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

5. december 2016

Indholdsfortegnelse

Versionshistorik 1

Version	Dato	Ansvarlig	Beskrivelse
1.0	2016-11-01	CSK	Første version af udviklingsdokuement 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 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

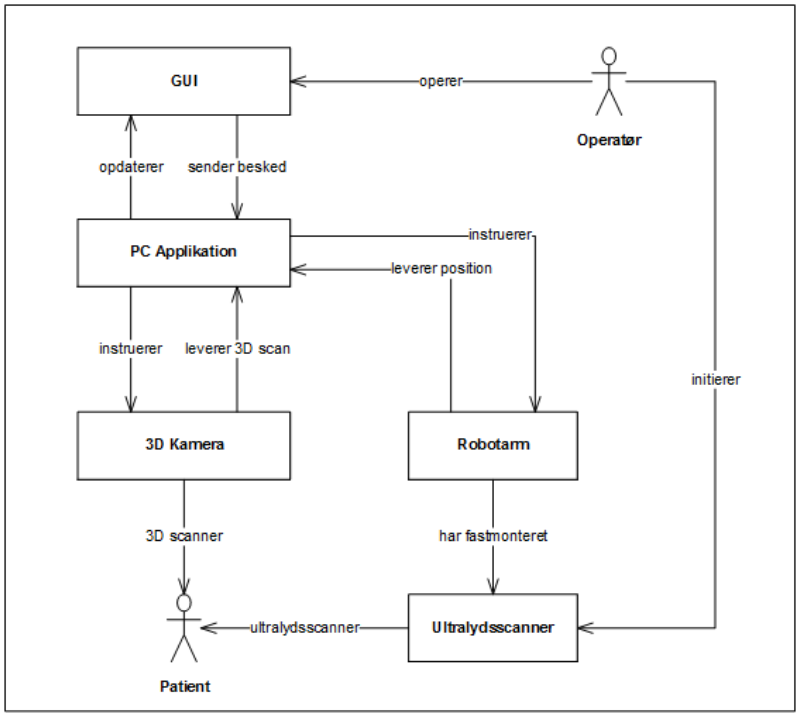
Systemet Automatisk Ultralydsscanner består af en PC applikation, en Ultralydsscanner, en Robotarm af typen Universal Robots UR10, et 3D kamera af typen Microsoft Kinect 2.0, samt et Access Point, af typten D-Link DAP-1160, til forbindelse mellem Robotarm og PC applikation. Der er fem aktører, Robotarm,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

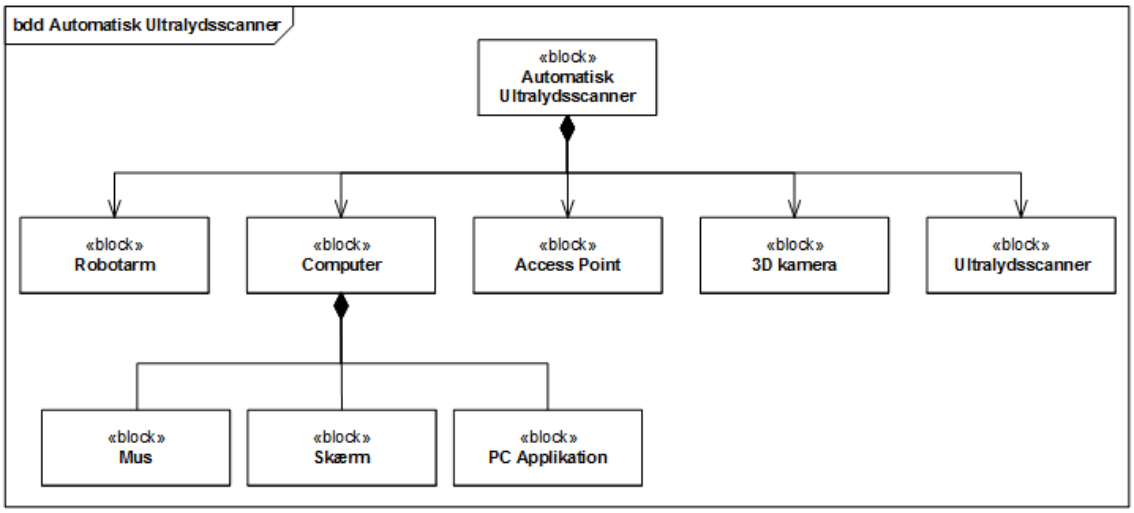
Domænemodellen på Figur 3.1 viser forbindelserne mellem de forskellige aktører i systemet.



Figur 3.1: Domænemodel for Automatisk Ultralydsscanner

3.1.2 Block definition diagram

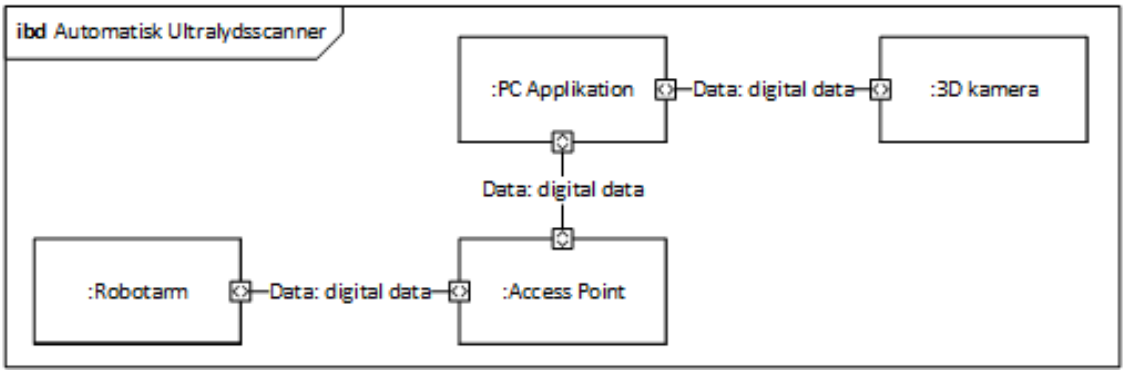
Block Definition Diagram Figur 3.2 viser systemet blokke.



