

**Faculteit Industriële
Ingenieurswetenschappen**
master in de industriële wetenschappen: informatica
Masterthesis

Een mobiel augmented reality systeem met behulp van een projector

Jan-Mathijs Pex
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: informatica

PROMOTOR :
Prof. dr. Nick MICHIELS

BEGELEIDER :
De heer Joni VANHERCK

Gezamenlijke opleiding UHasselt en KU Leuven



Universiteit Hasselt | Campus Diepenbeek | Faculteit Industriële Ingenieurswetenschappen | Agoralaan Gebouw H - Gebouw B | BE 3590 Diepenbeek

Universiteit Hasselt | Campus Diepenbeek | Agoralaan Gebouw D | BE 3590 Diepenbeek
Universiteit Hasselt | Campus Hasselt | Martelarenlaan 42 | BE 3500 Hasselt



2024
2025

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: informatica

Masterthesis

Een mobiel augmented reality systeem met behulp van een projector

Jan-Mathijs Pex

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: informatica

PROMOTOR :

Prof. dr. Nick MICHIELS

BEGELEIDER :

De heer Joni VANHERCK



KU LEUVEN

Woord vooraf

Sinds ik het vak Visual Computing volgde, heb ik altijd een grote interesse gehad in alles wat met visuele technologieën te maken heeft, zoals AMVR en projection mapping. Het fascineert me om digitale beelden tot leven te brengen in de echte wereld, maar ook hoe het werkt en de technologie die erachter zit. Dit project gaf me de kans om er veel over bij te leren en nieuwe technologieën te ontdekken die hiervoor gebruikt kunnen worden.

Ik wil graag mijn promotor, **prof. dr. Nick Michiels**, bedanken voor zijn begeleiding en constructieve feedback gedurende het hele proces. Ook wil ik mijn begeleider **Joni Vanherck** bedanken voor zijn onmisbare hulp en ondersteuning bij de praktische uitwerking van deze masterproef.

Daarnaast gaat mijn dank uit naar prof. dr. Lode Jorissen en Brent Zoomers, die me hebben bijgestaan met hun advies en inzicht, ook al waren ze niet direct betrokken bij dit project.

Mijn vrienden van de universiteit wil ik bedanken voor hun steun en de technische inzichten die ze soms konden geven. Eveneens mijn vrienden buiten school die me, vaak op afstand, hebben gesteund gedurende deze periode.

Tot slot wil ik mijn naaste familie bedanken voor hun voortdurende steun en vertrouwen. Een speciale dank aan mijn ouders en zus, die me altijd hebben aangemoedigd en geholpen wanneer dat nodig was.

Inhoudsopgave

Woord vooraf	1
Lijst met tabellen	7
Lijst met figuren	10
Abstract	11
Abstract in English	13
1 Inleiding	15
1.1 Doelstellingen	16
2 Bronnenstudie	17
2.1 Marker-gebaseerde tracking	17
2.2 Diepte-gebaseerde tracking	18
2.3 SLAM-gebaseerde tracking	19
2.3.1 Vergelijkende analyse van trackingmethodes	20
3 Methodologie	23
3.1 Gebruikte hardware en software	23
3.2 Kalibratie	24
3.2.1 Camerakalibratie	24
3.2.2 Projector- en projector-camerakalibratie	25
3.3 Trackingmethodes	26
3.3.1 Tracking met ArUco-markers	26
3.3.2 Dieptegebaseerde tracking	27
3.3.3 SLAM-gebaseerde tracking	28
3.4 Visualisatie	29
3.5 Communicatie	29
3.6 Evaluatie	30
4 Kalibratie	31
4.1 Camera kalibratie	31
4.1.1 Kalibratieproces	31
4.1.2 Reprojection error	32
4.1.3 Intrinsieke matrix	33
4.1.4 Distortiecoëfficiënten	33

4.2	Projector-camera kalibratie	34
4.2.1	Verschillen met camerakalibratie en patroonkeuze	34
4.2.2	Detecteren van patronen	35
4.2.3	Intrinsieke kalibratie	35
4.2.4	Perspective-n-point-probleem	36
4.2.5	Extrinsieke kalibratie	37
4.2.6	Extrinsieke matrix	37
5	Tracking	39
5.1	ArUco-markertracking	39
5.1.1	Detectie	39
5.1.2	Pose estimation	40
5.1.3	Visualisatie	40
5.2	Dieptegebaseerde tracking	41
5.2.1	3D-model omvormen naar puntenwolk	41
5.2.2	Puntenwolkregistratie met dieptesensor	41
5.2.3	Puntenwolken matchen	41
5.2.4	Visualisatie	42
5.2.5	Beperkingen	42
5.3	SLAM-gebaseerde tracking	42
5.3.1	SLAM-initialisatie en markerconfiguratie	42
5.3.2	Transformatie tussen SLAM en object	43
5.3.3	Berekenen van de uiteindelijke transformatie	44
5.3.4	Visualisatie	44
6	Visualisatie	45
6.1	Renderer	45
6.1.1	Projectiematrix	46
6.1.2	Objectpose	46
6.2	3D-model	46
6.2.1	3D-tekenen en printen	46
6.2.2	Preprocessing van 3D-model	48
6.2.3	3D-model inladen	49
7	Evaluatie	51
7.1	Voorbereiden van Qualisys-systeem	51
7.2	Gegevensverzameling	52
7.3	Datapreprocessing	53
7.4	Foutberekening	53
7.5	Analyse van de latency en frequentie	53
7.5.1	Latency	53
7.5.2	Frames per seconde	54
8	Resultaten	55
8.1	Empirische Evaluatie	55
8.1.1	Marker-gebaseerde tracking	55
8.1.2	SLAM-gebaseerde tracking	56

8.1.3	Bewegingstrajecten	58
8.2	Kwantitatieve Metingen	59
8.2.1	Positionele fout	60
8.2.2	Real time prestaties	60
8.3	Discussie	60
9	Besluit	63
	Dankwoord	65
	Literatuurlijst	68

Lijst van tabellen

2.1	Vergelijking van trackingmethodes voor dynamische projection mapping.	21
8.1	Prestaties van beide trackingmethodes, met en zonder projectie.	59

Lijst van figuren

1.1	Verschillende moderne AR-technologieën.	15
1.2	Projection mapping op het Disney-kasteel [1].	16
2.1	Verschillende soorten fiduciële markers [2].	17
2.2	Twee methodes om vervormbare oppervlakken te tracken.	18
2.3	Visuele representatie van een globaal registratie-algoritme.	19
2.4	Een SLAM-map van een gebouw, gemaakt met OpenVSLAM [3].	20
3.1	Een schematische voorstelling van een camerakalibratie.	25
3.2	Een ChArUco-bord gebruikt voor kalibratie van verschillende soorten camera's. . .	25
3.3	Schematische voorstelling van de projector-cameraopstelling tijdens kalibratie. Het ChArUco-bord met projectiezone wordt door de camera waargenomen, terwijl de projector het <i>dot-matrix</i> patroon projecteert.	26
3.4	Het ArUco-bord dat getrackt wordt in zowel de digitale voorstelling als het fysieke object.	27
3.5	Het projector-camerasysteem met retro-reflectieve markers voor tracking met een motion-capturesysteem.	30
4.1	Voorbeeldopstelling voor camerakalibratie met een ChArUco-bord in verschillende poses.	32
4.2	Een schematische voorstelling van de <i>reprojection error</i> [4].	32
4.3	Visueel effect van radiale vervorming [5].	33
4.4	Visueel effect van tangentiële vervorming [6].	34
4.5	Verschillende gebruikte <i>dot-matrix</i> patronen voor projector-camera-kalibratie. . .	35
4.6	Voorbeeldopstelling voor projector-camera-kalibratie met een kalibratiebord in verschillende poses.	36
4.7	Een overzicht van de oplossing voor een PnP-probleem [7].	37
5.1	Een gedetecteerde ArUco-marker, aangeduid met de groene rand.	40
5.2	Tracken van een ArUco-markerbord, gekoppeld aan de visualisatie.	40
5.3	Schematische voorstelling van hoe de transformatiematrix tussen het SLAM-coördinaatsysteem en het objectcoördinaatsysteem gevonden kan worden.	43
6.1	Overzicht van de visualisatiepipeline: data vanuit tracking wordt doorgestuurd naar de OpenGL-renderer die de visualisatie aanpast.	45
6.2	Het finale ontwerp van het te tracken object.	47
6.3	De Inventor-plugin die gebruikt werd voor het maken en 3D-printen van het model. .	47
6.4	De textuur die gemaakt werd in Blender en gebruikt wordt in de visualisatie. . . .	48

7.1	Het kalibratieassenstelsel en de <i>calibration wand</i> die gebruikt worden om een Qualisys-systeem te kalibreren.	51
7.2	Een voorbeeld van een <i>rigid-body</i> die getracket wordt door een Qualisys-systeem. .	52
8.1	Screenshot van ArUco-markertracking tijdens beweging van het systeem.	55
8.2	Screenshot van ArUco-markertracking terwijl onvoldoende markers gevonden zijn voor robuuste pose-inschatting.	56
8.3	Screenshot van SLAM-tracking tijdens het systeem op dezelfde plaats gehouden wordt.	57
8.4	Screenshot van SLAM-tracking tijdens snellere beweging.	57
8.5	Vergelijking tussen het traject van het Qualisys-systeem en de ArUco-markertracking.	58
8.6	Vergelijking tussen het traject van het Qualisys-systeem en de SLAM-tracking. . .	59

Abstract

Projector-camera (ProCam)-systemen worden breed ingezet in toepassingen zoals dynamische projectiemapping, augmented reality en mens-computerinteractie. Voor deze toepassingen is nauwkeurige en robuuste objecttracking cruciaal om aanpassing aan beweging of veranderende omgevingen mogelijk te maken. Deze masterproef onderzoekt en vergelijkt drie trackingmethoden voor realtime ProCam-opstellingen met als doel het identificeren van de meest geschikte methode voor specifieke toepassingsscenario's.

De evaluatie omvat ArUco-markertracking, diepteregistratie via KISS-Matcher en visuele SLAM. Deze methoden worden getest via gestandaardiseerde experimenten in realistische omgevingen en gesimuleerde tests waarbij nauwkeurigheid, robuustheid tegenover occlusie en beweging, en praktische inzetbaarheid in dynamische projectieomgevingen worden gemeten.

De resultaten tonen aan dat markergebaseerde technieken een precisie van gemiddeld 18,79 mm bieden maar fysieke plaatsing en continue zichtbaarheid vereisen, terwijl SLAM-alternatieven markerloos werken maar 40% minder beelden per seconde kunnen verwerken. Uit dit onderzoek blijkt dat de optimale trackingmethode afhangt van toepassingsspecifieke factoren zoals de beschikbare hardware, belichtingsomstandigheden en eigenschappen van het projectie-oppervlak. Deze masterproef biedt praktische richtlijnen voor het selecteren van de meest geschikte trackingbenadering binnen ProCam-systemen in verschillende gebruiksscenario's.

Abstract in English

Projector-camera (ProCam) systems are widely used in applications such as dynamic projection mapping, augmented reality, and human-computer interaction. For these applications, accurate and robust object tracking is crucial to enable adaptation to movement or changing environments. This master's thesis examines and compares three tracking methods for real-time ProCam setups with the aim of identifying the most suitable method for specific application scenarios.

The evaluation includes ArUco marker tracking, depth registration using KISS-Matcher and visual SLAM. These methods are tested through standardized experiments in realistic environments and simulated tests measuring accuracy, robustness against occlusion and movement, and practical applicability in dynamic projection environments.

The results show that marker-based techniques offer an average precision of 18.79 mm but require physical placement and continuous visibility, while SLAM alternatives work without markers but can process 40% fewer frames per second. This research demonstrates that the optimal tracking method depends on application-specific factors such as available hardware, lighting conditions, and properties of the projection surface. This master's thesis provides practical guidelines for selecting the most suitable tracking approach within ProCam systems in various usage scenarios.

Hoofdstuk 1

Inleiding

Augmented reality (AR) is een technologie waarbij digitale informatie geïntegreerd wordt in de fysieke wereld, met toepassingen die zich uitstrekken over uiteenlopende domeinen zoals medische beeldvorming, architecturale visualisatie en entertainment. Binnen AR bestaan er verschillende implementatietechnieken, waaronder optische see-through-brillen zoals de HoloLens (zie figuur 1.1a), video see-through-systemen, handheld AR via mobiele apparaten (zie figuur 1.1b), en projectiegebaseerde AR. In deze laatste categorie wordt digitale inhoud rechtstreeks geprojecteerd op fysieke oppervlakken in de omgeving. Deze techniek staat bekend als projection mapping.



(a) De Microsoft HoloLens, die virtuele elementen in de echte wereld kan plaatsen via het scherm in de bril [8].



(b) De AR-functionaliteit in een smartphone die het mogelijk maakt om een Pokémon in de echte wereld te plaatsen [9].

Figuur 1.1: Verschillende moderne AR-technologieën.

Projection mapping wordt traditioneel gebruikt in artistieke en commerciële contexten, bijvoorbeeld bij lichtshows op gebouwen of interactieve installaties. Figuur 1.2 toont hoe Disney projection mapping gebruikt in zijn parken. Traditioneel betreft dit statische projection mapping, waarbij zowel de projectie als het doeloppervlak zich op een vaste plaats in de omgeving bevinden. Recent is er groeiende belangstelling voor dynamische projection mapping, waarbij het doeloppervlak, de projector of beide zich vrij kunnen bewegen. Dit maakt een flexibelere interactie tussen de gebruiker en het systeem mogelijk.

Dynamische projection mapping brengt echter een aantal technische uitdagingen met zich mee, waarbij nauwkeurige tracking één van de voornaamste is. Om een gealigneerde projectie te behouden terwijl het object of de projector beweegt, moet de pose van het object ten allen



Figuur 1.2: Projection mapping op het Disney-kasteel [1].

tijde accuraat en met voldoende hoge frequentie bepaald worden. Hiervoor zijn verschillende trackingmethodes beschikbaar, elk met eigen voordelen en beperkingen.

1.1 Doelstellingen

Dit onderzoek situeert zich binnen dat kader en focust op het bepalen van de meest geschikte trackingtechniek voor dynamische projection mapping op maquettes. Het doel is om een relevant overzicht te geven van geschikte technieken voor projection mapping op schaalmodellen, zoals maquettes. De drie onderzochte methodes zijn markergebaseerde tracking (met ArUco-markers), dieptegebaseerde registratie op basis van puntenwolkalignatie, en SLAM-gebaseerde tracking. Uiteindelijk is besloten om enkel markergebaseerde en SLAM-gebaseerde tracking effectief te evalueren, omdat de dieptegebaseerde aanpak onvoldoende robuust bleek in de praktijk.

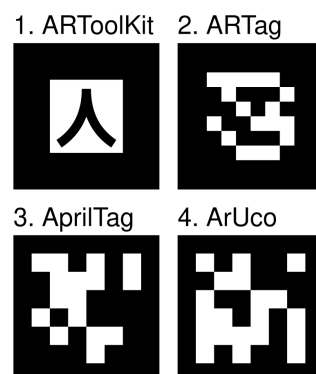
Het onderzoek werd uitgevoerd op een uniforme testopstelling waarbij alle systemen in C++ geïmplementeerd zijn en draaien op identieke hardware. De trackingresultaten worden geëvalueerd met behulp van een nauwkeurig motion-capturesysteem (Qualisys) dat dient als *ground truth*. De verzamelde data maken het mogelijk om de trackingnauwkeurigheid, stabiliteit en consistentie van beide methodes te analyseren, evenals hun prestaties in real time omstandigheden.

Het uiteindelijke doel is om vast te stellen welke methode het meest geschikt is voor dynamische projection mapping op schaalmodellen. Hierbij worden factoren zoals snelheid, nauwkeurigheid en complexiteit van implementatie meegenomen in de evaluatiecriteria.

Hoofdstuk 2

Bronnenstudie

Tal van methodes zijn beschikbaar voor het tracken van een object of van het projector-camerasysteem. Dit overzicht begint met het bekijken van objecttracking met behulp van fiduciële markers en een dieptegebaseerde techniek. Een fiduciële marker is een visueel herkenbaar patroon dat gebruikt wordt om de positie en oriëntatie van een object ten opzichte van een camerasysteem te bepalen. In figuur 2.1 worden verschillende soorten fiduciële markers weergegeven. Daarna wordt er een techniek genaamd *Simultaneous Localization And Mapping* (SLAM) besproken. SLAM is een techniek waarbij een systeem tegelijkertijd zijn positie zal bepalen en een onbekende omgeving zal opnemen en verwerken. Zo kan het systeem zijn eigen positie blijven bepalen, ook als het in een nieuw stuk van de reeds gekende omgeving zal terechtkomen.



