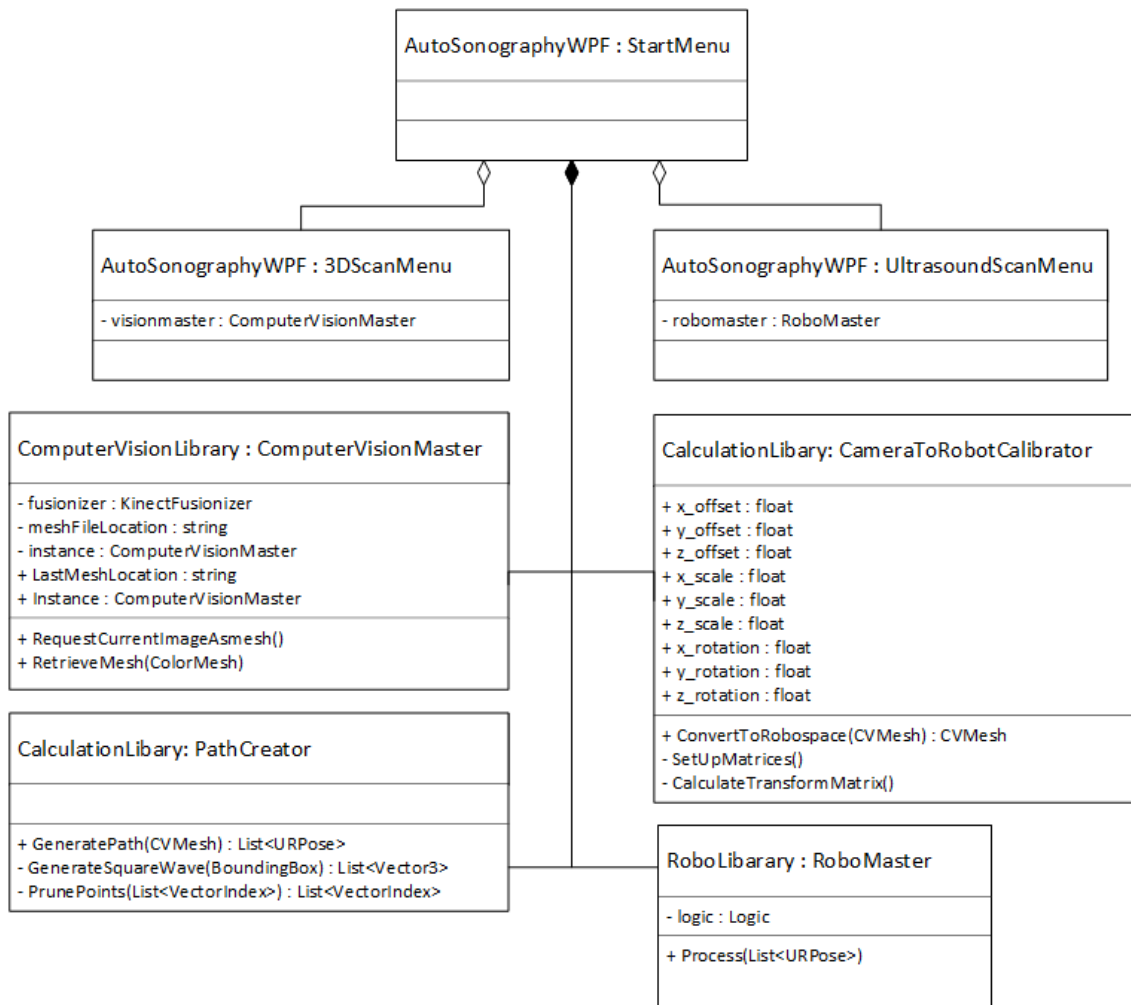
Figur 4.1: Pakkediagram for Automatisk Ultralydsscanner

4.2 Klassediagram

Dette afsnit beskriver klasserne fra pakkediagrammet. Klassediagrammerne viser strukturen i systemet og deres relationer. Hver klasse indeholder de vigtigste metoder og attributer i klassen, der udgør funktionaliteten i System.

4.2.1 GUI

Skriv tekst her...