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 grænseflader

Systemet består af to grænseflader, mellem PC Applikation og Kinect og mellem PC Aplikation og UR10. Kommunikationen mellem PC Applikation og UR10 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 i en uendelig løkke 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 reélt har, som ikke nødvendigvis er den samme som den sidste ønskede positur. 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. URScriptet 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 Scriptet 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 *movel*. 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

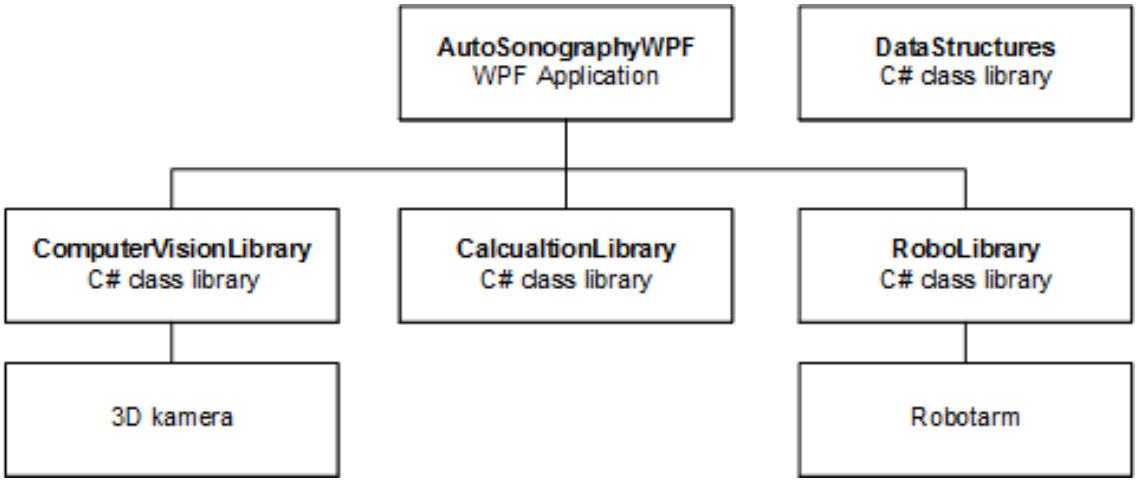
URScript

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 robottens 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 Kameras 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 forsellige moduler.

Tabel 3.1: Modulopdeling og ansvar

Systemdesign 4

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 designet af systemet.

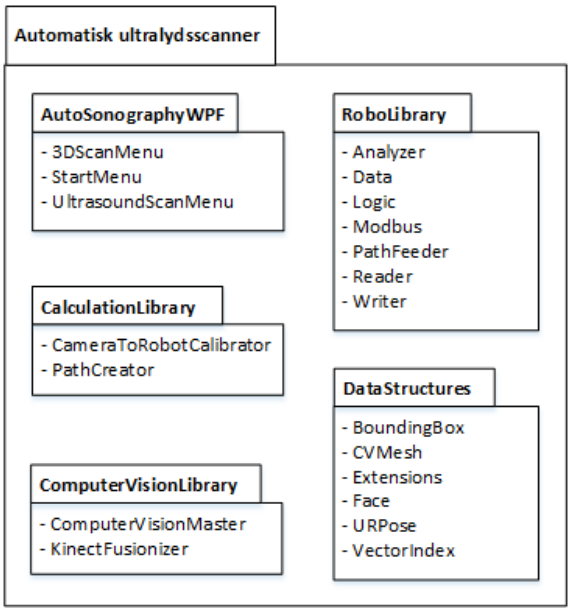
4.1 Pakkediagram

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.