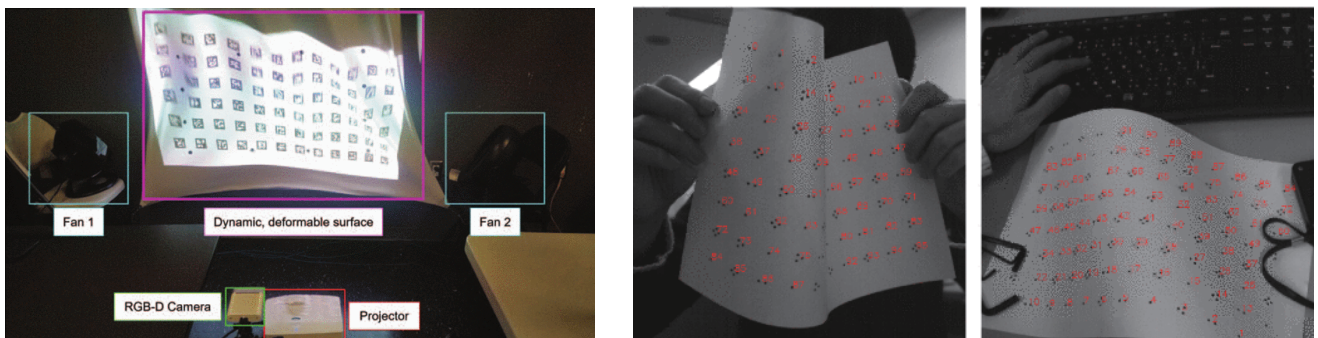
Figuur 2.1: Verschillende soorten fiduciële markers [2].

Het doel van objecttracking is het bepalen van de pose van het object ten opzichte van de camera. De pose van een object bevat zowel de positie, een 3D-coördinaat, als oriëntatie, de rotatie rond de drie assen, van het object in een driedimensionale ruimte. Samen resulteert dit in een *six degrees of freedom* (6-DOF) representatie.

2.1 Marker-gebaseerde tracking

Een vaak gebruikte methode is het bevestigen van fiduciële markers op het object en deze met een camera te volgen. Vierkante binaire markers zoals die uit de ArUco-bibliotheek leveren robuuste 6-DOF-pose-inschattingen op, doordat elke marker vier bekende hoekpunten bevat. De OpenCV ArUco-module wordt daarvoor vaak gebruikt [10].

In dit onderzoek wordt enkel tracking van niet-vervormbare objecten uitgevoerd, hoewel er in de literatuur ook implementaties bestaan die vervormbare oppervlakken of objecten kunnen tracken. Zo worden ArUco-markers geprojecteerd op een vervormbaar oppervlak, en wordt een RGB-D-camera gebruikt om markereigenschappen tussen frames te detecteren en te correleren, wat automatische kalibratie en real time pose-tracking mogelijk maakt [11]. Figuur 2.2a toont de opstelling voor het tracken van een vervormbaar oppervlak met geprojecteerde markers. Andere onderzoekers gebruiken gelijkaardige marker-gebaseerde systemen. Zo wordt in [12] een *dot matrix*-patroon ontworpen en geprint op flexibele vellen met infrarode inkt, wat tracking van niet-rigide objecten aan 1000 Hz mogelijk maakt. Deze *deformable dot cluster*-markers zijn onzichtbaar voor het menselijk oog en maken zeer snelle tracking die robuust is tegen oclusies mogelijk, zelfs bij sterke vervormingen. Figuur 2.2b toont de opstelling voor het tracken van een vervormbaar oppervlak met een reflecterende *dot matrix*.



(a) Tracking van vervormbaar oppervlak met behulp van geprojecteerde markers [11].

(b) Tracking van vervormbaar oppervlak met behulp van reflecterende *dot matrix* [12].

Figuur 2.2: Twee methodes om vervormbare oppervlakken te tracken.

Over het algemeen leveren marker-gebaseerde methodes nauwkeurige pose-inschattingen op, zolang de markers zichtbaar blijven voor de camera. Ze vereisen echter wel het plaatsen of projecteren van kunstmatige patronen op het doeloppervlak, wat de visuele kwaliteit van de projectie kan verstoren.

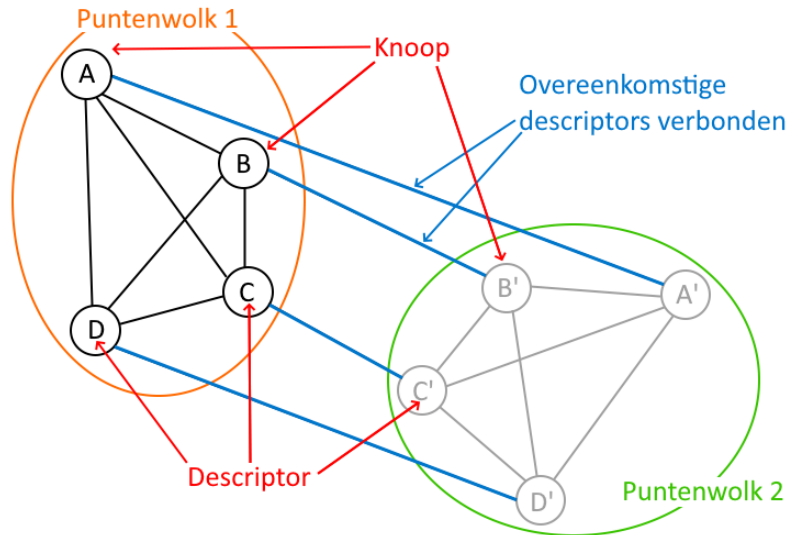
2.2 Diepte-gebaseerde tracking

Een andere strategie maakt gebruik van een RGB-D-sensor en een vooraf gekend 3D-model van het object, waarbij de diepte (of puntenwolk) wordt uitgelijnd met het gekende model. In de praktijk wordt hiervoor vaak het lokale registratie-algoritme *Iterative Closest Point* (ICP) gebruikt om de pose van het object te berekenen op basis van dieptegegevens [13]. De nauwkeurigheid hangt af van de precisie van de sensor, de kwaliteit van het 3D-model en een redelijke initiële poseinschatting.

Naast het ICP-algoritme bestaan er ook globale registratie-algoritmes zoals KISS-Matcher [13], die een initiële uitlijning kan doen zonder de vereiste beginschatting. Deze techniek maakt gebruik van robuuste descriptoren en grafentheoretische filtering om de pose te schatten.

In tegenstelling tot ICP is KISS-Matcher minder gevoelig voor een slechte initiële pose-inschatting. Dit komt omdat het algoritme de puntenwolkregistratie globaal aanpakt: in plaats van iteratief de punten uit te lijnen op basis van de initiële pose-inschatting (zoals bij lokale registratie-algoritmes), worden punten uitgelijnd aan de hand van een *descriptor*. Een *descriptor* is een

unieke representatie van de lokale geometrie rond een punt, waarmee punten uit beide puntenwolken gematcht kunnen worden. Deze overeenkomstige punten worden dan voorgesteld als verbonden knopen in de graaf. Door vervolgens grafentheorie toe te passen, worden de meest consistente *matches* geselecteerd. Op basis van deze matches kan de uiteindelijke uitlijning bepaald worden [13]. In figuur 2.3 wordt een visuele voorstelling gegeven van het globale registratie-algoritme.



Figuur 2.3: Visuele representatie van een globaal registratie-algoritme.

Met moderne dieptecamera's (zoals Kinect of RealSense) kan men kleur- en dieptedata combineren voor robuuste tracking. In [14] wordt een real time CPU-gebaseerd trackingsysteem voorgesteld dat modelcontouren en dieptemetingen uitlijnt via een gecombineerde contour- en ICP-kostfunctie. Deze functie beschrijft de fout tussen de te matchen puntenwolken en wordt geoptimaliseerd om zo de best mogelijke uitlijning te realiseren. Het toont aan dat het combineren van beeldcontouren met de dieptekaart leidt tot nauwkeurige 6-DOF-tracking, zelfs voor modellen zonder textuur. Over het algemeen maakt RGB-D-modeltracking markerloze uitlijning van het object mogelijk, bijvoorbeeld door punt-tot-punt- of punt-tot-vlakafstanden te minimaliseren.

2.3 SLAM-gebaseerde tracking

Een derde methode is om het projector-camera-systeem te behandelen als een bewegende sensor, en visuele *Simultaneous Localization And Mapping* (SLAM) toe te passen om het systeem te lokaliseren in de omgeving. Standaard SLAM-systemen (feature-gebaseerd zoals ORB-SLAM of directe methodes zoals LSD-SLAM) kunnen een schaars model van de scène opbouwen terwijl ze in real time de 6-DOF-pose van de camera berekenen [15], [16]. Moderne SLAM-algoritmes zoals Open-VSLAM zijn gemaakt om meerdere soorten camera-opstellingen te ondersteunen, waaronder monoculair (één RGB-camera), stereo (meerdere RGB-camera's) en RGB-D (een camera met zowel RGB- als een dieptebeeld) [3]. Ook laat het toe om eerder opgenomen SLAM-mappen op te slaan en later opnieuw in te laden. Er zijn verdere uitbreidingen gemaakt op het Open-VSLAM-algoritme om een robuuste markerintegratie toe te voegen, zoals in het UcoSLAM-algoritme [17]. Figuur 2.4 toont een voorbeeld van een SLAM-map.



Figuur 2.4: Een SLAM-map van een gebouw, gemaakt met OpenVSLAM [3].

Als het gezichtsveld van de camera echter overlapt met het projectiegebied, dan verschijnt de dynamische projectie-inhoud ook in de camerabeelden. Dit kan problematisch zijn, omdat SLAM doorgaans uitgaat van een grotendeels statische omgeving, terwijl geprojecteerde beelden natuurlijke kenmerken kunnen veranderen of verbergen. In [18] wordt opgemerkt dat bij het projecteren van content voor de camera “veel goede features die voor tracking worden gebruikt zullen verslechteren” en effectief verloren gaan. Kortom, SLAM met overlappend gezichtsveld moet omgaan met snel veranderende belichting en texturen (van het geprojecteerde licht), wat het tracken en mappen bemoeilijkt.

2.3.1 Vergelijkende analyse van trackingmethodes

Op basis van de besproken literatuur in deze bronnenstudie is er een overzichtstabel opgesteld waarin de belangrijkste sterktes en zwaktes van elke methode worden opgelijst. De vergelijking focust op aspecten zoals nauwkeurigheid, robuustheid, vereisten voor implementatie en gevoeligheid voor omgevingsfactoren. De tabel hieronder dient als samenvatting en als ondersteuning bij het maken van een onderbouwde keuze in het verdere verloop van dit onderzoek.

Tabel 2.1: Vergelijking van trackingmethodes voor dynamische projection mapping.

Methode	Sterktes	Zwaktes	Bronnen
Markergebaseerd	- Zeer nauwkeurige 6-DOF-pose bij goede zichtbaarheid - Lage complexiteit en robuust met ArUco-markers	- Afhankelijk van zichtbaarheid - Visuele verstoring bij projectie (zichtbare markers)	[10], [11], [12]
Dieptegebaseerd (ICP)	- Markerloze tracking - Goede lokale precisie mits goede initialisatie	- Gevoelig voor ruis en slechte initiële schatting - Computatieel zwaar	[13], [14]
Dieptegebaseerd (KISS)	- Geen initiële pose nodig - Robuust door descriptor matching en grafentheorie	- Hogere rekenkost - Afhankelijk van kwaliteit puntenwolk en descriptors	[13]
SLAM-gebaseerd	- Real time en robuust in onbekende omgeving - Ondersteunt RGB-D en marker-integratie	- Moeilijk in dynamische projectie-omgevingen - Drift zonder herlocalisatie - Complexe implementatie	[15], [3], [17], [18]

Hoofdstuk 3

Methodologie

Deze studie heeft als doel om de meest geschikte object-trackingmethode voor real time projector-camerasystemen te bepalen, door drie benaderingen te vergelijken: marker-gebaseerde tracking met ArUco-markers, dieptegebaseerde registratie en SLAM. Enkel marker-gebaseerde en SLAM-tracking werden geëvalueerd, de dieptegebaseerde tracking is onvoldoende robuust om er een evaluatie op uit te voeren. Alle systemen werden geïmplementeerd in C++ en draaien op identiek dezelfde hardware. De beweging van het camera-projectorsysteem wordt handmatig uitgevoerd door een operator, waarbij het werkelijke traject wordt geregistreerd met een Qualisys motion-capturesysteem.

3.1 Gebruikte hardware en software

Voor de uitvoering van dit project werd gebruikgemaakt van een laptop met een **AMD Ryzen 9 5900HS** processor, **16GB RAM** en een **NVIDIA RTX 3050 Ti Mobile** GPU. Als besturingssysteem werd **Linux (Ubuntu 22.04)** gebruikt.

De visuele input werd geleverd door een **Intel RealSense** met dieptesensor, terwijl de projectie gerealiseerd werd met een **Dangbei Atom** projector.

Voor de implementatie werd zowel **C++** als **Python** gebruikt. De **C++**-componenten, waaronder de trackingmethodes en OpenGL-gebaseerde visualisatie, zijn gecompileerd met behulp van **CMake**. Voor de ontwikkeling werd gebruikgemaakt van **Qt Creator** als IDE. De trackingsoftware maakt gebruik van verschillende externe C++ libraries, waaronder:

- **librealsense2** (voor toegang tot de camera)
- **OpenCV** versie 4 (tracking)
- **OpenCV** versie 5 (voor kalibratiecode gebaseerd op onderzoek van Vanherck et al. [18].)
- **Eigen**
- **PCL** (Point Cloud Library)
- **kiss-matcher**
- **stella-vslam**
- **GLAD**, **GLFW**, **GLM**, **stb_image** (voor OpenGL-rendering)

De **Python**-code werd voornamelijk gebruikt voor eigen kalibratiecode, data-acquisitie en verwerking. Hiervoor werd **Python 3.12** gebruikt in combinatie met **Visual Studio Code** als ontwikkelomgeving. De gebruikte Python-libraries omvatten onder andere:

- `opencv-python` (versie 4), `numpy`
- `matplotlib`, `json`, `csv` (voor het opslaan, inladen en visualiseren van de data.)
- `socket`, `struct`, `screeninfo`
- `qtm_rt`, `xml.etree.ElementTree`, `threading`, `signal`, `sys`, `asyncio` (voor communicatie met het Qualisys-systeem.)

Een deel van de kalibratiecode werd overgenomen van een eerder systeem, beschreven in [18]. Het draait in een Windows-omgeving en maakt gebruik van **OpenCV 5**.

3.2 Kalibratie

Kalibratie is een essentiële stap in het opstellen van een accuraat projector-camerasysteem. Zowel de camera als projector en het gehele systeem zullen gekalibreerd worden. Zo zullen er intrinsieke parameters van zowel de camera als de projector geschat worden, samen met de extrinsieke parameters van het gehele systeem. Een uitgebreide beschrijving van de gebruikte kalibratiemethoden volgt in hoofdstuk 4.

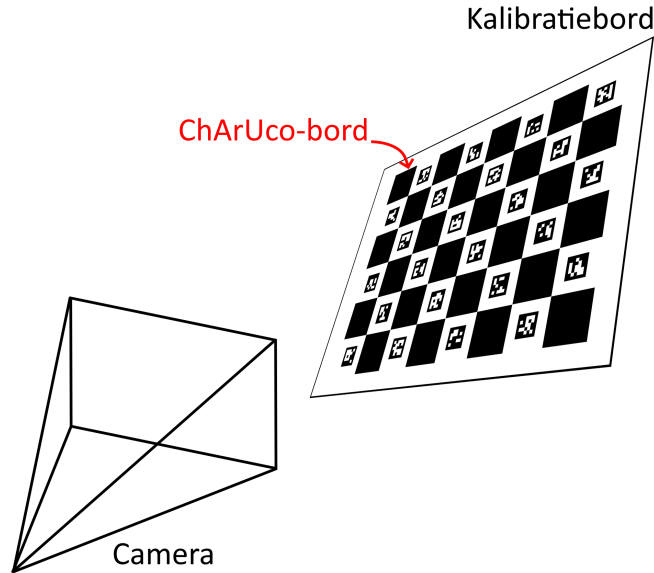
De intrinsieke parameters beschrijven de fysieke eigenschappen van een lens (die van de camera of projector) en bestaan uit de intrinsieke matrix en distortieparameters. De intrinsieke matrix bevat informatie zoals de brandpuntsafstand en het optische centrum. Distortieparameters beschrijven verschillende soorten lensvervormingen, zoals radiale en tangentiële vervormingen.

Extrinsieke parameters beschrijven de relatieve pose van de camera ten opzichte van de projector binnen hetzelfde coördinatensysteem. Zoals eerder besproken bestaat de pose uit een rotatiematrix en een translatievector die samen de gehele transformatiematrix vormen. Deze matrix kan gebruikt worden om punten van het ene referentiekader om te rekenen naar het andere.