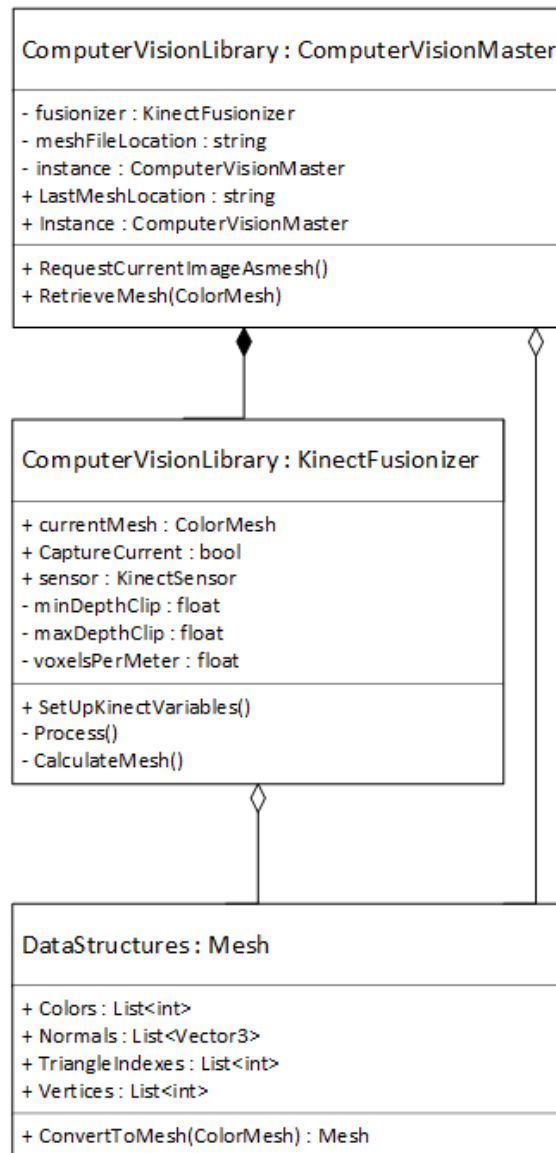
Figur 4.2: KLassediagram for GUI

4.2.2 ComputerVisionLibrary

Formålet med dette bibliotek er at få en afgrænset 3D scanning fra et 3D kamera.

KinectFusionizeren har til ansvar at åbne Kinect-sensoren, tage det nuværende dybdebillede fra sensoren og konvertere det til en mesh.

ComputerVisionMaster virker som den logiske grænseflade til KinectFusionizeren, hvor instansen af den nuværende mesh lagres her. Andre klasser kan subscribe til ComputerVisionMasteren for at høre hvornår der er en ny mesh tilgængelig.

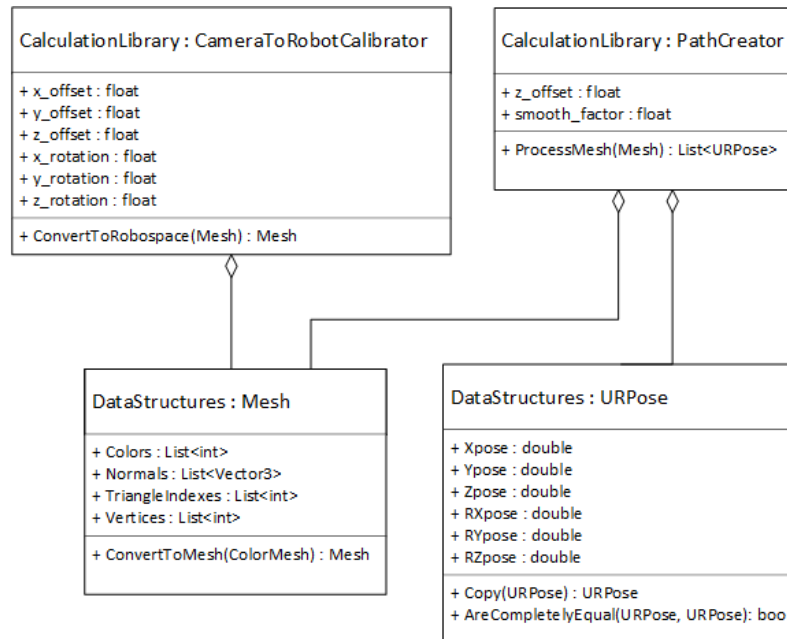


Figur 4.3: Klassesdiagram for ComputerVisionLibrary

4.2.3 ConversionLibrary

Dette bibliotek agerer som bindeledet mellem ComputerVisionLibrary og RoboLibrary. CameraToRobotCalibrator sørger for at konvertere 3D scanningen givet fra ComputerVisionLibrary fra 3D Kameras rum til Robot Arms rum.