AutoSocograophyWPF 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

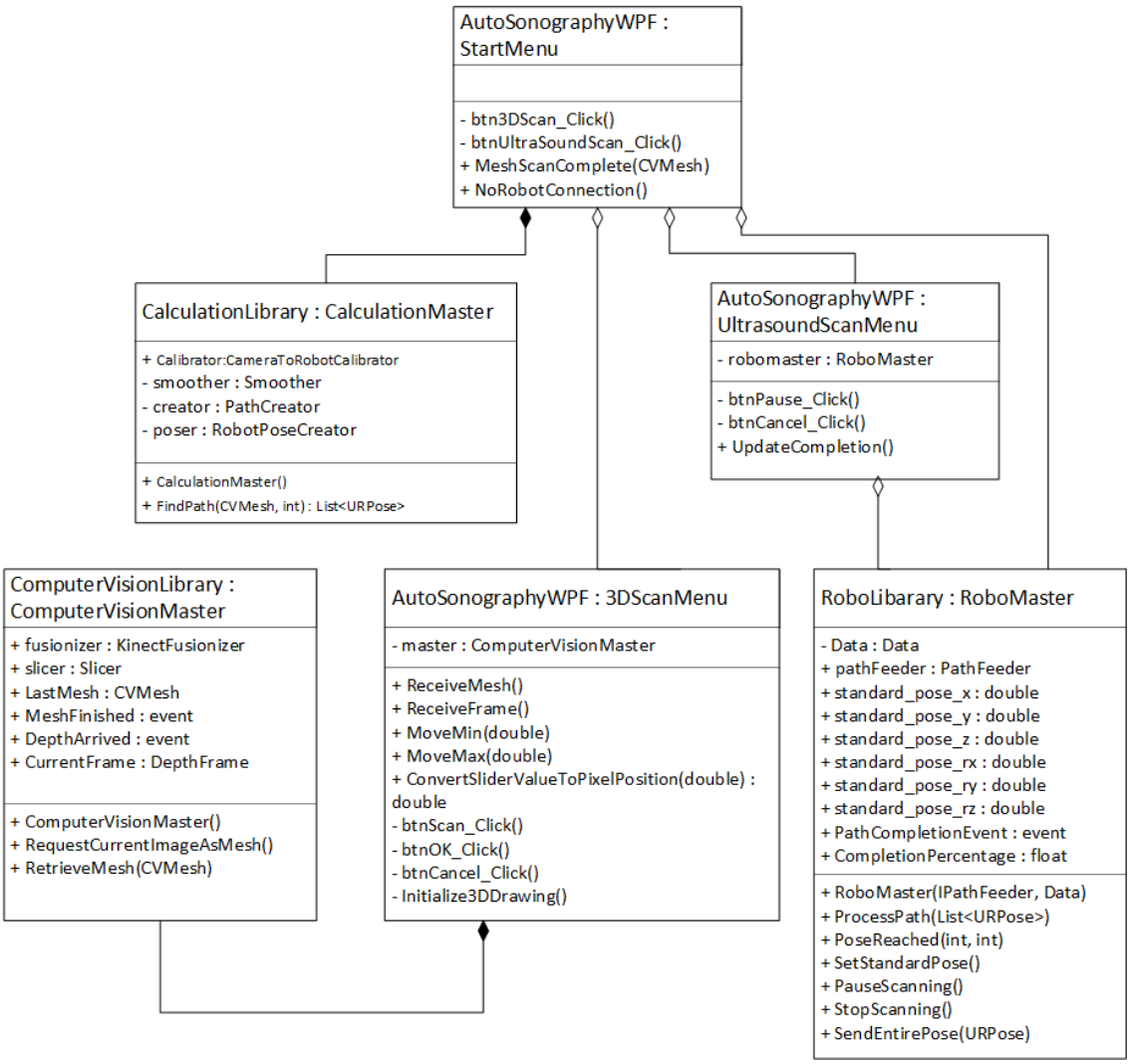
4.2 Klassediagram

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

Dette projekt indeholder brugergrænsefladen af PC Applikation.

- MainWindow Giver anledning til at foretage et 3D scan. Såfremt en 3D scanning er gennemført giver det også anledning til at starte en ultralydsscanning. Når menuen startes, oprettes en instans af RoboMaster, for at sætte Robotarm i standard positur. Dette er nødvendigt, hvis Robotarm skulle være i vejen for en 3D scanning. Hvis der ikke er nogen forbindelse til Robotarm vil der
- 3DScanMenu I denne menu er der mulighed for at se det nuværende dybdebillede, afgrænse området der skal 3D scannes og foretage en 3D scanning.
- UltrasoundScanMenu I denne menu kan den procentvise færdiggørelse af ultralydsscanningen følges. Der er også mulighed for at pause samt afbryde ultralydsscanningsprocessen.