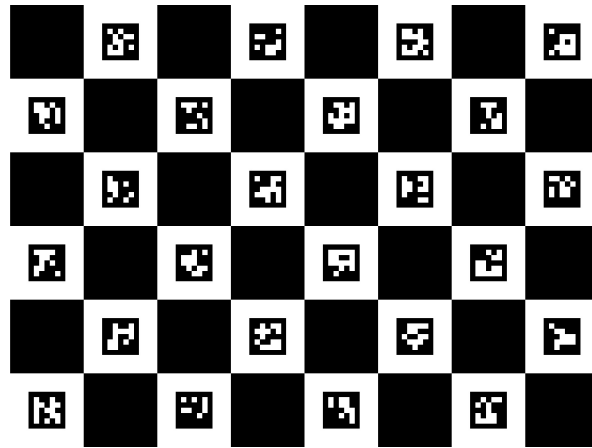
3.2.1 Camerakalibratie

De camerakalibratie is gebaseerd op de methode van Zhang [19]. Er wordt een ChArUco-bordpatroon in beeld gebracht van de camera. Het patroon wordt gedetecteerd en de 2D-pixelcoördinaten van de detectiepunten worden opgeslagen. Het verzamelen van de detectiepunten wordt herhaald, waarbij het bord telkens in een verschillende oriëntatie en positie staat. Hierbij is het ook van belang dat het bord zichtbaar is aan de randen en hoeken van het camerabeeld om de kwaliteit van de distortieparameters te verhogen. Distortieparameters treden voornamelijk op aan de randen van het camerabeeld. Figuur 3.1 toont een schematische voorstelling van een camerakalibratie.

Een ChArUco-bord is een combinatie van een *chessboard*-patroon en ArUco-markers. Het patroon bestaat uit de afwisselende zwarte en witte vierkanten van een *chessboard*, waarbij er in elk wit vierkant een ArUco-marker met unieke identifier geplaatst is. Figuur 3.2 geeft een voorbeeld van een ChArUco-bord weer.



Figuur 3.1: Een schematische voorstelling van een camerakalibratie.



Figuur 3.2: Een ChArUco-bord gebruikt voor kalibratie van verschillende soorten camera's.

3.2.2 Projector- en projector-camerakalibratie

De projector en het projector-camerasysteem worden samen gekalibreerd op basis van dezelfde dataset die verzameld wordt tijdens het kalibratieproces. Deze methode is gebaseerd op het werk van Falcao et al. [20].

De opstelling bestaat uit een kalibratiebord waarop een ChArUco-patroon is aangebracht, met rondom voldoende lege ruimte om projectie van een patroon door de projector op te vangen. Figuur 3.3 toont een schematische voorstelling van de projector-cameraopstelling tijdens de kalibratie. Tijdens het volledige kalibratieproces projecteert de projector een detecteerbaar patroon op dit bord. In deze toepassing is gekozen voor een asymmetrisch *dot-matrix* patroon.

Het proces verloopt als volgt:

Eerst detecteert de camera het ChArUco-bord. Met behulp van de 2D-pixelcoördinaten van de gedetecteerde punten en hun gekende 3D-objectcoördinaten wordt een homografie geschat tussen het beeldvlak van de camera en het vlak van het kalibratiebord. Deze homografiematrix beschrijft de transformatie van punten in het camerabeeld naar corresponderende punten op het

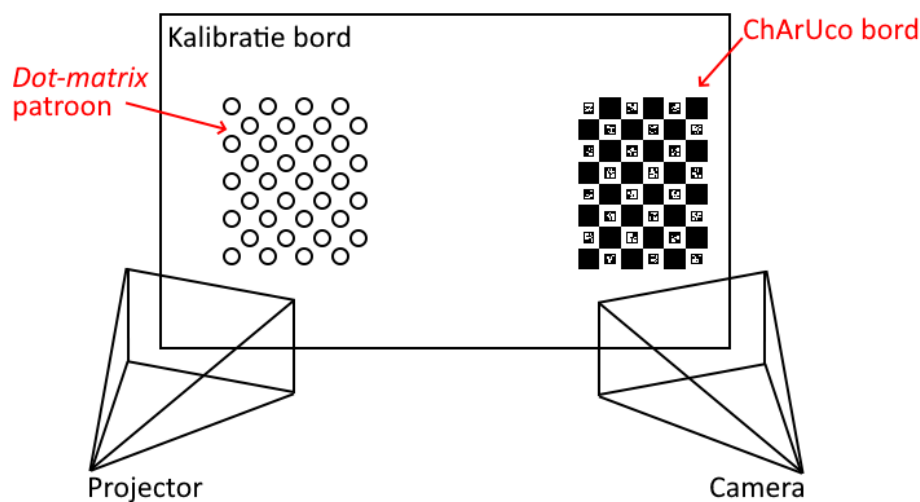
kalibratiebord.

Vervolgens detecteert de camera het geprojecteerde *dot-matrix* patroon. De 2D-coördinaten van deze punten worden omgezet naar het coördinatenstelsel van het kalibratiebord door vermenigvuldiging met de eerder berekende homografiematrix. Hierdoor zijn de posities van de projectiepunten nu eveneens gekend in het referentiekader van het bord.

Momenteel zijn er drie kalibratiedatasets beschikbaar:

- De projectiepunten in het camerabeeld,
- De corresponderende punten in het beeldvlak van de projector,
- En hun 3D-posities op het kalibratiebordvlak (berekend via homografie).

Deze informatie wordt vervolgens gebruikt om de intrinsieke parameters van de projector te schatten, analoog aan de kalibratie van de camera via de methode van Zhang [19]. Daarnaast kunnen hiermee ook de extrinsieke parameters berekend worden.



Figuur 3.3: Schematische voorstelling van de projector-cameraopstelling tijdens kalibratie. Het ChArUco-bord met projectiezone wordt door de camera waargenomen, terwijl de projector het *dot-matrix* patroon projecteert.

3.3 Trackingmethodes

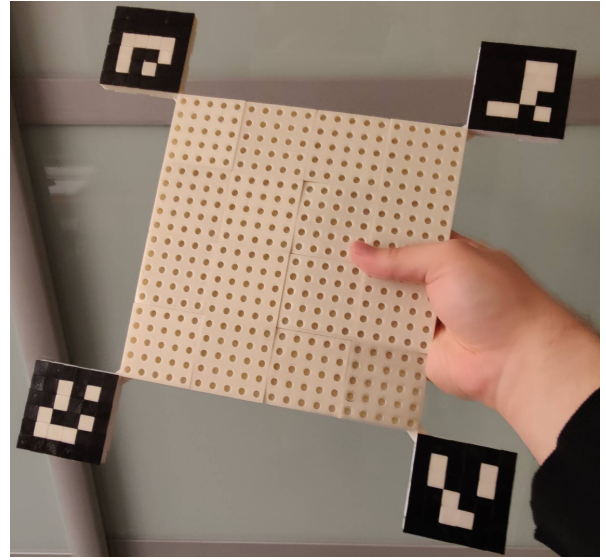
Het bepalen van de pose van een object ten opzichte van de camera kan op verschillende manieren gedaan worden. Er zijn drie methodes geïmplementeerd, waarvan er twee verder geëvalueerd worden.

3.3.1 Tracking met ArUco-markers

Om de pose te bepalen, wordt gebruikgemaakt van een ArUco-bord, bestaande uit vier individuele markers in een vaste configuratie, elke marker met een unieke identicator. Figuur 3.4 toont het gebruikte ArUco-bord, beide in digitale representatie (a) en fysieke opstelling (b).



(a) Afbeelding van ArUco-bord.



(b) ArUco-bord als fysiek object.

Figuur 3.4: Het ArUco-bord dat getrackt wordt in zowel de digitale voorstelling als het fysieke object.

De markers worden gedetecteerd in het camerabeeld, eerst een normale detectie, daarna een verfijnde detectie waar de huidige gevonden markers en de structuur van het ArUco-bord mee worden genomen als parameters, waardoor markers die eerder niet gevonden werden nu mogelijk wel gedetecteerd worden. De gedetecteerde 2D-punten samen met de 3D-objectpunten van het aruco-bord en de intrinsieke parameters van de camera worden gebruikt om het *Perspective-n-Point* (PnP)-probleem op te lossen. Indien de berekening mogelijk is, geeft het de 6-DOF-pose van de camera ten opzichte van het ArUco-bord en dus ook het object.

Het PnP-probleem is het probleem dat beschrijft hoe de 6-DOF-pose van de camera ten opzichte van een object berekend kan worden op basis van de 3D-punten van het object en de bijbehorende 2D-punten in het camerabeeld. De intrinsieke parameters worden gebruikt om lensvervalsingen te elimineren in de berekeningen. Hoofdstuk 4 bevat een uitgebreide uitleg over het oplossingsalgoritme van dit probleem.

3.3.2 Dieptegebaseerde tracking

Een tweede methode voor het bepalen van de objectpose maakt gebruik van het dieptebeeld dat wordt opgenomen door een dieptecamera. Deze aanpak vereist dat vooraf een puntenwolk van het object dat getrackt gaat worden ingeladen wordt.

Bij het uitvoeren van de tracking wordt een nieuw dieptebeeld opgenomen. Op basis van dit dieptebeeld wordt er een puntenwolk berekend, vanuit het perspectief van de camera. Vervolgens wordt er gebruikgemaakt van de KISS-matcher om de puntenwolk van het object te aligneren met de huidige puntenwolk van de camera [13]. De uitvoer van de KISS-matcher is de pose van het object ten opzichte van de camera. Deze matrix beschrijft dus hoe de puntenwolk van het object getransformeerd moet worden om optimaal te aligneren met de puntenwolk van de camera.

Deze methode vereist geen visuele markers en is daardoor bijzonder geschikt in situaties waar het plaatsen van markerstructuren op het object niet wenselijk of mogelijk is. Wel vereist ze

voldoende diepteinformatie en een kwalitatief goede puntwolkregistratie.

3.3.3 SLAM-gebaseerde tracking

In deze methode zal het projector-camerasysteem enkel kunnen bewegen, terwijl het object onderdeel van de omgeving zal worden en dus stil moet blijven staan. De tracking van het systeem gebeurt via Stella-VSLAM. Dit SLAM-algoritme is direct afgeleid van het Open-VSLAM-algoritme [3]. Er wordt op voorhand in een configuratiebestand gezet wat voor camera er gebruikt zal worden, samen met hoeveel en welke soort markers Stella-VSLAM zal mogen verwachten in de trackingomgeving. In dit onderzoek wordt er gebruikgemaakt van een monoculaire camera, dit is een camera die enkel een RGB-beeld opneemt. In de configuratie wordt ook ingesteld dat er vier ArUco-markers in de omgeving gevonden kunnen worden, dezelfde markers die te zien zijn in figuur 3.4b.

Deze stappen zullen eerst uitgevoerd worden voor de tracking zelf:

1. **Initialisatie van SLAM en detectie van markers:** Stella-VSLAM wordt gestart en verwerkt continu een RGB-beeld van de camera. Ondertussen detecteert het SLAM-algoritme markers die gezien worden die voldoen aan de configuratie en slaat het deze op in de achterliggende SLAM-map.
2. **Berekening van de transformatie tussen het SLAM-coördinaatsstelsel en het objectcoördinaatsstelsel:** Om correcte tracking te verkrijgen, moet de transformatie van het SLAM-coördinatensysteem naar het object-coördinatensysteem bepaald worden. Hiervoor moeten twee parameters worden berekend, de transformatiematrix die punten van de SLAM-ruimte naar de objectruimte kan brengen en de schaalfactor die tussen de twee ruimtes zit. Het is nodig om de schaal te berekenen, omdat er met een monoculaire camera geen diepte bepaald kan worden. De enige beschikbare informatie is een RGB-beeld, hierdoor zal het SLAM-algoritme zijn eigen schaal kiezen bij het initialiseren.
3. **Transformatiematrix:** De markers zijn bekende punten in het object-coördinatensysteem. Door de markers in de camerabeelden te detecteren, kan de pose van de camera ten opzichte van het object worden bepaald via een *Perspective-n-Point*-oplossing, analoog aan de ArUco-markertracking. Dit levert de transformatiematrix van object naar camera op.
4. **Schaalbepaling:** Tegelijkertijd heeft Stella-VSLAM de markers in zijn eigen SLAM-map geregistreerd. De 3D-coördinaten van deze markers zijn gedefinieerd in het SLAM-coördinatensysteem. Met de bekende 3D-markerposities in het object-coördinatensysteem kan een 3D-affine transformatie en de schaalfactor tussen de SLAM-markerposities en de object-markerposities worden geschat. Enkel de schaalfactor is hier nodig uit deze berekening. De affine transformatie geeft een inconsistent resultaat om de rechtstreekse transformatie te bepalen. Daarom wordt de volgende stap uitgevoerd.
5. **Berekening van de uiteindelijke transformatie:** De uiteindelijke transformatie van SLAM-coördinaten naar objectcoördinaten wordt als volgt verkregen. De camerapose is verkrijgbaar via SLAM en staat dan ook in het SLAM-coördinaatsysteem. De camerapose in SLAM wordt eerst gecombineerd met de schaalfactor in de vorm van een schaalmatrix.

De schaalmatrix is 4×4 identiteitsmatrix, waarbij de drie eerste diagonalen vervangen zijn door schaalfactor s , het laatste element in de diagonaal blijft één. Onderstaande matrix

3.1 toont de opbouw van een schaalmatrix.

$$\mathbf{S} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

Daarna wordt de eerder berekende transformatiematrix tussen het object en de camera gecombineerd met de geschaalde pose van de camera en dit bekomt de transformatiematrix die gezocht wordt. De transformatie tussen het SLAM-coördinaatsysteem en het objectcoördinaatsysteem.

Als de transformatie tussen de twee ruimtes gekend is, kan de pose van het object ten opzichte van de camera voor elk frame bepaald worden als volgt. Opnieuw wordt de camerapose genomen vanuit SLAM en gecombineerd met de schaalmatrix. Vervolgens wordt de geschaalde camerapose gecombineerd met de transformatiematrix die eerder berekend werd om zo direct de camerapose ten opzichte van het object te bekomen.

3.4 Visualisatie

Voor de visualisatie van het getrackte object is er een eenvoudige OpenGL-renderer gemaakt. De renderer kan een obj-bestand inladen, samen met een bijhorende textuurafbeelding. Beide de projectiematrix van de *cameraview* en de pose van het ingeladen model kunnen in real time aangepast worden.

De renderer is intentioneel minimalistisch opgezet en heeft verder geen andere functionaliteiten. Het doel van de renderer is om de projectoroutput te renderen met zo weinig mogelijk invloed op andere onderdelen van het systeem. Denk bijvoorbeeld aan het beperken van het gebruik van *system resources*, zodat de prestaties van andere onderdelen zoals de tracking niet negatief beïnvloed worden.

3.5 Communicatie

Alle poses die door een trackingalgoritme berekend worden, worden in real time doorgestuurd naar de OpenGL-renderer via een UDP-connectie. Elk UDP-pakket bevat een *flag* en een lijst van *doubles* die de intrinsieke parameters van de projector of de pose van het object voorstellen.

De *flag* in het datapakket bepaalt het type matrix dat verstuurd wordt. Als de *flag* een projectiematrix aanduidt met een char 'i', wordt deze matrix gebruikt om de OpenGL-view te configureren. Als de flag een posematrix aanduidt met een char 'p', wordt deze toegepast als transformatie op het ingeladen model in de renderer.

Aan de kant van de OpenGL-renderer wordt de UDP-data ontvangen, geïnterpreteerd op basis van de meegegeven flag, en vervolgens gebruikt om de projectiematrix of posematrix van het object dynamisch aan te passen. Deze eenvoudige communicatie zorgt voor een efficiënte koppeling tussen tracking en visualisatie zonder onnodige overhead.

3.6 Evaluatie

Voor de evaluatie van de trackingmethodes wordt er gebruikgemaakt van een Qualisys motion-capturesysteem. Dit systeem maakt het mogelijk om nauwkeurig de pose van een object te bepalen met een snelheid van 300 Hz. Dit doet het Qualisys-systeem met behulp van infrarood-camera's en reflectieve markers. Er werd een *rigid body* gedefinieerd in de Qualisys QTM-software door een set retro-reflectieve markers te groeperen. De retro-reflectieve werden bevestigd op het projectie-camerasysteem. Deze opstelling is te zien in figuur 3.5.



Figuur 3.5: Het projector-camerasysteem met retro-reflectieve markers voor tracking met een motion-capturesysteem.