Det gør at hvert eneste punkt i modellen kan gives videre til Robot Arm som en positur. Det er nødvendigt at finde ud af hvilken rute Robot Arm skal tage for at ultralydsscanne et objekt - det står PathCreator for. Efter transformation til Robot Arms rum afgør PathCreator listen af positurer som Robot Arm skal løbe igennem.



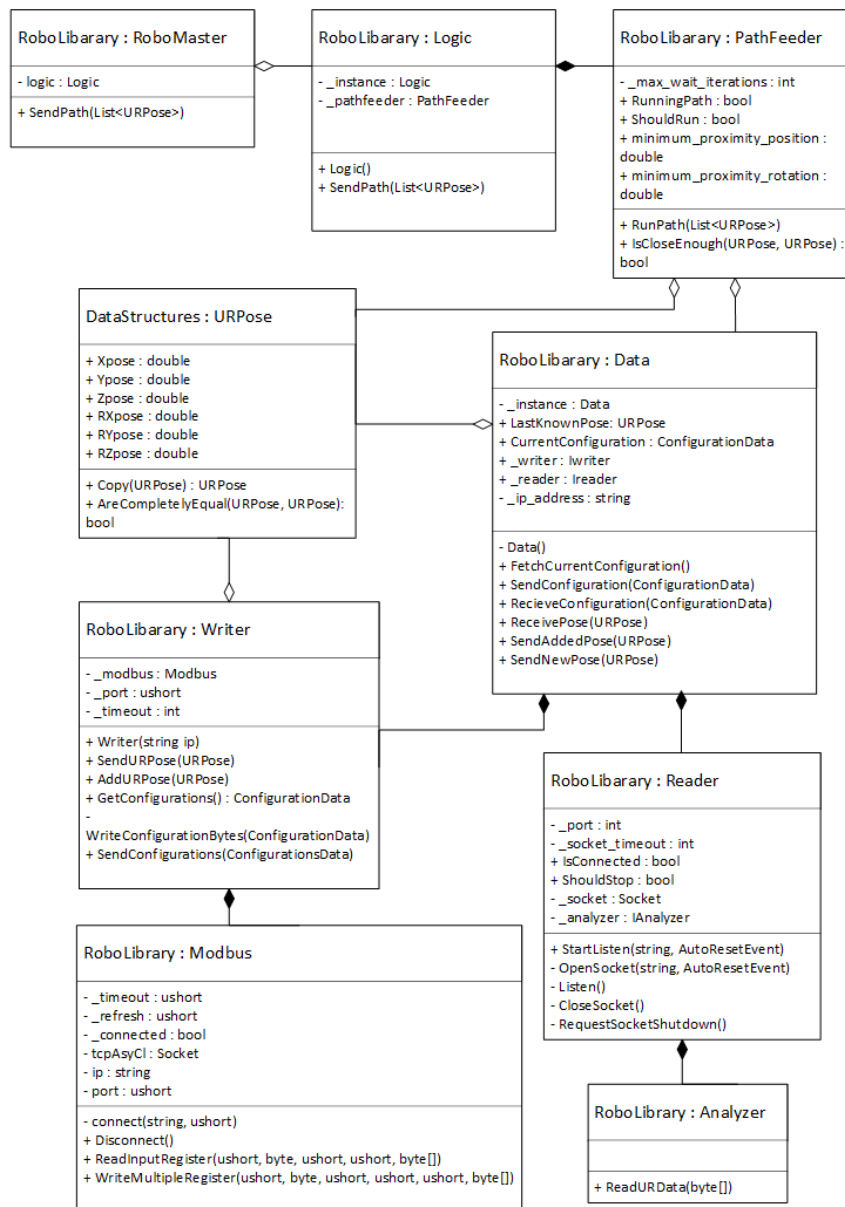
Figur 4.4: Klassesediagram for ConversionLibrary

4.2.4 RoboLibrary

Biblioteket giver mulighed for kommunikation med Robot Arm. I det øverste modul, RoboMaster, af biblioteket vil der være mulighed for at instruere Robot Arm i at gennemløbe en mængde af positurer.