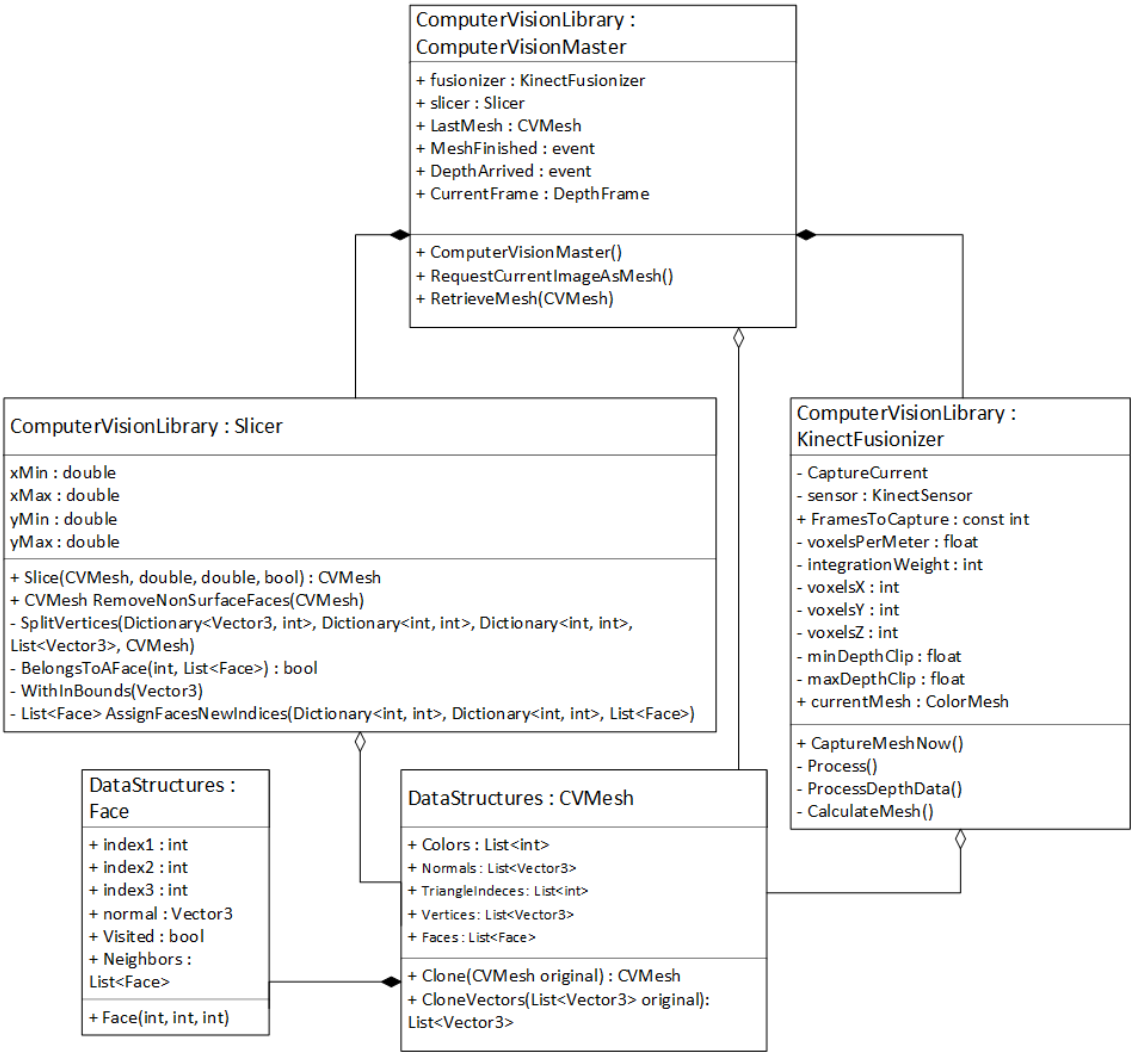


Figur 4.2: KLassediagram for GUI

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 Denne klasse 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.
- Slicer Denne klasse sørger for at fjerne de punkter i en mesh der er uinteressante:
    1. Faces der peger nedaf, dvs fejlpunkter. Da 3D kameraet er monteret i loftet, vil den ikke kunne se undersiden af det den skanner.
    2. Duplikerede punkter. KinectFusionizer outputter punkter der er ens. Disse fjernes af optimeringsårsager.
    3. Punkter og faces der er uden for det område der ønskes skannet. Dette inkluderer nærtliggende objekter som fx en væg, eller områder på patienten der ikke ønskes skannet.

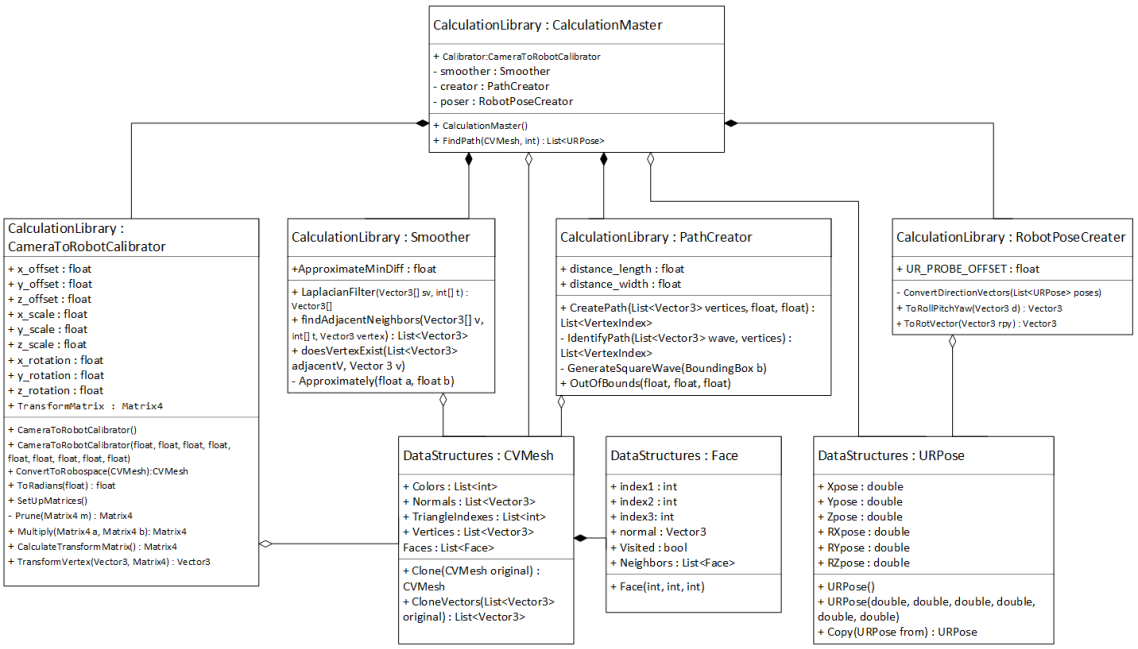


Figur 4.3: KLassediagram for ComputerVisionLibrary

4.2.3 CalculationLibrary

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. Dette sker ved en kæde af matrix-transformationer i en speciel rækkefølge. Normaltvis har man en translation, rotation og skalering, men da Robot Arms og 3D Kameras koordinatsystemer begge er angivet i millimeter, er skaleringen unødvendig. I tilfældet for dette projekt sker der først en rotering og derefter en translation, for at bestemme transformationsmatricen. Hvert punkt i en mesh konverteres så til det nye space.
- Smoother Denne klasse har til ansvar at udjævne en mesh. Med udjævning forstås at 'ensforme' normalerne, altså retningsvektorer i en mesh's faces. Dette er nødvendigt da 3D Kameras output kan være uperfekt og dermed vil normalerne være ekstreme/deforme. Udjævningen sker gennem laplacian smoothing, se [1] for forklaring af algoritme.
- PathCreator Klassen afgør listen af punkter i en mesh som der skal findes positurer til Robot Arm ud fra. For at afgøre stien genereres der en 'bølge' - i implementeringen en squarewave - af punkter der draves over meshen. De vertices i meshen der tilnærmer sig punkterne i bølgen bedst vil blive udvalgt til stien.
- RobotPoseCreator I denne klasse vil konverteringen af en mesh-sti til en liste af positurer ske. For hvert punkt i mesh-stien, vil en vertex' normal findes. Ved hjælp af normalen, sti-punktets koordinater samt længden på Robotarms probe kan den forskudte Robot Arm position findes. Inverteres denne normal, kan det ses som en retningsvektor for en Robotarm. Retningsvektoren konverteres først til en roll, pitch og yaw - altså roteringer omkring de tre retningsakser; X, Y og Z. Da man ikke kan afgøre alle tre værdier ud fra en retningsvektor alene, sættes pitch til 0. Disse værdier konverteres herefter til en rotationsvektor. Positionsvektoren og rotationsvektoren udgør til sammen en positur, som tilføjes til listen af positurer.