Er werden twee trackingmethodes geëvalueerd, namelijk ArUco-markertracking en SLAM-tracking. Voor beide trackingmethodes zijn er metingen opgenomen waar de projector aan en uit staat. De data van de ArUco-markermethode is de camerapose ten opzichte van het object en bij SLAM-tracking is het de camerapose verkregen van SLAM. Deze poses worden verstuurd via een UDP naar een ontvanger-script en worden elke keer opgeslagen samen met de meest recente Qualisys-pose en het bijbehorende framenummer. Dit zorgt ervoor dat alle data synchroon opgeslagen wordt. De data wordt in een JSON-bestand opgeslagen, omdat dit makkelijk is voor het opslaan van matrices.

Deze data kan later verwerkt worden om de fout op de rotatie en translatie te bepalen. Ook kan de hoeveelheid *frames* per seconde (fps) van de trackingmethodes bepaald worden.

Hoofdstuk 4

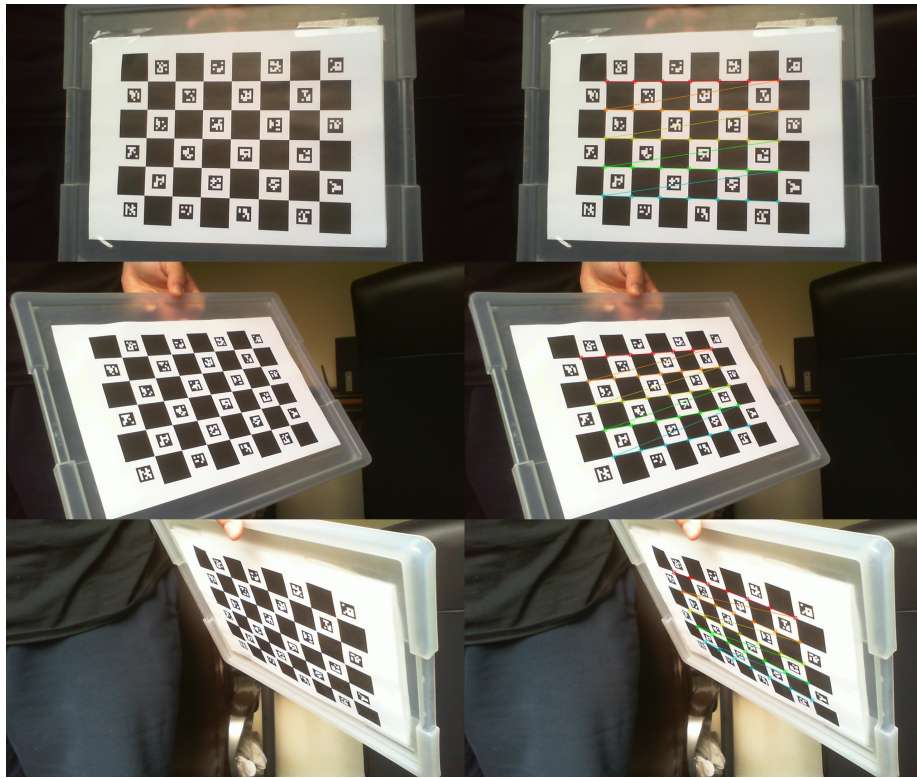
Kalibratie

4.1 Camera kalibratie

Een correcte camerakalibratie is cruciaal voor het omzetten van 2D-pixelcoördinaten naar 3D-coördinaten. In deze sectie wordt de methode van Zhang [19] toegelicht, gevolgd door een gedetailleerde bespreking van de bekomen kalibratieparameters.

4.1.1 Kalibratieproces

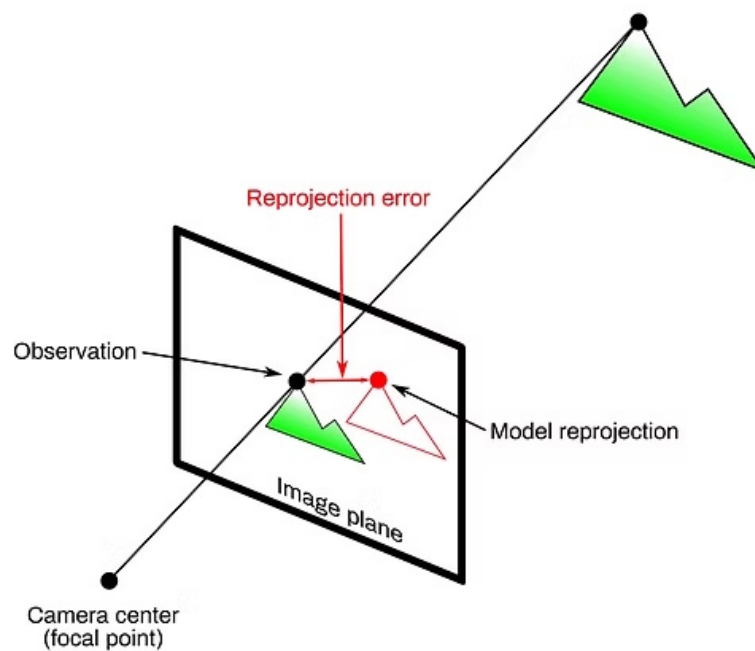
Zhang heeft aangetoond dat een vlak kalibratiepatroon, bijvoorbeeld een schaakbord of ChArUco-bord, in diverse willekeurige oriëntaties en afstanden voor de camera volstaat voor een correcte kalibratie [19]. Voor elk beeld wordt een homografiematrix $\mathbf{H}$ bepaald die het vlak van het patroon naar het beeldvlak projecteert. Door een reeks van dergelijke homografieën te verzamelen, kan men zowel de intrinsieke matrix $\mathbf{K}$ als de distortiecoëfficiënten $\mathbf{d}$ schatten via een niet-lineaire optimalisatie. Hierbij wordt de *reprojection error* geminimaliseerd. Figuur 4.1 toont een voorbeeldopstelling met het kalibratiebord in verschillende oriëntaties en afstanden.



Figuur 4.1: Voorbeeldopstelling voor camerakalibratie met een ChArUco-bord in verschillende poses.

4.1.2 Reprojection error

De *reprojection error* is de afstand tussen de gemeten 2D-pixelcoördinaat van een kalibratiepunt en de 2D-pixelcoördinaat waarop datzelfde punt wordt geprojecteerd met de geschatte camera-parameters. Dit verschil wordt in figuur 4.2 schematisch voorgesteld [4].



Figuur 4.2: Een schematische voorstelling van de *reprojection error* [4].

Een lage reprojectie-error duidt op een nauwkeurige kalibratie, waarbij de geschatte parameters

goed overeenkomen met de werkelijkheid. Tijdens het kalibratieproces worden de intrinsieke parameters geoptimaliseerd om deze fout over alle kalibratiedata zo klein mogelijk te maken.

4.1.3 Intrinsieke matrix

De intrinsieke matrix $\mathbf{K}$ bevat alle interne kenmerken van de camera:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

f_x, f_y Brandpuntsafstanden in respectievelijk x - en y -richting; uitgedrukt in pixels. Deze term vertegenwoordigt de effectieve brandpuntsafstand in zowel de x - als de y -richting, uitgedrukt in pixels. Hierbij wordt rekening gehouden met zowel de fysieke brandpuntsafstand van de lens als de fysieke pixelgrootte van de sensor. Deze kunnen verschillen wanneer de pixels niet vierkant zijn.

s Shear (skew) tussen de x - en y -as. Deze parameter beschrijft de hoekafwijking tussen de x - en y -as van de sensorpixels. Bij moderne camera's is de skew doorgaans nul, wat betekent dat de sensorpixels rechthoekig zijn en geen onderlinge scheefstand vertonen.

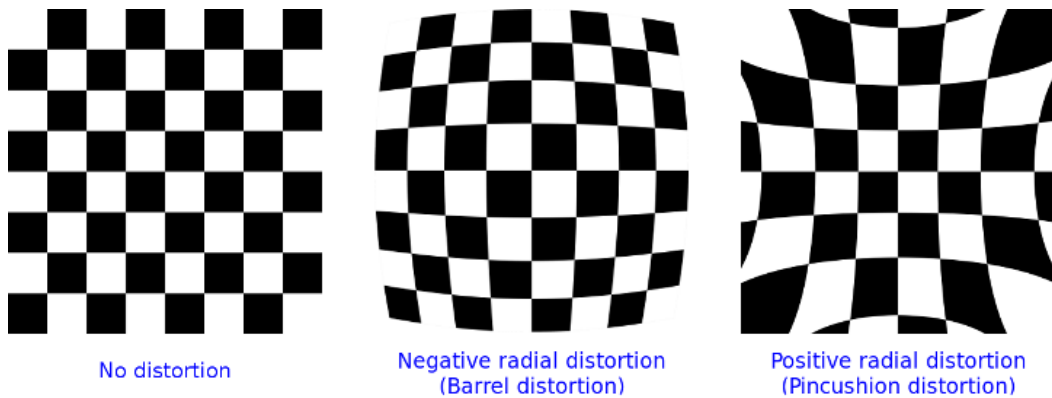
c_x, c_y Dit zijn de coördinaten van het hoofd- of optisch middelpunt van de projectie op de sensor. Bij een camera ligt dit punt in het midden van het beeld, maar in de praktijk kan het iets verschoven zijn. Bij een projector ligt dit middelpunt typisch lager.

4.1.4 Distortiecoëfficiënten

Tijdens de beeldvorming treden lensvervalsingen op. Zhangs methode gebruikt typisch vijf parameters, gegroepeerd in een vector

$$\mathbf{d} = [k_1, k_2, p_1, p_2, k_3] \quad (4.2)$$

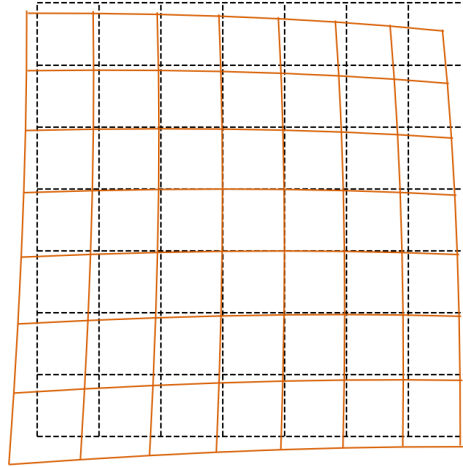
k_1, k_2, k_3 Deze coëfficiënten modelleren de radiale vervorming van respectievelijk eerste, tweede en derde orde. Het veroorzaakt ton- of kussenvormige afwijkingen die toenemen naarmate de afstand tot het optische centrum toeneemt. Onderstaande figuur 4.3 toont een voorbeeld van ton- en kussenvormige distortie.



Figuur 4.3: Visueel effect van radiale vervorming [5].

$\mathbf{p}_1, \mathbf{p}_2$ Tangentiële (of decentratie) vervorming ontstaat wanneer de lens niet perfect gecentreerd is ten opzichte van de sensor. Figuur 4.4 vergelijkt een niet-vervormd raster met een dat tangentiële vervorming bevat.

Tangential (Decentering) Distortion



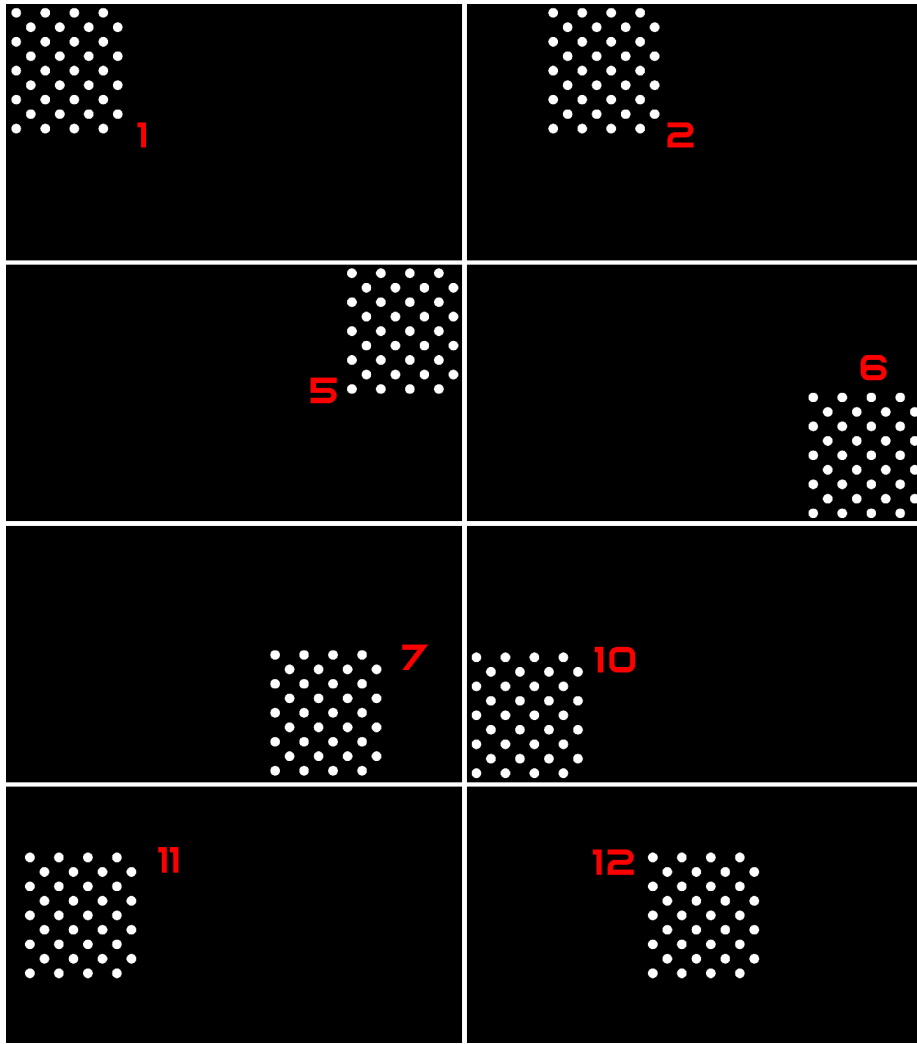
Figuur 4.4: Visueel effect van tangentiële vervorming [6].

4.2 Projector-camera kalibratie

Na de camerakalibratie zal het ProCam-systeem en de projector zelf gekalibreerd worden. De data die vereist is voor de kalibratie van de projector, is ook nodig voor het kalibreren van het gehele systeem. Daarom zullen zowel de intrinsieke van de projector als de extrinsieke van het gehele systeem samen besproken worden. De gebruikte methode is gebaseerd op die van Falcao et al. [20].

4.2.1 Verschillen met camerakalibratie en patroonkeuze

De kalibratie van de projector verloopt op analoge wijze als die van de camera, het voornaamste verschil is de manier waarop de kalibratiedata verzameld wordt. Bij camerakalibratie werd er een patroon gebruikt waarvan de 3D-objectcoördinaten gekend zijn, de corresponderende 2D-pixelcoördinaten moesten gevonden worden. Voor het kalibreren van een projector zal het omgekeerde nodig zijn, het patroon dat de projector projecteert is gekend en dus ook de 2D-pixelcoördinaten. Omdat de 2D-pixelcoördinaten al beschikbaar zijn, moeten enkel de 3D-objectcoördinaten bepaald worden. In dit onderzoek wordt er gebruikgemaakt van een *dot-matrix* patroon. Hier is voor gekozen, omdat het niet kan interfereren met de detectie van andere patronen, zoals een ChArUco-bord. Om het volledige projectiebeeld in de kalibratie te betrekken, worden de *dot-matrix*-patronen op uiteenlopende posities binnen het beeld geprojecteerd. Hierdoor kunnen de distortieparameters van de projector betrouwbaar geschat worden. Figuur 4.5 geeft een aantal patronen weer.



Figuur 4.5: Verschillende gebruikte *dot-matrix* patronen voor projector-camera-kalibratie.

4.2.2 Detecteren van patronen

Het bepalen van de 3D-objectcoördinaten van het geprojecteerde patroon vereist het gebruik van de camera en werkt als volgt. De reeds gekalibreerde camera zal zowel een ChArUco-bord als het *dot-matrix* patroon detecteren. Aan de hand van het ChArUco-bord en zijn gekende 3D-objectcoördinaten en de gedetecteerde 2D-pixelcoördinaten kan de homografie berekend worden tussen het camerabeeldvlak en het kalibratiebord. Deze homografie zal ook geldig zijn voor het *dot-matrix* patroon, aangezien deze ook geprojecteerd wordt op hetzelfde kalibratiebord. Door de homografie toe te passen op de 2D-pixelcoördinaten van het *dot-matrix* patroon, dat de camera heeft gedetecteerd, worden de 3D-objectcoördinaten gevonden. Figuur 4.6 toont het kalibratiebord in verschillende poses tijdens het kalibreren.