PathFeeder står for at gennemløbe hver positur i listen, og kommunikere med Data for at finde ud af hvornår den næste positur skal sendes til Robot Arm. Data-klassen er der som en grænseflade mellem den 'logiske' del af biblioteket og dens underliggende reader/writer klasser. Reader står for kontinuerligt at læse data fra Robot Arm. Analyser konverterer det indlæste data til en objekt-orienteret model, altså transformation af bytes til Robot Arms nuværende positur.

Writer står for at omskrive værdier til binær data. Modbus skriver binær data ud på Robot Arms IP gennem modbus-protokollen.



Figur 4.5: Klassesdiagram for RoboLibrary

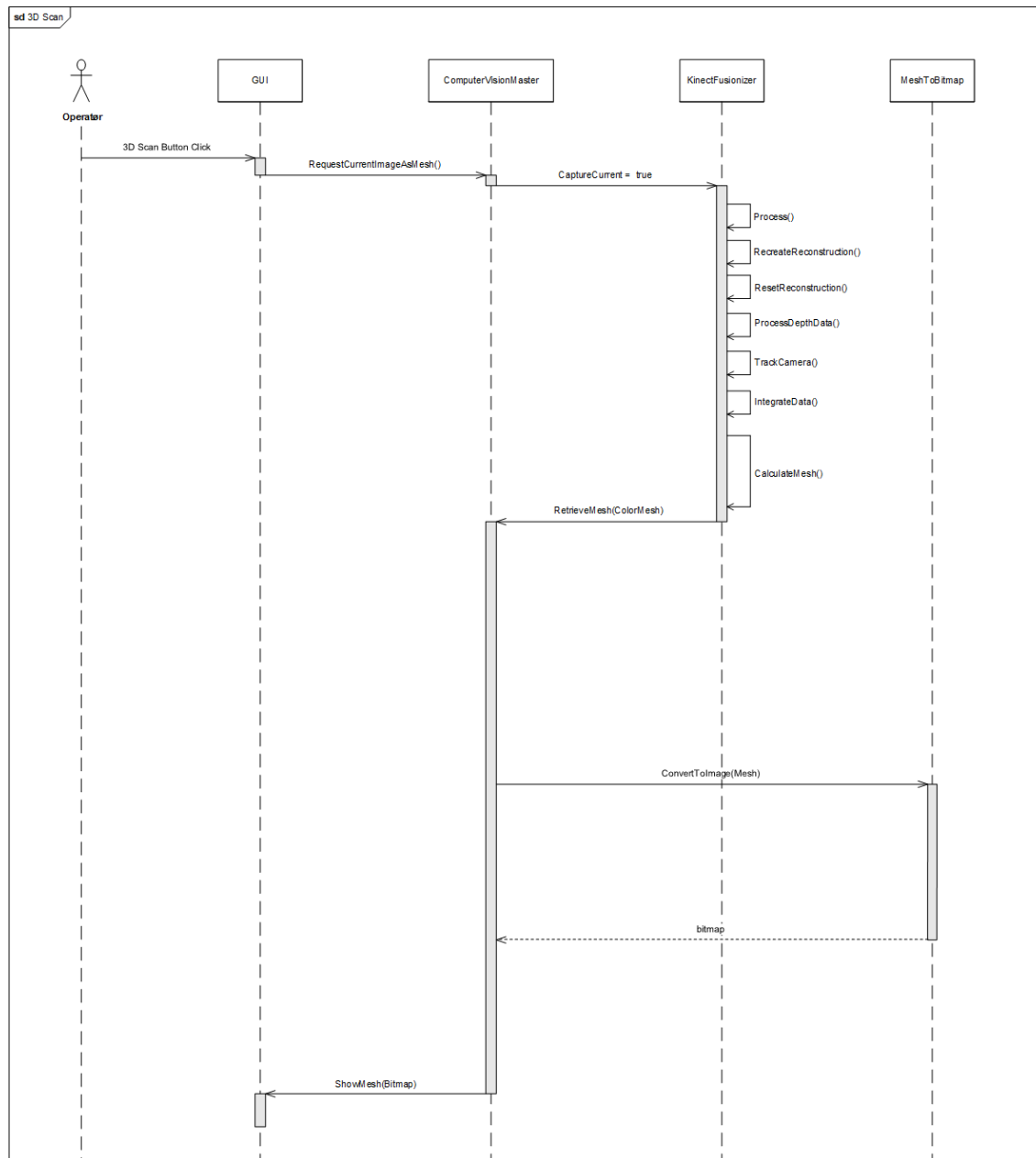
4.3 Sekvensdiagrammer

Der er på baggrund af klassediagrammerne lavet sekvensdiagrammer, som beskriver systemets funktionalitet, og hvor de vigtigste metoder og attributter imellem klasserne er identificeret.

Nedenstående sektioner vil beskrive de vigtigste sekvensdiagrammer i system og fremvise, hvordan klasserne indbyrdes kommunikerer.

4.3.1 3D scan

Ved scan vil KinectFusionizeren stå for at tage et dydbillede og et farvebillede fra Kinect-sensoren. Den vil dernæst konvertere dydbilledet til en point cloud og give hvert punkt i skyen en farve fra farvebilledet. Dette trianguleres, så der fås en mesh der efterfølgende kan bearbejdes. ComputerVisionMaster får meshen tilbage, får lavet et 2D billede af meshen fra en bestemt vinkel og giver begge tilbage GUI. GUI viser billedet og holder på meshen indtil den skal sendes videre til ultralydsscanning