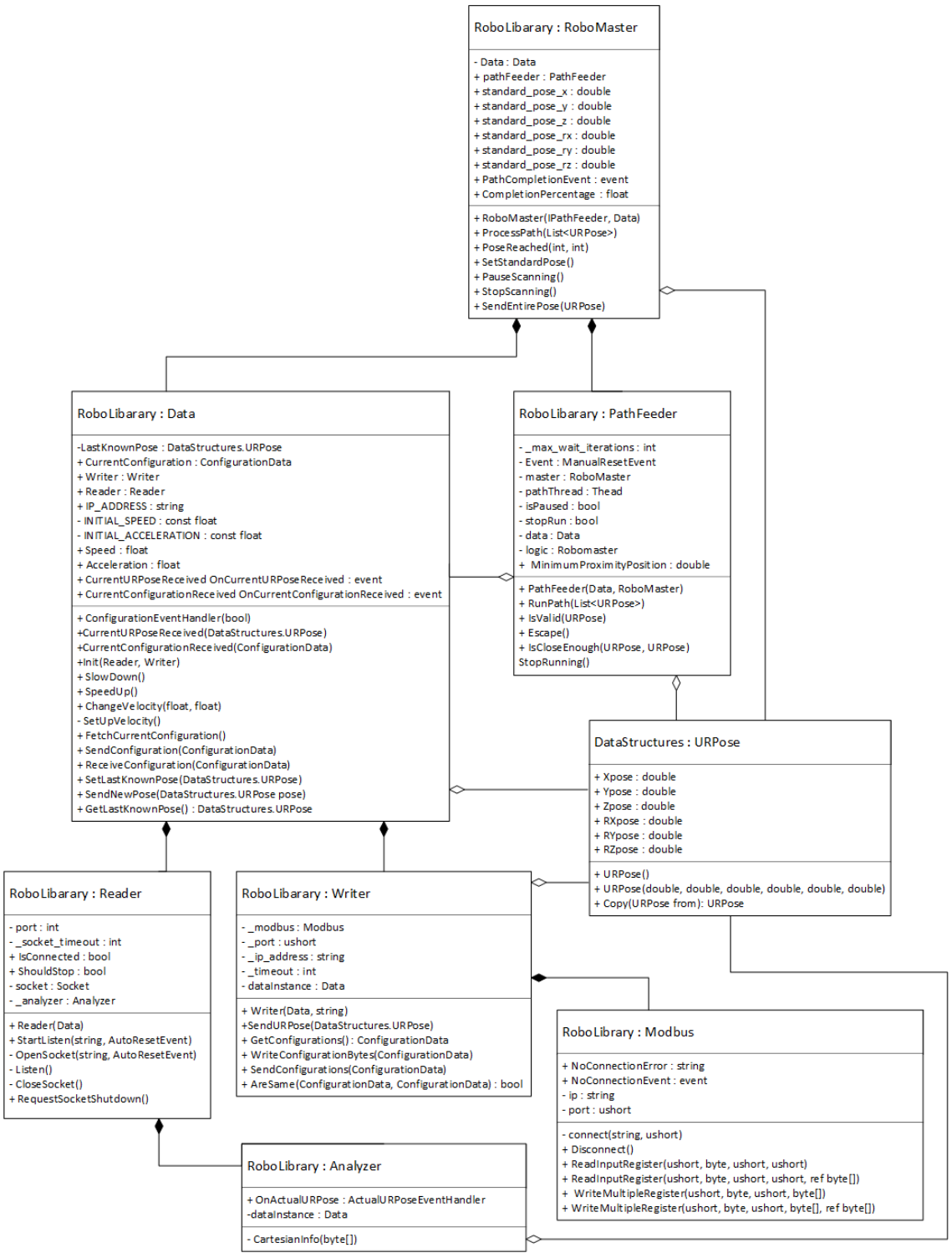


Figur 4.4: KLassediagram for CalculationLibrary

4.2.4 RoboLibrary

Biblioteket giver mulighed for kommunikation med Robot Arm.

- RoboMaster Klassen agerer som bindeled mellem de øvrige klasser i biblioteket og GUI.
- 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.
- klassen Klassen virker som en grænseflade mellem den 'logiske' del af biblioteket og dens underliggende reader/writer klasser.
- Reader Denne klasse står for kontinuerligt at læse data fra Robot Arm, for at afgøre dens nuværende positur.
- Analyzer Klassen konverterer det indlæste data til en objekt-orienteret model, altså transformation af bytes til Robot Arms nuværende positur.
- Writer Klassen har til ansvar at omskrive værdier til binær data. Den omskriver både positurer samt konfigurationer.
- Modbus Denne klasse skriver binær data ud på Robot Arms IP gennem modbus-protokollen.