4.2.3 Intrinsieke kalibratie

Met nu de 2D-pixelcoördinaten van het *dot-matrix* patroon gekend, in zowel de projector als de camera, en de bijbehorende 3D-objectcoördinaten kan zowel de intrinsieke kalibratie van de projector als de extrinsieke kalibratie van het gehele systeem uitgevoerd worden. Zoals eerder vermeld zal de intrinsieke projector-kalibratie op dezelfde manier uitgevoerd kunnen worden als een camerakalibratie. De parameters hebben identiek dezelfde structuur als die van de camera.



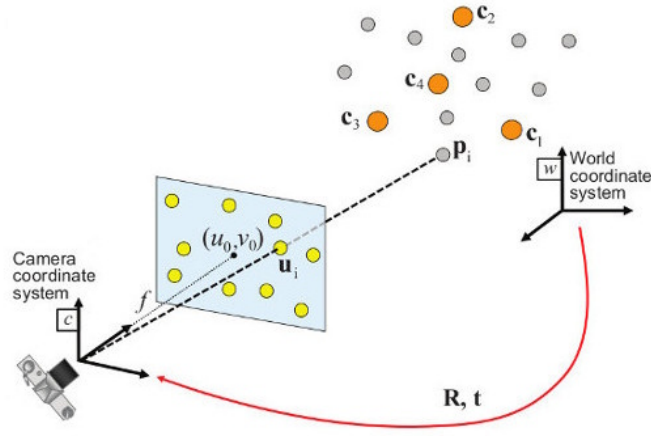
Figuur 4.6: Voorbeeldopstelling voor projector-camera-kalibratie met een kalibratiebord in verschillende poses.

4.2.4 Perspective-n-point-probleem

De extrinsieke kalibratie maakt gebruik van een oplossingsalgoritme voor het PnP-probleem. Hieronder volgt een uitgebreide uitleg die meer inzicht biedt in hoe dit algoritme werkt.

Het PnP-probleem is een minimalisatieprobleem met als einddoel het bepalen van de rotatie en translatie tussen een object en de camera. Het algoritme werkt als volgt, de rotatie en translatie starten met standaardwaarden. De 3D-punten zullen dan geprojecteerd worden op het beeldvlak van de camera op basis van de rotatie en translatie en de intrinsieke parameters van de camera. De 2D-punten uit deze berekening worden vergeleken met de gegeven detectiepunten. De vergelijking gebeurt op basis van de berekeningen van de *reprojection error* tussen alle overeenkomende punten.

Vervolgens kan op basis van de *reprojection error* en de gebruikte rotatie- en transformatieparameters de Jacobiaan berekend worden. De Jacobiaan geeft aan hoe sterk elke foutcomponent verandert bij een verandering in de rotatie of translatie. Daarnaast toont het ook in welke richting de rotatie en translatie gecorrigeerd moeten worden om de fout te verkleinen. De rotatie en translatie worden vervolgens dus aangepast op basis van de Jacobiaan. Dit proces wordt herhaald tot de *reprojection error* van de punten onder een bepaalde grens ligt of tot het maximumaantal iteraties is bereikt. De uitkomst is dan een rotatie- en translatievector die omgevormd kan worden tot een transformatiematrix of pose. Onderstaande figuur 4.7 geeft een overzicht van de oplossing voor een PnP-probleem [7].



Figuur 4.7: Een overzicht van de oplossing voor een PnP-probleem [7].

4.2.5 Extrinsieke kalibratie

De extrinsieke kalibratie zelf werkt als volgt. Voor elk frame waarin het kalibratiebord zichtbaar is, wordt eerst afzonderlijk in het camera- en projectorbeeld een PnP-probleem opgelost. Dit levert voor elk frame de pose op van het kalibratiebord ten opzichte van elk apparaat. Omdat het kalibratiebord in elk frame op een andere plek ligt, krijgen we voor elk frame een andere combinatie van bordposes. Aangezien de ruimtelijke relatie tussen projector en camera vast is, kunnen al deze bordposes gecombineerd worden om de relatieve positie van de projector ten opzichte van de camera af te leiden. Dit gebeurt via een kalibratie-algoritme dat, gegeven de poses van het kalibratiebord ten opzichte van beide toestellen, de vaste transformatie bepaalt die de camera- en projectorcoördinaten met elkaar verbindt. Het algoritme zoekt dus de extrinsieke matrix die al die relatieve bordposes het best omschrijft. Na deze eerste schatting wordt meestal nog een optimalisatiestap uitgevoerd. Daarbij wordt de volledige *reprojection error* over alle frames en punten geminimaliseerd, door tegelijk de poses van het bord en de extrinsieke matrix zelf te verfijnen. Het resultaat is een nauwkeurige en robuuste schatting van de extrinsieke matrix tussen camera en projector.

4.2.6 Extrinsieke matrix

De extrinsieke matrix $\mathbf{T}$ bevat de ruimtelijke relatie tussen de camera en de projector:

$$\mathbf{T} = [R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

R De 3×3 rotatiematrix beschrijft de oriëntatie van het ene coördinatenstelsel ten opzichte van het andere. In het geval van een projector-camerasysteem geeft deze matrix aan hoe het assenstelsel van de projector geroteerd is ten opzichte van dat van de camera. Elke rij of kolom van de matrix komt overeen met een richtingsvector van het lokale assenstelsel. De matrix is orthonormaal, wat betekent dat de assen loodrecht op elkaar staan en een

eenheidslengte hebben.

- t** De translatievector beschrijft de verplaatsing ten opzichte van de oorsprong. In dit geval de verplaatsing van de projector ten opzichte van de camera en geeft dus aan waar de projector zich bevindt ten opzichte van de camera, gemeten in drie dimensies (x, y, z) .

Hoofdstuk 5

Tracking

In dit hoofdstuk worden de drie geïmplementeerde trackingmethodes uitgebreid besproken: markergebaseerde tracking, dieptegebaseerde tracking en SLAM-gebaseerde tracking. Elke methode probeert de 6-DOF-pose van het projector-camerasysteem ten opzichte van een vast object of omgeving te bepalen. Het doel van deze tracking is om in real time een correcte inschatting te maken van de positie en oriëntatie van het systeem, zodat de projectie correct afgestemd is op het fysieke object.

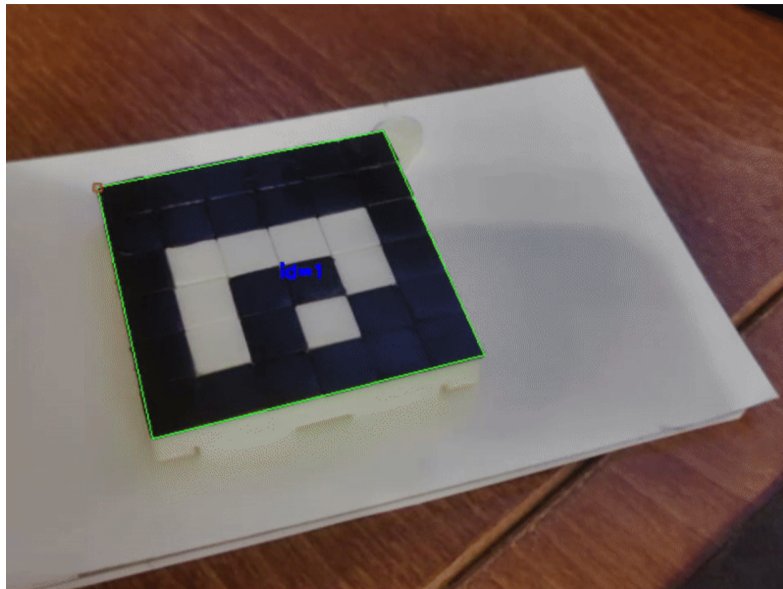
In de volgende secties wordt elk systeem in detail besproken, inclusief gebruikte algoritmes, parameters, voordelen, beperkingen en specifieke implementatiekeuzes.

5.1 ArUco-markertracking

De eerste methode maakt gebruik van visuele markers, meer bepaald ArUco-markers, die bevestigd zijn op het object waarvan de pose bepaald moet worden. Deze markers worden herkend in het camerabeeld en vormen vaste referentiepunten voor het oplossen van de camerageometrie ten opzichte van het object.

5.1.1 Detectie

De implementatie volgt een typische opbouw van detectie, validatie en poseberekening. De eerste stap is het detecteren van de ArUco-markers in het camerabeeld. Hiervoor wordt gebruikgemaakt van de OpenCV-bibliotheek, die deze functionaliteit ondersteunt. Het camerabeeld wordt vastgelegd en geanalyseerd met het detectiealgoritme. Indien er markers worden gedetecteerd, geeft het algoritme de hoeken in pixelcoördinaten terug, samen met de identificatoren van de markers en een lijst van onzekere detecties. Daarna wordt de detectie verfijnd met behulp van het ArUco-bord in combinatie met de eerder bepaalde parameters. Op basis van de gekende geometrie van het ArUco-bord kan het algoritme bepalen of er nog bijkomende markers gevonden kunnen worden. Hiervoor interpoleert het algoritme op basis van gedetecteerde markers waar de niet-gevonden markers kunnen staan. Op deze manier kunnen extra markers worden herkend, zelfs bij gedeeltelijke occlusie of beeldruis. Figuur 5.1 toont een voorbeeld van een gedetecteerde marker.



Figuur 5.1: Een gedetecteerde ArUco-marker, aangeduid met de groene rand.

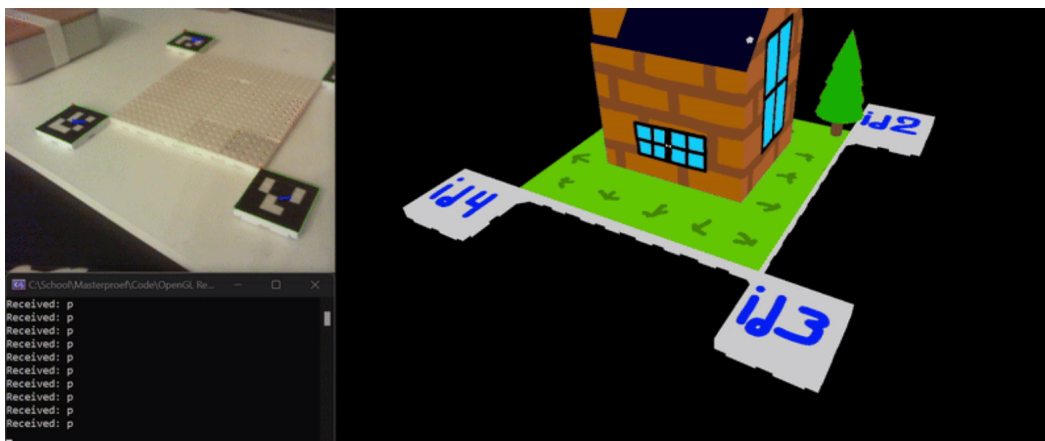
5.1.2 Pose estimation

Na het detecteren van de markers kan de pose van het ArUco-bord bepaald worden ten opzichte van de camera met behulp van de 2D-pixelcoördinaten en de 3D-objectpunten van het bord. Hierbij is het belangrijk dat de overeenkomstige 3D- en 2D-punten in identiek dezelfde volgorde staan. Met de overeenkomstige punten kan het PnP-probleem opgelost worden.

Wanneer het aantal gedetecteerde markers onder een drempelwaarde ligt, wordt de meting als ongeldig beschouwd. In de praktijk wordt meestal een minimum van drie gedetecteerde markers vereist voor een betrouwbare pose. Hoewel één marker technisch volstaat voor een pose-inschatting, verhoogt een groter aantal markers de betrouwbaarheid van het resultaat.

5.1.3 Visualisatie

Voordat de pose verzonden wordt naar de visualisatie, moet deze eerst gecombineerd worden met de extrinsieke matrix van het ProCam-systeem. Hierdoor verschuift het referentiepunt van de pose van de camera naar de projector, en slotte wordt de berekende pose via een UDP-verbinding doorgestuurd naar de visualisatie. Figuur 5.2 toont de tracking in combinatie met de visualisatie.



Figuur 5.2: Tracken van een ArUco-markerbord, gekoppeld aan de visualisatie.

5.2 Dieptegebaseerde tracking

De dieptegebaseerde tracking maakt gebruik van een vooraf ingeladen puntenwolk van het te tracken object en vergelijkt deze met de puntenwolk die gegenereerd wordt door de dieptecamera in real time. In tegenstelling tot markertracking zijn er geen visuele aanwijzingen zoals markers vereist.

5.2.1 3D-model omvormen naar puntenwolk

Deze trackingmethode vergelijkt twee puntenwolken met elkaar. Daarom dient voorafgaand aan de tracking een puntenwolk van het 3D-model gegenereerd te worden. Dit wordt gerealiseerd door het 3D-model in te laden en willekeurig punten over het oppervlak te verspreiden.

5.2.2 Puntenwolkregistratie met dieptesensor

Vervolgens kan de tracking starten. De camera registreert tegelijk een RGB-beeld en een dieptebeeld. Uit elk dieptebeeld wordt er een puntenwolk berekend. Deze puntenwolk wordt berekend met behulp van de intrinsieke parameters van de camera om per pixel een 3D-positie te reconstrueren.

5.2.3 Puntenwolken matchen

De puntenwolk van het 3D-model wordt gematcht met de puntenwolk van de camera met behulp van de KISS-matcher [13]. De matcher zoekt een rigide transformatie die de optimale alignering tussen de twee puntenwolken beschrijft.

De KISS-matcher is een globaal registratie-algoritme en vindt de transformatie tussen puntenwolken door gebruik te maken van een combinatie van technieken. Figuur 2.3 toont een schematisch overzicht van het globale registratie-algoritme. Hieronder worden de vier hoofdcomponenten van het algoritme toegelicht.

De KISS-matcher begint met het uitvoeren van een geometrische suppressie. Dit zorgt ervoor dat punten met weinig onderscheidende geometrische kenmerken, zoals vlakke oppervlakken, verwijderd worden. Dit draagt bij aan het reduceren van *outliers* (foute matches) en verbetert de efficiëntie van het algoritme.

Vervolgens wordt het Faster Point Feature Histogram (Faster-PHF) op de puntenwolken toegepast. Dit algoritme extraheert lokale geometrische kenmerken van punten in een puntenwolk. Dit algoritme is een geoptimaliseerde variant van het Fast Point Feature Histogram-algoritme en is geoptimaliseerd voor snelheid en nauwkeurigheid.

Na het matchen van kenmerken tussen de twee puntenwolken, wordt een graaf opgebouwd waarin de knopen de overeenkomsten voorstellen. Door k-core-decompositie toe te passen, worden zwakke verbindingen (mogelijk foutieve overeenkomsten) verwijderd, wat resulteert in een robuustere set matches voor de uiteindelijke registratie.

De laatste stap is het berekenen van de optimale rotatie en translatie die de twee puntenwolken op elkaar afstemt. Hiervoor gebruikt KISS-matcher een Graduated Non-Convexity (GNC) methode, die robuust is tegen *outliers* en geen initiële schatting vereist.

5.2.4 Visualisatie

Analoog aan de markertracking zal de transformatiematrix eerst nog gecombineerd moeten worden met de extrinsieke matrix. Zo verandert het referentiepunt van de pose opnieuw van de camera naar de projector. Ten slotte wordt deze pose verstuurd naar de visualisatie via een UDP-verbinding.

5.2.5 Beperkingen

Tijdens de ontwikkeling en het testen van deze trackingmethode werd een probleem vastgesteld. De KISS-matcher kan de puntenwolken niet correct met elkaar matchen. Dit heeft twee oorzaken, oorzaak één is dat er ruis zit in de puntenwolk van de camera. Daardoor vormt de puntenwolk geen perfecte representatie van de omgeving vanuit het perspectief van de camera. De tweede oorzaak is dat de camera telkens maar de helft van het te tracken object kan zien. Dit impliceert dat de KISS-matcher de puntenwolk van het 3D-model moet gaan matchen met maximaal de helft van de informatie die aanwezig kan zijn in de puntenwolk van de camera.

De combinatie van deze twee factoren en de beperkte afmetingen van het object leidt ertoe dat de trackingmethode faalt. Om deze reden wordt de methode niet verder geëvalueerd in het kader van deze studie.

5.3 SLAM-gebaseerde tracking

De derde trackingmethode baseert zich op een SLAM-algoritme voor het bepalen van de 6-DOF-pose van het projector-camerasysteem. In tegenstelling tot de andere methodes wordt het object hier beschouwd als een vast onderdeel van de omgeving, waarbij uitsluitend het projector-camerasysteem vrij kan bewegen. De tracking wordt uitgevoerd met behulp van Stella-VSLAM, een open-source SLAM-systeem gebaseerd op OpenVSLAM [3]. Deze methode vereist een kalibratiefase waarin de relatie tussen het SLAM-coördinatensysteem en het objectcoördinatensysteem wordt vastgesteld. Figuur 2.4 toont een voorbeeldvisualisatie van een SLAM-map.