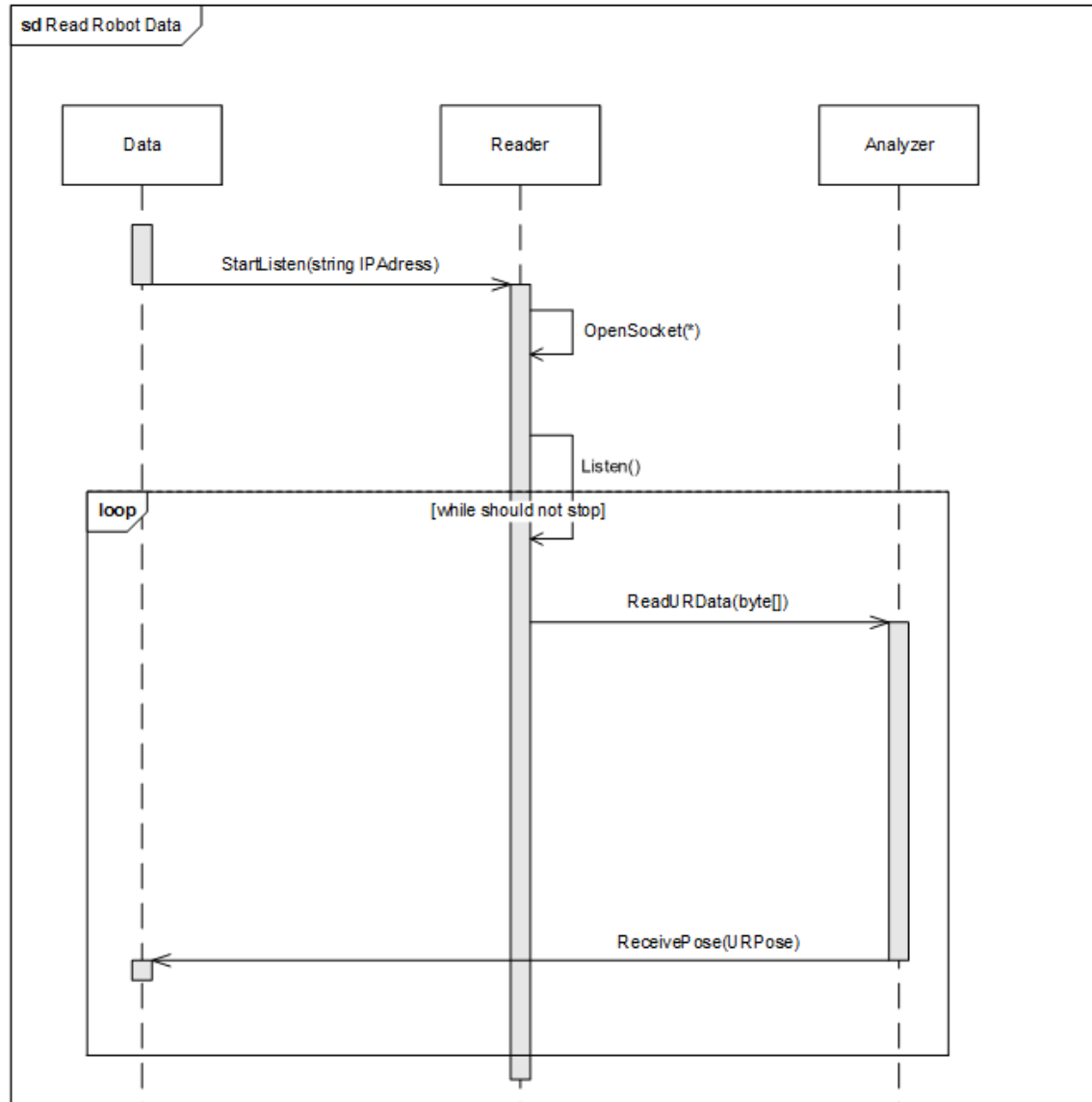
Figur 4.6: Sekvensdiagram for 3D scan

4.3.2 Feed Path

Efter konvertering af robot space og konstateringen af stien er blevet oprettet i PathCreator skal Robot Arm instrueres i at flytte sig gennem punkterne. PathFeeder får en liste af de positurer som den skal gennemløbe. Derefter sendes den næste i listen til Data, som videregiver posituren til Writer. Writer konverterer posituren til binær data, og ModBus skriver dataen ud på Robot Arms register. Hernæst ser PathFeeder på om Robot Arms nuværende positur har nærmet sig den ønskede positur. Når den er tæt nok på, hoppes der ud af 'while'-løkken, og den næste positur kan sendes. Den Alt der er her skal forstås som at PathFeeder kører i sin egen baggrundstråd der kan pauses ved at sætte et flag. Ved terminering af denne baggrundstråd vil PathFeeder stoppe med at videregive nye positurer til Robot Arm.