Figur 4.5: KLassediagram for RoboLibrary

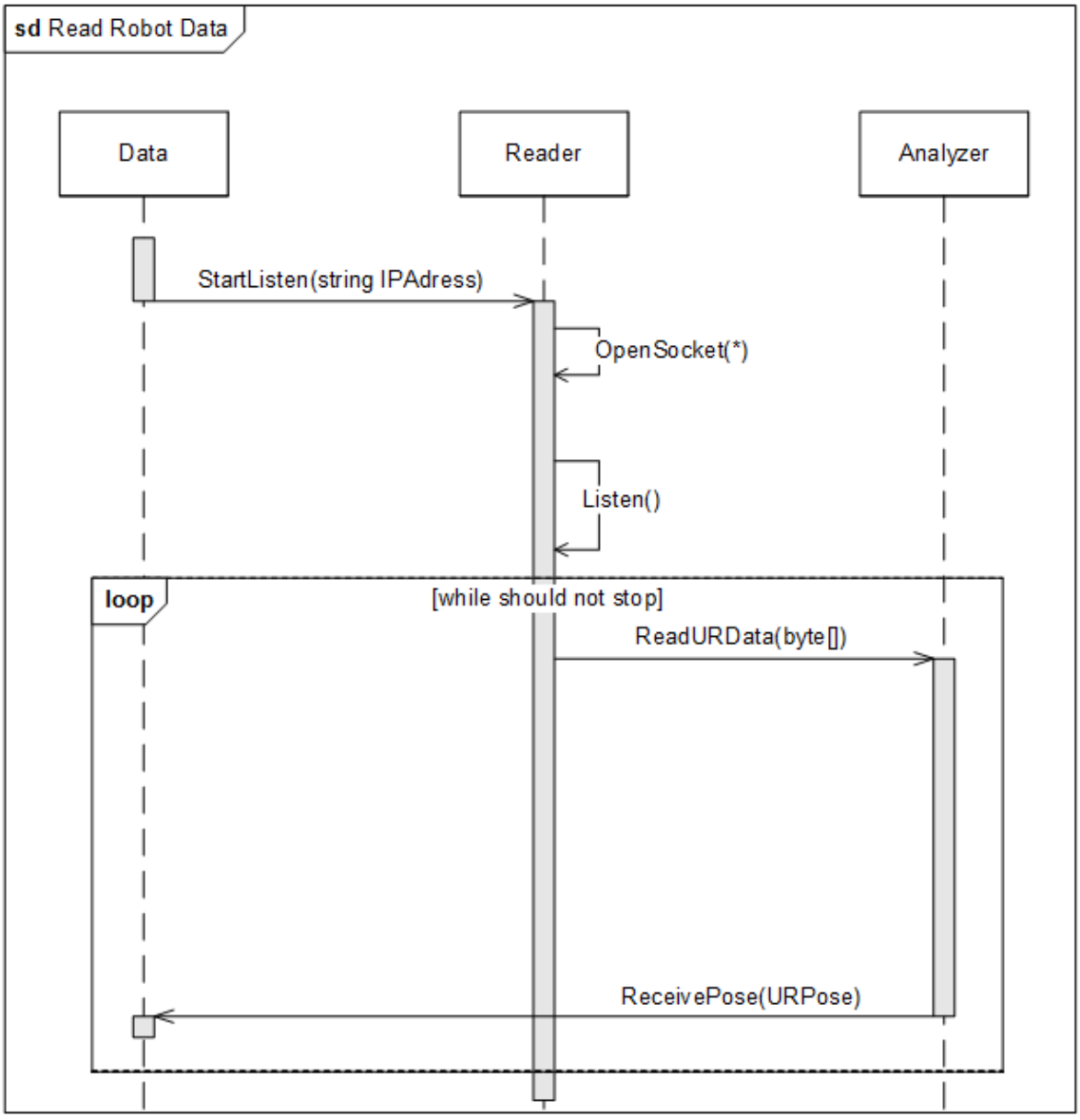
4.3 Sekvensdiagrammer

Der er på baggrund af klassediagrammerne lavet sekvensdiagrammer, som beskriver systemets funktionalitet, og hvor de vigtigse 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 Reading

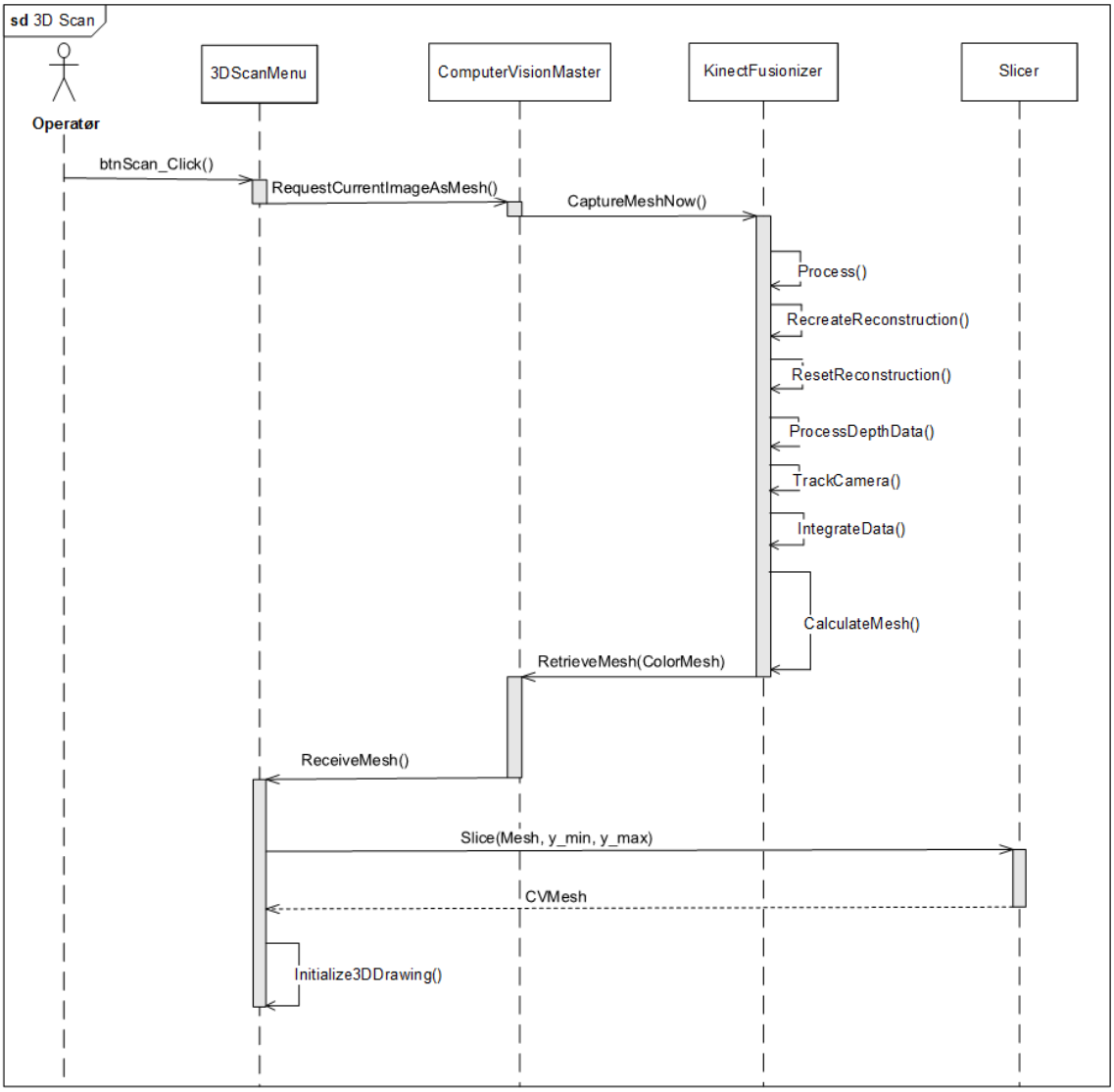
Reader initieres med en IP hvor den skal lytte på. Der åbnes en socket på denne IP, og derefter lytter den kontinuert i en baggrundstråd. Readeren giver de rå data videre til Analyzer som konverterer dem til Robot Arms nuværende positur.