5.3.1 SLAM-initialisatie en markerconfiguratie

Voor aanvang van de tracking dient Stella-VSLAM correct geconfigureerd te worden. Dit gebeurt via een configuratiebestand waarin onder meer het cameratype, de resolutie, framerate en de gebruikte markers worden gespecificeerd. In dit onderzoek wordt een monoculaire RGB-camera gebruikt. Aangezien een monoculaire camera geen absolute schaalinformatie ter beschikking heeft, is additionele informatie vereist voor het bepalen van de schaal van het SLAM-model. Daarom worden vier ArUco-markers toegevoegd aan de omgeving (zie figuur 3.4), waarvan de posities in het objectcoördinatensysteem bekend zijn.

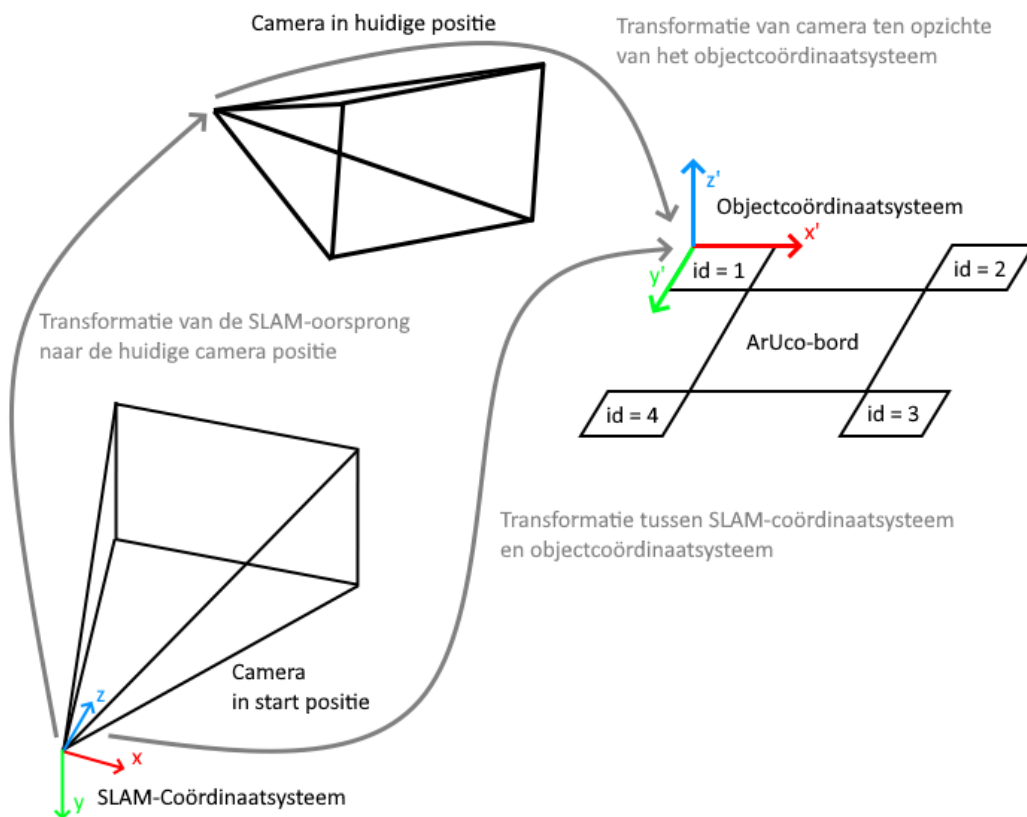
Na initiatie verwerkt Stella-VSLAM continu de camerabeelden en detecteert visuele kenmerken (*features*) en markers. De gedetecteerde markers worden geïntegreerd in de SLAM-map en gebruikt om de camera-pose nauwkeuriger te schatten.

5.3.2 Transformatie tussen SLAM en object

Omdat de bewegingen van de camera ten opzichte van het SLAM-coördinatensysteem worden geregistreerd, is het noodzakelijk de relatie tussen dit coördinatensysteem en het objectcoördinatensysteem te bepalen. Dit omvat twee stappen: het bepalen van de transformatiematrix en het schatten van de schaalfactor. Figuur 5.3 geeft de onderstaande uitleg schematisch weer.

De eerste stap is het berekenen van de transformatiematrix tussen het object en de camera. Dit gebeurt op een manier die analoog is aan de ArUco-trackingmethode. De vier markers die zich op bekende posities bevinden in het objectcoördinatensysteem worden gedetecteerd in het camerabeeld. Vervolgens wordt het PnP-algoritme toegepast om de camera-pose ten opzichte van het object te bepalen. De output is een 4×4 transformatie die de positie en oriëntatie van de camera in het objectcoördinatensysteem beschrijft.

Aangezien het SLAM-algoritme geen absolute schaal kan bepalen bij gebruik van een monoculaire camera, wordt de schaal geschat door de markerposities in de SLAM-map te vergelijken met de objectpunten. Door een affine transformatie te berekenen tussen deze twee verzamelingen van coördinaten, kan een schaalfactor geschat worden. Deze schaalfactor wordt vervolgens gebruikt om de SLAM-pose van de camera om te zetten naar dezelfde schaal als het objectcoördinaatsysteem (in dit geval metrische schaal). De affine transformatie wordt niet verder toegepast vanwege onjuiste berekeningen van rotatie en translatie.



Figuur 5.3: Schematische voorstelling van hoe de transformatiematrix tussen het SLAM-coördinaatsysteem en het objectcoördinaatsysteem gevonden kan worden.

5.3.3 Berekenen van de uiteindelijke transformatie

De uiteindelijke transformatie tussen SLAM- en objectcoördinaten wordt bepaald door meerdere stappen te combineren:

1. De camerapose uit SLAM wordt geschaald met behulp van een schaalmatrix S . Vergelijking 3.1 toont hoe de schaalmatrix opgesteld wordt. Deze matrix wijzigt uitsluitend de schaal zonder rotatie of translatie te beïnvloeden.
2. Vervolgens wordt deze geschaalde pose gecombineerd met de eerder berekende object-naar-camera-transformatiematrix. Door matrixvermenigvuldiging wordt de camera-pose correct geconverteerd naar het objectcoördinatensysteem.

Deze transformatie wordt nu als referentie gebruikt tijdens het tracken. Voor elk nieuw camera-beeld waarin een nieuwe SLAM-pose van de camera wordt berekend, wordt dezelfde schaling en transformatiematrix toegepast om de pose van de camera ten opzichte van het object te bepalen.

5.3.4 Visualisatie

Zoals bij de andere trackingmethoden wordt de verkregen pose gecombineerd met de extrinsieke matrix van het projector-camerasysteem. Hierdoor wordt het referentiepunt van de pose verplaatst van de camera naar de projector. Ten slotte wordt de volledige pose via een UDP-verbinding naar de visualisatie verzonden.

Hoofdstuk 6

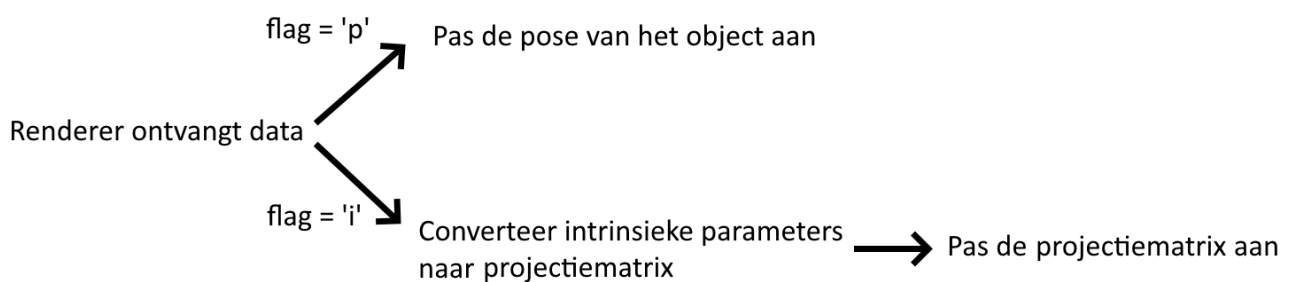
Visualisatie

Een essentieel onderdeel van het systeem betreft de correcte visualisatie van het getrackte object op het projectievlak. De visualisatiecomponent zorgt ervoor dat het virtuele object in real time op de correcte positie en oriëntatie wordt weergegeven vanuit het perspectief van de projector. Dit vereist een nauwkeurige integratie van de objectpose, de extrinsieke en intrinsieke parameters van zowel camera als projector, alsook een efficiënte rendertechniek.

6.1 Renderer

Voor dit project werd een eenvoudige OpenGL-renderer ontwikkeld die het ingeladen object visualiseert op basis van de ontvangen trackinggegevens. De renderer ondersteunt het inladen van 3D-objecten in het .obj-formaat, inclusief de bijbehorende texturen. Tijdens de uitvoering van het programma wordt het model gerenderd op basis van de meest recente pose- en projectiematrix.

Aangezien alle componenten van het systeem op één computer draaien, is de renderer bewust minimalistisch ontworpen om reken capaciteit te reserveren voor andere systeemonderdelen zoals tracking en communicatie. Figuur 6.1 toont een overzicht van de visualisatiepipeline.



Figuur 6.1: Overzicht van de visualisatiepipeline: data vanuit tracking wordt doorgestuurd naar de OpenGL-renderer die de visualisatie aanpast.

De pipeline functioneert als volgt: de renderer ontvangt gegevens via een UDP-verbinding. De ontvangen data bevat een *flag* die de renderer gebruikt om te bepalen welke matrix geüpdatet dient te worden. Wanneer de *flag* de waarde 'p' heeft, interpreteert de renderer dit als een nieuwe objectpose en wordt de posematrix van het 3D-object aangepast en weergegeven. Indien de *flag* de waarde 'i' heeft, betekent dit dat nieuwe intrinsieke parameters zijn ontvangen, waarop de renderer de projectiematrix bijwerkt.

6.1.1 Projectiematrix

Om de visualisatie correct weer te geven, is een juiste projectiematrix nodig die overeenkomt met de interne parameters van de projector. Deze matrix wordt bepaald op basis van de intrinsieke parameters die de renderer ontvangt. Deze matrix wordt in real time via een UDP-verbinding naar de renderer verzonden, zoals eerder beschreven.

De projectiematrix bepaalt hoe de 3D-wereld geprojecteerd wordt op het beeldvlak van de projector. Deze matrix is essentieel voor de correcte rendering van het object, zodat het nauwkeurig gealigneerd wordt met het fysieke object.

6.1.2 Objectpose

Naast de projectiematrix is ook de pose van het te visualiseren object noodzakelijk. Deze pose wordt op elk moment berekend door het actieve trackingalgoritme en doorgestuurd naar de renderer. Binnen de renderer wordt deze matrix toegepast als transformatie op het 3D-model, zodat het object correct georiënteerd is.

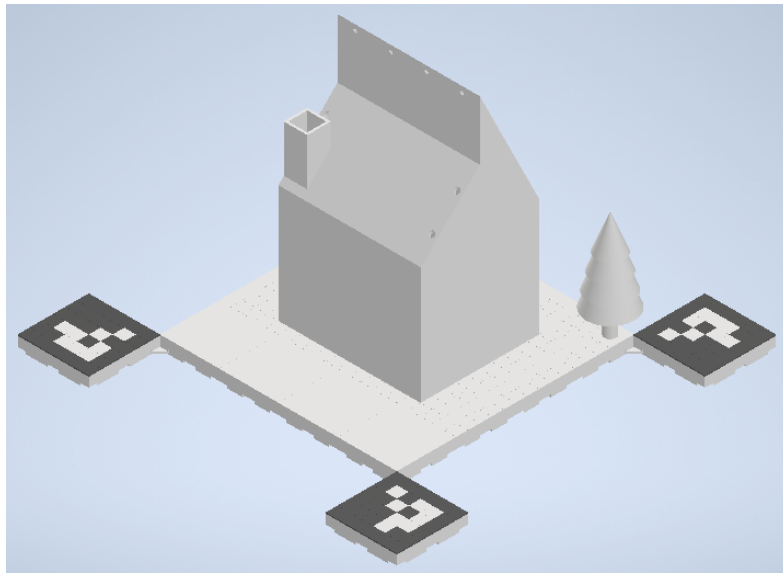
De combinatie van projectiematrix en objectpose garandeert dat het geprojecteerde beeld overeenkomt met de reële positie van het object in de fysieke omgeving.

6.2 3D-model

Het geprojecteerde object wordt gemodelleerd en voorbereid op een correcte en efficiënte weergave binnen de renderer. Dit omvat zowel het 3D-tekenen en printen van het fysieke object als het voorbereiden van het virtuele model dat gebruikt wordt in de visualisatiepipeline.

6.2.1 3D-tekenen en printen

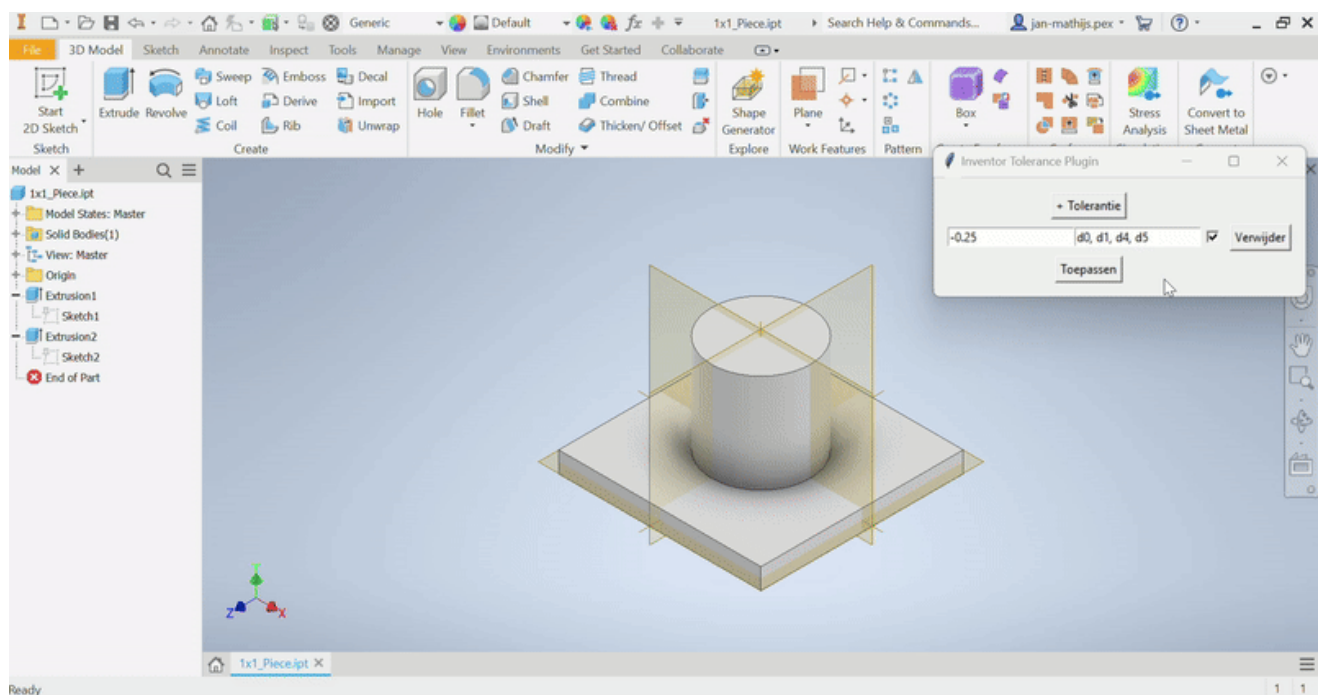
Het fysieke object werd ontworpen in Autodesk Inventor en onderverdeeld in meerdere componenten. Deze modulariteit maakt het mogelijk om elk onderdeel afzonderlijk te 3D-printen, hetgeen het printproces vereenvoudigt. Het uiteindelijke ontwerp van het te tracken object wordt weergegeven in figuur 6.2.



Figuur 6.2: Het finale ontwerp van het te tracken object.

Ter bevordering van flexibiliteit en ter voorkoming van printproblemen zijn enkele automatiseringen geïmplementeerd. Er werd een Python-script geschreven dat zich koppelt aan het Inventor-proces via de API. Hiermee kunnen meerdere afmetingen tegelijk aangepast worden. Dit is handig voor het snel toevoegen van toleranties, zodat de geprinte onderdelen goed in elkaar passen ondanks de kleine afwijkingen die typisch zijn voor 3D-printers. Zonder deze toleranties zouden onderdelen mogelijk niet correct passen, aangezien de afmetingen exact overeenkomen.

Dankzij het Python-programma zal het maken van het 3D-model, zowel fysiek als virtueel, sneller en eenvoudiger verlopen. Hierdoor kan efficiënt worden geschakeld tussen de versie met toleranties voor het fysieke object en de versie zonder toleranties voor het samengestelde 3D-model. Figuur 6.3 geeft de inventor-plugin weer.