4.3.3 Indlæsning af Robotarms positur

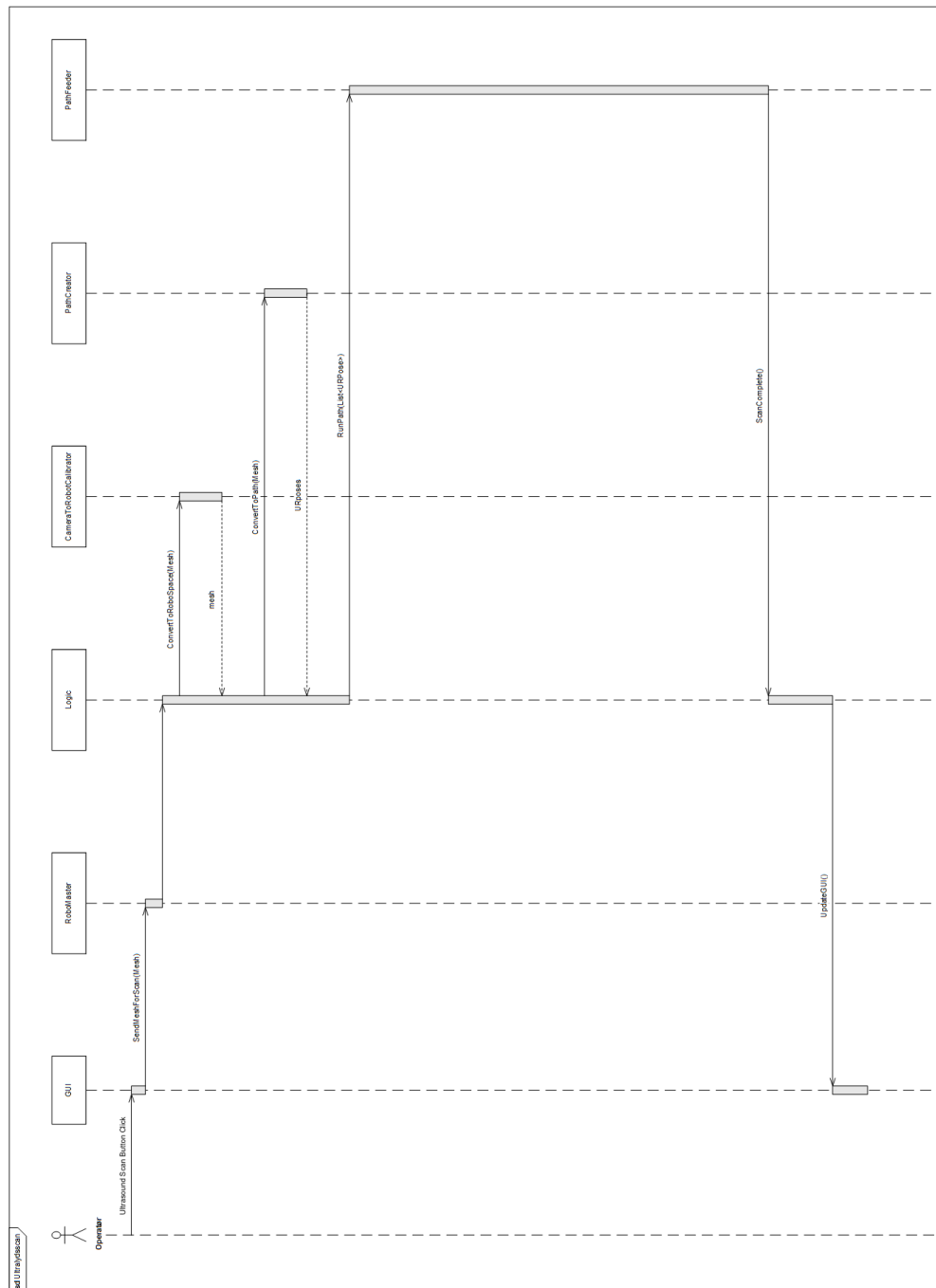
Reader initieres med en IP hvor den skal lytte på. Der åbnes en socket på denne IP, og derefter lytter den kontinuert i en baggrundstråd. Readeren giver de rå data videre til Analyzer som konverterer dem til Robot Arms nuværende positur.



Figur 4.7: Sekvensdiagram for Read Robot Data - indlæsning af Robotarms positur

4.3.4 Ultralydsscan

Ved tryk på Ultralydsscan-knappen sendes den scannede mesh videre til Robomasteren (ÆNDRE PLS). Først skal meshen konverteres fra 3D Kameras rum til