Figur 4.6: Sekvensdiagram for Read Robot Data - indlæsning af Robotarms positur

4.3.2 3D scan

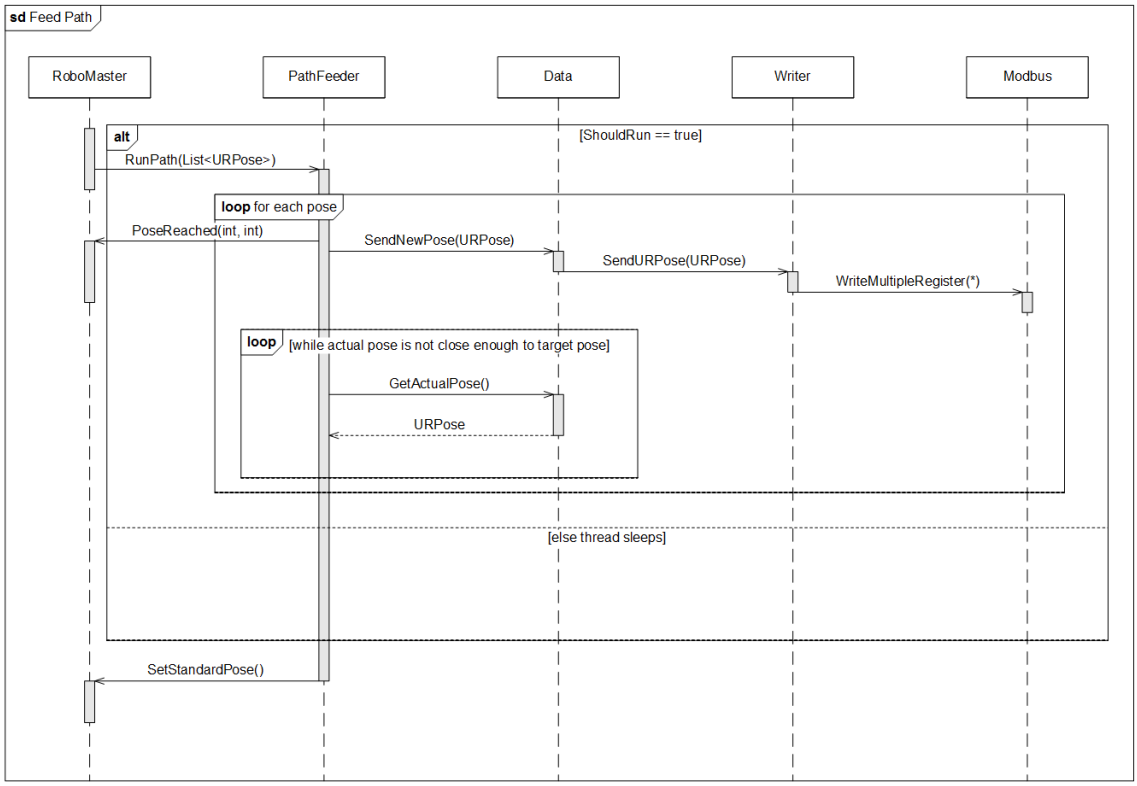
Ved scan vil KinectFusionizeren stå for at tage et dybdebillede fra Kinect-sensoren. Den vil dernæst konvertere dybdebilledet til en point cloud.Dette trianguleres, så der fås en mesh der efterfølgende kan bearbejdes. 3DScanMenu modtager meshen gennem event-kommunikation. Gennem de beskræringsparametre der er valgt i menuen vil meshen dernæst beskæres i Slicer, så kun det ønskede område fremkommer. Efter beskæringen vises meshen som en rotérbar 3D model i menuen.



Figur 4.7: Sekvensdiagram for 3D scan

4.3.3 Feed Path

Efter konvertering af mesh output til robot positur sti, vil listen af positurer sendes fra RoboMaster til PathFeeder. Se sekvensdiagrammet 'Path Creation' for en gennemgang af hvordan disse positurer findes. PathFeeder gennemløber hver positur og sender den næste i listen til Data, som videregiver posituren til Writer. RoboMaster informeres løbende om hvor langt PathFeeder er med at gennemløbe punkter. 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 terminering af denne baggrundstråd vil PathFeeder stoppe med at videregive nye positurer til Robot Arm.