Figuur 6.3: De Inventor-plugin die gebruikt werd voor het maken en 3D-printen van het model.

Het toevoegen van toleranties aan een model met bestaande restricties kan bovendien conflicten veroorzaken, waarbij er twee of meerdere restricties op dat 3D-model niet meer tegelijk geldig kunnen zijn.

Na het tekenen en printen wordt er een volledige samenstelling van het object aangemaakt in Inventor. Deze samenstelling wordt vervolgens geëxporteerd als één nieuw 3D-model. Om een correcte visuele weergave te bekomen, wordt dit model geïmporteerd in Blender voor verdere verwerking.

48

6.2.3 3D-model inladen

De renderer bevat functionaliteit om .obj-bestanden correct te kunnen inlezen. Hiervoor werd een eenvoudige OBJ-parser ontwikkeld, afgestemd op de vereisten van dit project. De parser opent het bestand en leest systematisch alle gegevens in, zoals de *vertices*, textuurcoördinaten, normaalvectoren en de *faces*.

Vertices zijn driedimensionale punten die de snijpunten van *edges* representeren, bijvoorbeeld hoekpunten. *Edges* zijn lijnen die *vertices* met elkaar verbinden. Deze verbindingen zijn niet willekeurig, maar worden bepaald door de structuur van de *faces*. De *faces* bepalen tussen welke *vertices*, en dus ook tussen welke *edges*, er een vlak gevormd wordt. Een *face* bestaat typisch uit drie punten, maar kan ook worden gedefinieerd door een groter aantal punten.

Textuurcoördinaten, of UV-coördinaten, bepalen hoe de 2D-textuur op het 3D-model wordt geprojecteerd. De normaalvectoren geven weer hoe licht interageert met het oppervlak van het model. In systemen met virtuele belichting kunnen deze vectoren gebruikt worden voor reflecties of schaduwen. Binnen dit project worden de normaalvectoren niet benut, aangezien er geen virtuele belichting voorzien is in de OpenGL-omgeving.

Deze gegevens worden vervolgens omgezet naar een interne datastructuur die compatibel is met OpenGL, zo kan het model correct worden weergegeven. Deze aanpak biedt volledige controle over het inladen en verwerken van het 3D-model, wat voordelig is voor foutopsporing en potentiële toekomstige uitbreidingen.

Hoofdstuk 7

Evaluatie

Om de performantie van de geïmplementeerde trackingmethodes te evalueren, werd het Qualisys motion-capturesysteem gehanteerd als referentie, dat in deze context als *ground truth* wordt beschouwd. Dit systeem biedt een hoge nauwkeurigheid en levert posedata aan een frequentie van 300 Hz, hetgeen ruim toereikend is voor de validatie van real time trackingmethodes.

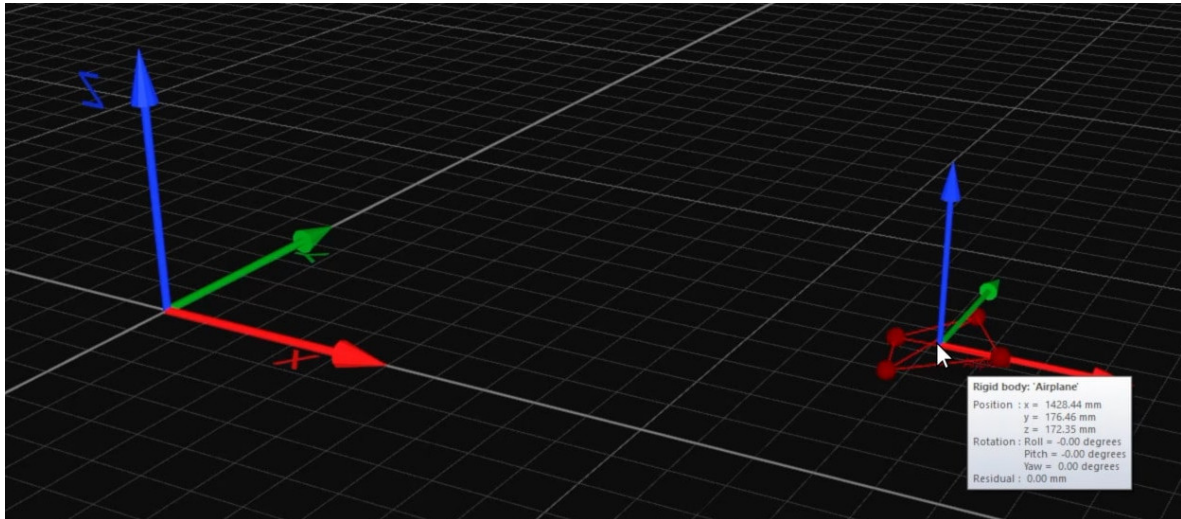
7.1 Voorbereiden van Qualisys-systeem

Vooraleer metingen konden worden uitgevoerd, diende het Qualisys-systeem eerst te worden gekalibreerd. Deze kalibratie gebeurde met behulp van een assenstelsel en een zogenaamde *calibration wand*, een staaf voorzien van twee retro-reflectieve markers op een vaste onderlinge afstand van elkaar. Beide zijn te zien in figuur 7.1. Door hiermee het kalibratievolume te doorlopen, kon het systeem de ruimtelijke positie en oriëntatie van de camera's nauwkeurig vaststellen, wat essentieel is voor het verkrijgen van betrouwbare motion-capturegegevens.



Figuur 7.1: Het kalibratieassenstelsel en de *calibration wand* die gebruikt worden om een Qualisys-systeem te kalibreren.

Na succesvolle kalibratie werd het systeem geconfigureerd om een *rigid body* te tracken die bevestigd was op het projector-camerasysteem. Deze *rigid-body* bestond uit een set van drie reflectieve markers die stevig aan het systeem bevestigd waren, zodat elke systeembeweging accuraat werd weerspiegeld in de geregistreerde poses. In figuur 3.5 is het projector-camerasysteem te zien met deze retro-reflectieve markers. Hierdoor kon het volledige projector-camerasysteem als één geheel met hoge precisie worden getrackt. Figuur 7.2 toont een voorbeeld van *rigid-body* tracking met een Qualisys-systeem.



Figuur 7.2: Een voorbeeld van een *rigid-body* die getrackt wordt door een Qualisys-systeem.

7.2 Gegevensverzameling

Tijdens de evaluatie werd het projector-camerasysteem gelijktijdig getrackt door zowel het Qualisys-systeem als door zijn eigen interne trackingmethode. Bij iedere nieuwe posebepaling door het systeem via zijn eigen trackingalgoritme, wordt gelijktijdig de meest recente pose van het Qualisys-systeem opgevraagd. Beide poses, de geschatte pose van de trackingmethode en de *ground truth* van Qualisys, worden gezamenlijk opgeslagen. Daarnaast wordt het bijbehorende framenummer van het Qualisys-systeem ook opgeslagen. Deze aanpak garandeert dat elke gegenereerde trackingpose gekoppeld is aan een zo nauwkeurig mogelijke referentiepose.

De gegevensverzameling vindt plaats via UDP-communicatie. De poses van de trackingmethodes worden vanuit de C++ code in real time doorgestuurd naar een Python-script dat verantwoordelijk is voor het ontvangen en verwerken van de gegevens. De poses van het Qualisys-systeem worden ook in real time doorgestuurd naar hetzelfde script. Het script slaat de inkomende poses op in een JSON-bestand telkens wanneer een pose van een trackingmethode wordt ontvangen. Het JSON-formaat is gekozen wegens de flexibiliteit bij het opslaan van complexe gegevensstructuren zoals matrices.

Voor beide trackingmethodes werd data verzameld onder twee verschillende omstandigheden: één keer met de projector ingeschakeld en één keer met de projector uitgeschakeld. Op deze manier wordt tevens geëvalueerd of actieve projectie invloed heeft op de trackingprestaties, bijvoorbeeld door interferentie met de detectie van ArUco-markers of SLAM-kenmerken.

7.3 Datapreprocessing

Voordat de afwijkingen tussen de tracking-poses en de *ground truth* kunnen worden geanalyseerd, moeten de poses correct worden uitgelijnd. Omdat beide systemen hun eigen referentie- of coördinatensysteem gebruiken, is het noodzakelijk om de relatieve transformatie tussen deze systemen te bepalen.

In de eerste stap wordt het referentiepunt van alle poses verschoven naar de eerste pose in de reeks. Hierdoor worden de poses van zowel de trackingmethode als van Qualisys uitgedrukt ten opzichte van een gemeenschappelijk beginpunt, in plaats van binnen hun eigen coördinatensysteem. In een tweede stap wordt de uitlijning bepaald tussen de poses van de trackingmethode en die van het Qualisys-systeem. Daartoe wordt een deelverzameling van de data gebruikt om een optimale rigide transformatie (rotatie, translatie en eventueel schaal) te schatten. Nadat de datasets correct zijn uitgelijnd, kunnen de individuele poses onderling worden vergeleken op basis van positionele fout.

7.4 Foutberekening

Voor elke pose wordt de afwijking ten opzichte van de *ground truth* berekend. De positionele fout wordt gedefinieerd als de euclidische afstand tussen de pose van de trackingmethode en die van het Qualisys-systeem, uitgedrukt in millimeters.

De positionele fout wordt voor elk *frame* berekend en vervolgens geaggregeerd tot een gemiddelde foutwaarde. Daarnaast wordt ook de stabiliteit van de tracking bekeken door de trajecten afgeleid uit de poses te analyseren.

7.5 Analyse van de latency en frequentie

Om inzicht te krijgen in de prestaties van de trackingmethodes op vlak van snelheid en reactiviteit, wordt een analyse uitgevoerd van zowel de *latency* tussen opeenvolgende metingen als de effectieve trackingfrequentie (fps). Daarbij wordt gebruikgemaakt van het framenummer dat bij elke set van poses werd opgeslagen. Aangezien de Qualisys gegevens vastlegt met een constante frequentie van 300 Hz, kunnen deze framenummers direct omgezet worden naar tijdsintervallen.

7.5.1 Latency

De *latency* tussen twee opeenvolgende trackingmetingen wordt als volgt berekend: voor elk datapunt wordt het verschil genomen tussen het framenummer van de huidige pose en dat van de vorige pose. Aangezien elk *frame* overeenkomt met 1/300e seconde, wordt dit verschil gedeeld door 300 om de tijdsduur tussen beide metingen in seconden te verkrijgen. Vergelijking 7.1 toont de berekening van de *latency*.

$$\text{latency}_i = \frac{\text{framenummer}_i - \text{framenummer}_{i-1}}{300} \quad (7.1)$$

Deze *latency* kwantificeert de verstreken tijd tussen twee opeenvolgende updates van het trackingsysteem, zoals geregistreerd door Qualisys.

7.5.2 Frames per seconde

Op basis van deze latencywaarden kan de effectieve frequentie van het trackingsysteem bepaald worden, uitgedrukt in *frames* per seconde (fps). Dit wordt vastgesteld door te bepalen hoeveel trackingmetingen plaatsvinden binnen een tijdsinterval van één seconde.

Het algoritme verloopt als volgt: start bij een eerste meetpunt en tel opeenvolgend de latencywaarden op totdat de som van deze tussenliggende latencies één seconde bedraagt. Het aantal trackingberekeningen dat in deze tijdspanne werd uitgevoerd, geeft dan de lokale fps-waarde. Door dit proces over de volledige dataset te herhalen, kan een tijdsafhankelijke grafiek van de fps worden opgesteld.

Deze aanpak maakt het mogelijk om niet enkel het gemiddelde aantal fps te berekenen, maar ook de variatie ervan over tijd te observeren, wat nuttig is om te evalueren hoe stabiel de trackingfrequentie is tijdens dynamische bewegingen of onder verschillende lichtomstandigheden (bv. met of zonder projector).

Hoofdstuk 8

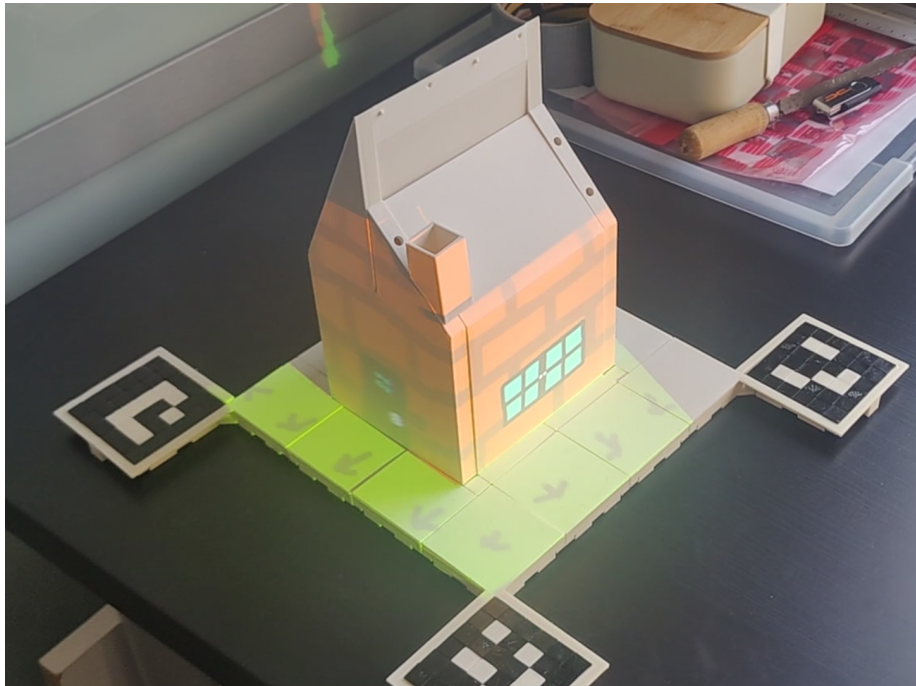
Resultaten

8.1 Empirische Evaluatie

Voor een initiële beoordeling van de werking van beide trackingmethodes wordt een visuele observatie uitgevoerd op basis van screenshots uit videofragmenten van het systeem in actie. Deze empirische evaluatie maakt het mogelijk om verschillen in stabiliteit en nauwkeurigheid te observeren.

8.1.1 Marker-gebaseerde tracking

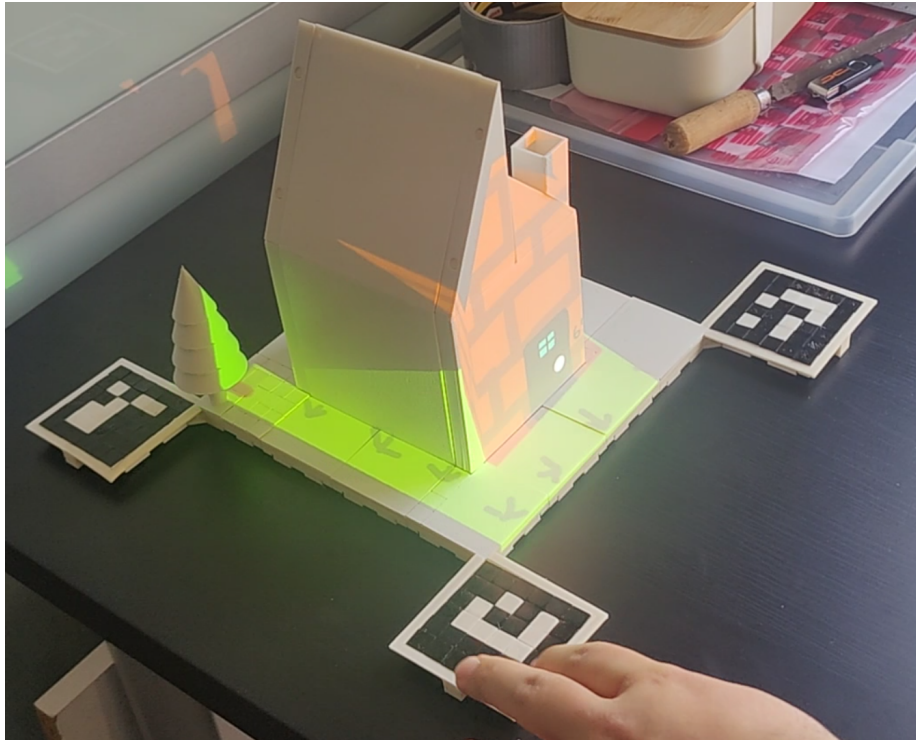
Figuur 8.1 toont een momentopname van het systeem in beweging, waarbij de projectie de maquette volgt met beperkte vertraging zolang de markers zichtbaar blijven.



Figuur 8.1: Screenshot van ArUco-markertracking tijdens beweging van het systeem.