Robot Arms rum, dette sker i CameraToRobotCalibrator. Dernæst sendes den konverterede mesh til PathCreator så der oprettes en sti. Med denne sti får Robot Arm afdækket overfladen på Patient, hvis den har en Ultralydsscanner monteret. Stien gives videre til PathFeeder hvor den gennemløbes. Se sekvensdiagrammet REFERENCE for forløbet her. Når scanningen er færdig, vil GUI opdateres så Operatør ved hvornår scanningen er færdig.



Figur 4.8: Sekvensdiagram for Ultralydsscan

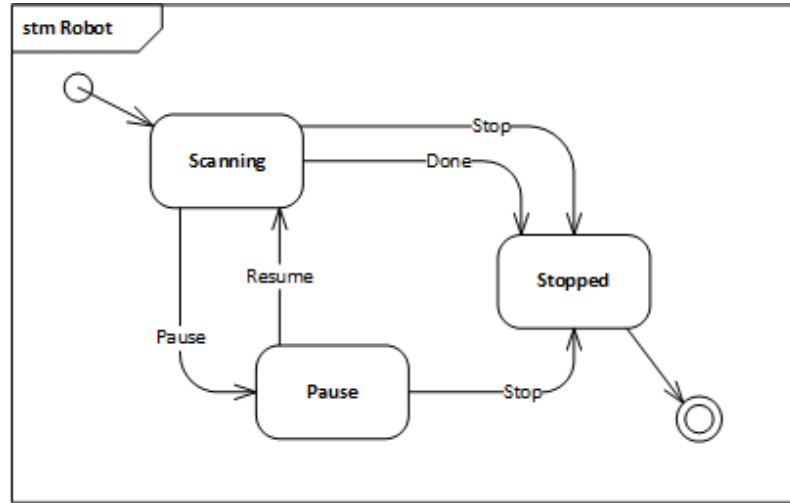
4.4 Detajleret specifikation af klassesdiagrammer