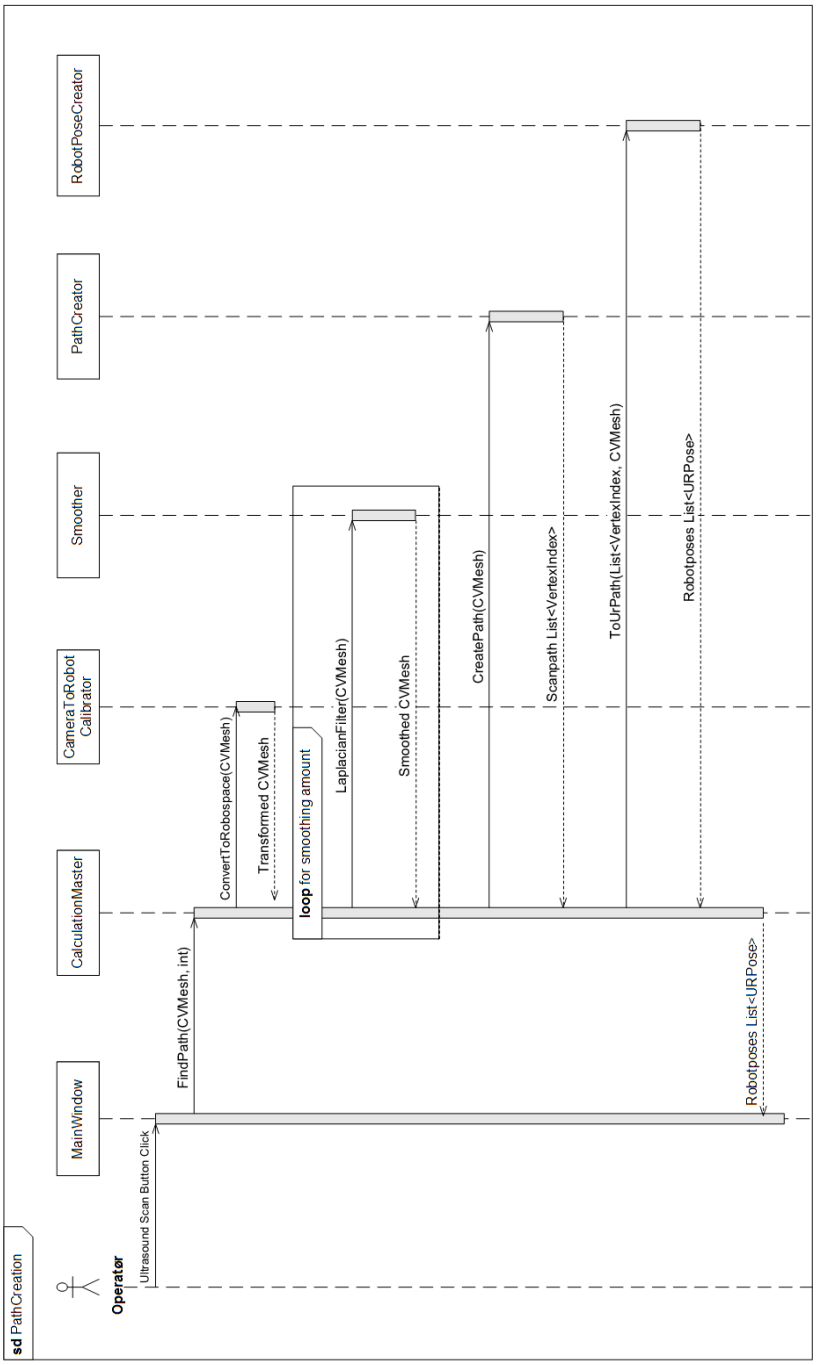


Figur 4.8: Sekvensdiagram for Feed Path

4.3.4 Pathcreation

Ved tryk på Ultralydsscan-knappen sendes den scannede mesh videre til CalculationMaster. Se sekvensdiagrammet '3D scan' for at se hvordan den scannede mesh opnås. Først skal meshen konverteres fra 3D Kameras rum til Robot Arms rum, dette sker i CameraToRobotCalibrator, hvor meshen roteres og translateres ift. hvor 3D Kamera befinder sig i virkeligheden. Efter transformationen skal meshen udjævnes, så de rå og ekstreme normaler i meshen udglattes. Loopet afgør hvor mange gangen meshen skal gennemkøres filteret. Dernæst sendes den konverterede mesh til PathCreator så der oprettes en liste af de punkter i meshen vi ønsker at Robot Arm skal gennemgå. Til sidst skal stien der findes i PathCreator konverteres til Robot Arm positurer, da stien i PathCreator kun fortæller

position direkte på meshen. Med denne sti får Robot Arm afdækket overfladen på Patient, hvis den har en Ultralydsscanner monteret. Stien kan nu videresendes til RoboMaster - se sekvensdiagrammet 'Feed Path'.



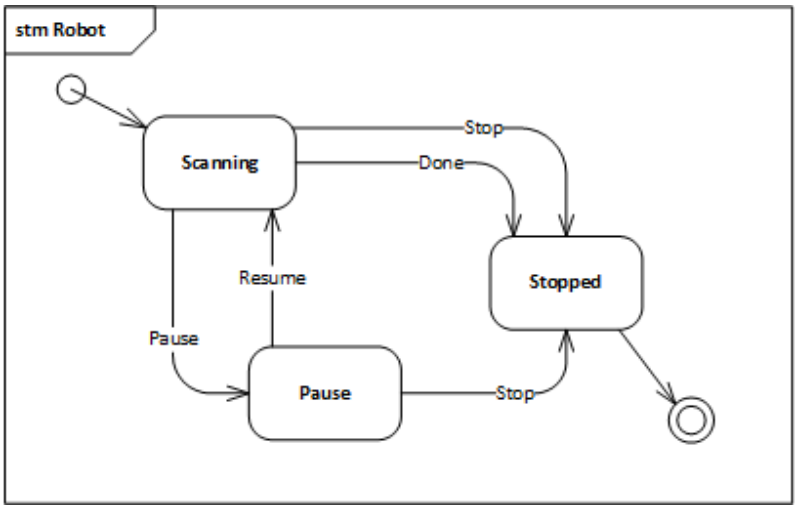
Figur 4.9: Sekvensdiagram for Pathcreation

4.4 Detajleret specifikation af klassediagrammer

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 scanningen midlertidig og enten genoptage eller helt stoppe scanning. Figur 4.10 beskriver Robotarms forskellige tilstande under udførelse af ultralydsscanning.



Figur 4.10: Tilstandsdiagram for ultralydsscanning

4.6 Beregninger

Bare for at have det et sted og forsøge

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

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

Er vel det samme som

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

$$yaw = 0$$

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

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

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

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

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

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

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

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

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

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

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

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

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 moduleringsprogam, 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.

Test 6

6.1 Unittest

6.2 Integrationstest

Bilag

16
17

Litteratur

[1] Wikipedia. Laplacian smoothing. https://en.wikipedia.org/wiki/Laplacian_smoothing. Senest besøgt den 5. december 2016.