Her skal der vises metoderne i hver klasse.

4.5 Tilstandsdiagram

Dette afsnit beskriver adfærden i systemet ved brug af et tilstandsdiagram. Tilstandsdiagrammet beskriver overgange mellem forskellige tilstande. I UC3: Ultralydsscan brystområde kan Operatør vælge at pause screeningen midlertidig og enten genoptage eller helt

stoppe screeningen. Figur 4.9 beskriver Robotarms forskellige tilstande under udførelse af ultralydsscanning.



Figur 4.9: Tilstandsdiagram for ultralydsscanning

4.6 Beregninger

Bare for at have det et sted og forsøge

$$roll = \begin{cases} \frac{\pi}{2} - \sin^{-1}(z) & y \leq 0 \\ \pi + \cos^{-1}(-z) & y > 0 \end{cases}$$

$$pitch = -(\pi - \text{atan2}(y, x))$$

Er vel det samme som

$$pitch = \text{atan2}(y, x) - \pi$$

$$yaw = 0$$

$$R_M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(roll) & -\sin(roll) \\ 0 & \sin(roll) & \cos(roll) \end{bmatrix}$$

$$P_M = \begin{bmatrix} \cos(pitch) & 0 & \sin(pitch) \\ 0 & 1 & 0 \\ -\sin(pitch) & 0 & \cos(pitch) \end{bmatrix}$$

$$Y_M = \begin{bmatrix} \cos(yaw) & -\sin(yaw) & 0 \\ \sin(yaw) & \cos(yaw) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$roll \geq (-\pi, \pi)$$

$$pitch \geq (-\pi, \pi)$$

$$\mathbb{R} = Y_M \cdot P_M \cdot R_M$$

$$\alpha = \cos^{-1} \left(\frac{\mathbb{R}_{0,0} + \mathbb{R}_{1,1} + \mathbb{R}_{2,2} - 1}{2} \right)$$

$$\theta = \begin{cases} \alpha & roll \geq 0 \wedge pitch \geq 0 \\ 2\pi - \alpha & roll < 0 \vee pitch < 0 \end{cases}$$

$$\mu = \frac{1}{2 * \sin(\theta)}$$

$$r_x = \mu \times (\mathbb{R}_{2,1} - \mathbb{R}_{1,2}) \times \theta$$

$$r_y = \mu \times (\mathbb{R}_{0,2} - \mathbb{R}_{2,0}) \times \theta$$

$$r_z = \mu \times (\mathbb{R}_{1,0} - \mathbb{R}_{0,1}) \times \theta$$

Udviklingsmiljø 5

Der er i nedenstående tabel 5.1 opstillet de programmer og versioner der er brugt til udviklingen af Automatisk Ultralydsscanner.

Emne	Version
Microsoft Visual Studio	2013
MeshLab	1.3.3
Notepad++	7.2.1
Unity	5.1
Google SketchUp	16.1
MathCad	14

Tabel 5.1: Udviklingsmiljø

I projektet blev Microsoft Visual Studio 2013 og kodesproget C# anvendt til at udvikle PC Applikation.

Til debugging er MeshLab og Notepad++ anvendt.

Google SketchUp version 16.1, er et 3D moduleringsprogram, som i projektet er blevet anvendt til beregninger på 3D modeller detekteret af 3D kamera.

Til 3D visualisering af Automatisk Ultralydsscanner blev Unity 5.1 anvendt. Hvilket gjorde det nemmere at finde ud af hvad 3D kameras offset var.

Til matematiske beregninger af Robotarms rotation blev Mathcad 14 anvendt.

Bilag

16

17