Wanneer een of meerdere markers tijdelijk buiten beeld vallen, neemt de stabiliteit van de pose-inschatting af. In sommige gevallen ontstaat een korte hapering doordat het systeem een foutieve

pose berekent, zoals te zien in figuur 8.2. Zodra de markers opnieuw gedetecteerd worden, herstelt het systeem automatisch.

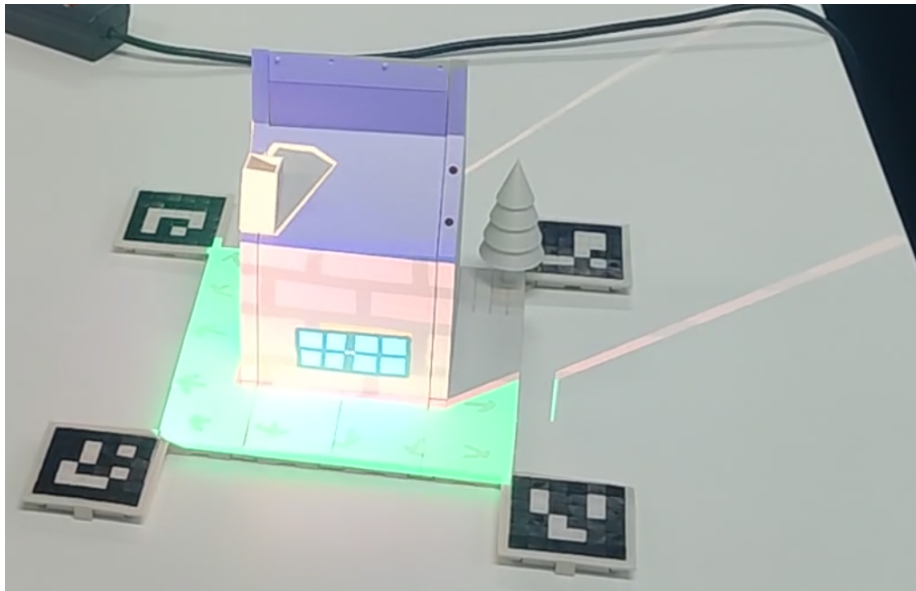


Figuur 8.2: Screenshot van ArUco-markertracking terwijl onvoldoende markers gevonden zijn voor robuuste pose-inschatting.

Ook kan, tijdens het maken van snellere bewegingen, de projectie niet correct gealigneerd zijn. Dit komt omdat de detectie van markers onstabiel kan zijn.

8.1.2 SLAM-gebaseerde tracking

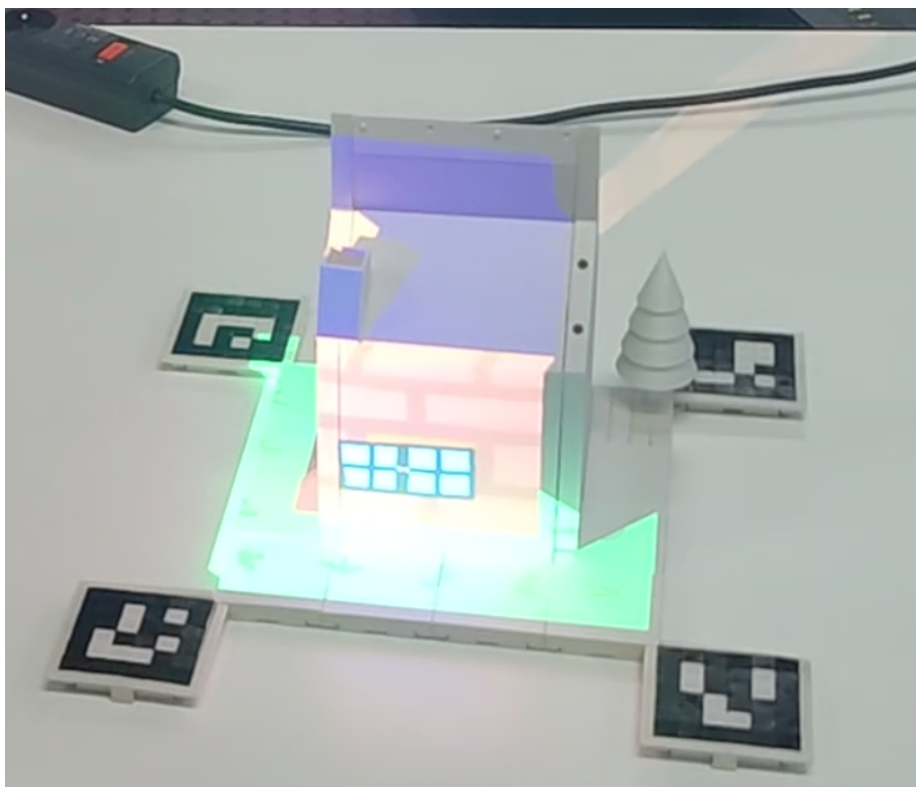
Figuur 8.3 toont dat SLAM ook een gealigneerde trackingervaring biedt, zonder nood aan constant zichtbare markers. Tijdens zeer langzame bewegingen en als het systeem volledig stilstaat, blijft de projectie gealigneerd.



Figuur 8.3: Screenshot van SLAM-tracking tijdens het systeem op dezelfde plaats gehouden wordt.

Ook hier is de projectie niet volledig gealigneerd, te zien aan twee lijnen die voorbij de randen van de maquette op de tafel vallen. Een waarschijnlijke oorzaak hiervoor is een onvoldoende accurate extrinsieke matrix van het ProCam-systeem.

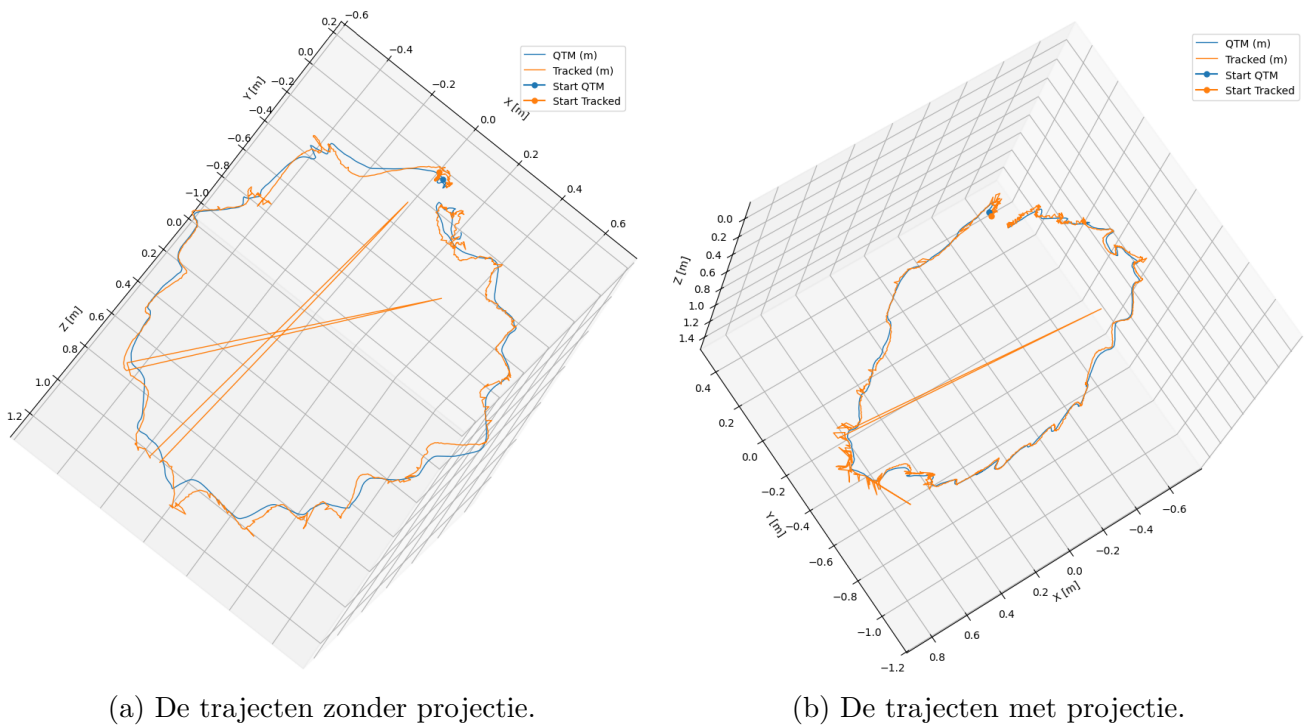
Bij snellere rotaties of verplaatsingen treedt echter merkbare vertraging op. Dit leidt tot tijdelijke misalignments tussen projectie en maquette. Daarnaast kan SLAM gevoelig zijn voor variaties in de geprojecteerde beelden, omdat geprojecteerde beelden kunnen interfereren met de visuele features die voor tracking worden gebruikt [18].



Figuur 8.4: Screenshot van SLAM-tracking tijdens snellere beweging.

8.1.3 Bewegingstrajecten

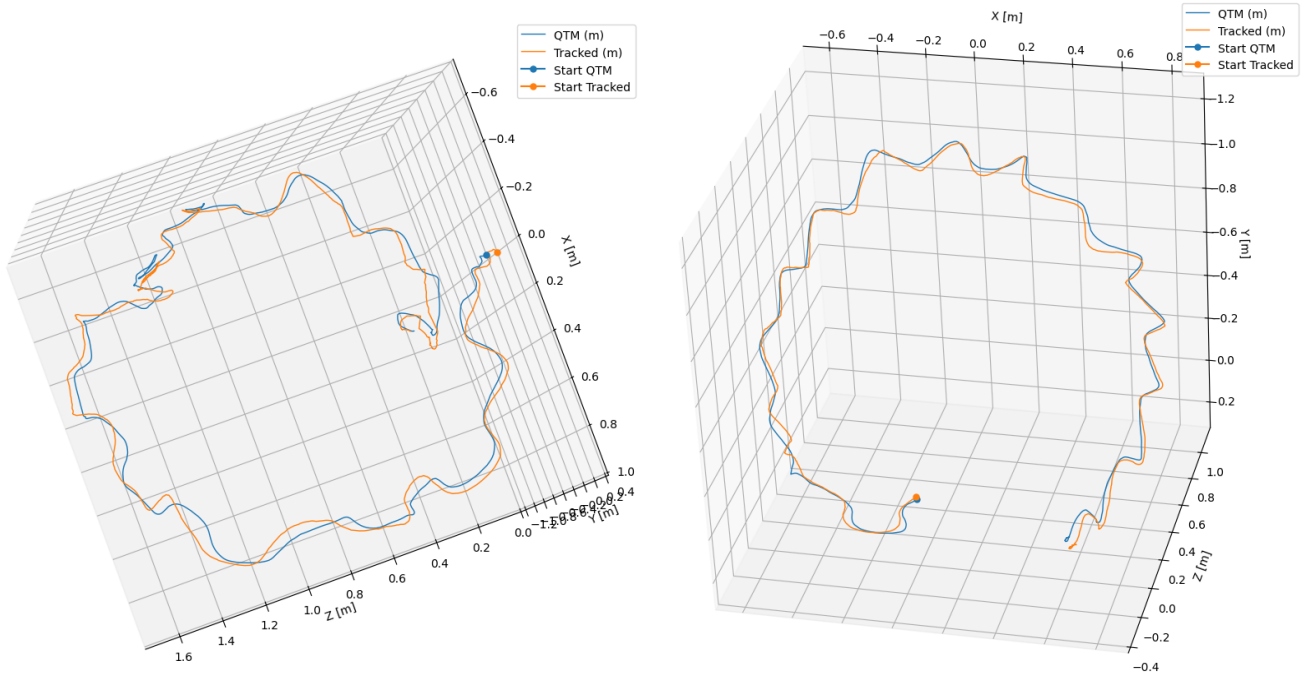
Figuur 8.5 toont de vergelijking tussen het traject van het Qualisys-systeem en het traject van de ArUco-markertracking. Figuur 8.5a toont het traject zonder projectie, terwijl figuur 8.5b het traject toont met projectie.



Figuur 8.5: Vergelijking tussen het traject van het Qualisys-systeem en de ArUco-markertracking.

Het eerder beschreven instabiele gedrag van het trackingsysteem vertaalt zich in de zichtbare uitschieters en de kleinere piekjes in het traject. Vermoedelijk zijn de grote uitschieters veroorzaakt door een foutieve pose-inschatting bij te weinig markers die zichtbaar zijn. Dit gedrag is in beide trajecten aanwezig, maar het traject met de projector ingeschakeld vertoont duidelijk meer van dit gedrag.

Vergelijkbaar met de resultaten bij markergebaseerde tracking toont figuur 8.6 de trajecten van de SLAM-tracking. Figuur 8.6a toont het traject zonder projectie, terwijl figuur 8.6b het traject toont met de projector ingeschakeld.



(a) De trajecten zonder projectie.

(b) De trajecten met projectie.

Figuur 8.6: Vergelijking tussen het traject van het Qualisys-systeem en de SLAM-tracking.

Daarentegen vertoont SLAM-tracking nauwelijks haperingen, dit is ook duidelijk te zien in beide trajecten. SLAM-tracking levert een stabielere posebepaling, waardoor het traject geen duidelijke uitschieters of plotse afwijkingen vertoont.

8.2 Kwantitatieve Metingen

Voor een objectieve vergelijking werd de geschatte pose van het systeem vergeleken met de data opgenomen door het Qualisys-systeem, dat gebruikt wordt als *ground truth*. Van elke trackingmethode werden twee metingen uitgevoerd: één met de projector uitgeschakeld en één met de projector ingeschakeld. Dit resulteerde in vier afzonderlijke configuraties, elk geëvalueerd op basis van één meetreeks.

De positionele fout, ook wel *Average Trajectory Error* (ATE) genoemd, werd berekend. Daarnaast werden de hoeveelheid *frames* per seconde (fps), ook wel *framerate* genoemd, en *latency* bepaald. De resultaten staan opgelijst per methode in tabel 8.1.

Tabel 8.1: Prestaties van beide trackingmethodes, met en zonder projectie.

Configuratie	Gemiddelde positionele fout (mm)	Gemiddelde framerate (fps)	Gemiddelde latency (ms)
ArUco-markertracking zonder projectie	30,13	28,53	33,90
ArUco-markertracking met projectie	18,79	29,07	33,56
SLAM-tracking zonder projectie	36,51	16,85	56,44
SLAM-tracking met projectie	26,48	18,24	52,13

8.2.1 Positionele fout

De gemiddelde positionele fout van de markergebaseerde methode bedraagt 30,13 mm wanneer de projector uitgeschakeld is. Bij ingeschakelde projector daalt deze fout tot 18,79 mm. Voor SLAM-tracking bedraagt de gemiddelde fout respectievelijk 36,51 mm zonder projectie en 26,48 mm met projectie.

Deze resultaten worden verder besproken in de volgende sectie.

8.2.2 Real time prestaties

De gemiddelde *framerate* en *latency* van het systeem werden per configuratie bepaald. ArUco-markertracking haalt gemiddeld 28,53 fps zonder projectie en 29,07 fps met projectie. De bijbehorende *latency* bedraagt respectievelijk 33,90 ms en 33,56 ms. SLAM-tracking behaalt 16,85 fps zonder projectie en 18,24 fps met projectie. De bijbehorende *latency* bedraagt respectievelijk 56,44 ms en 52,13 ms.

Daarnaast werd ook de laagst gemeten *framerate* genoteerd. Voor ArUco-markertracking zonder projectie bedroeg de minimumwaarde 23 fps (één keer gemeten), en met projectie 27 fps (drie keer gemeten). Bij SLAM-tracking was de laagste waarde 12 fps zonder projectie en 17 fps met projectie; beide waarden kwamen elk eenmaal voor.

Ook deze resultaten worden verder besproken in de volgende sectie.

8.3 Discussie

Uit de resultaten blijkt dat de gemiddelde positionele fout bij alle configuraties in de orde van millimeters ligt, met relatief kleine variaties tussen de toestanden. Hoewel de fout iets lager ligt bij ingeschakelde projector, valt dit verschil binnen de verwachte meetvariatie van de tracking-methodes.

Aangezien er slechts één meetreeks per configuratie beschikbaar is, kan niet met zekerheid worden vastgesteld of de verschillen systematisch zijn. De observaties suggereren dat de aanwezigheid van projectie geen significante invloed heeft op de positionele nauwkeurigheid. De waargenomen afwijkingen kunnen mogelijk verklaard worden door variaties in meetomstandigheden zoals belichting, oriëntatie of interne ruis in het algoritme. Een bredere testreeks is nodig om eventuele subtiele of systematische effecten betrouwbaar te kunnen detecteren.

De *framerate* en *latency* vertonen weinig variatie binnen eenzelfde trackingmethode. Het aan- of uitzetten van de visualisatie heeft dus verwaarloosbare invloed op de prestaties van het systeem. Dit bevestigt dat de eigen ontwikkelde renderer haar doel bereikt: de real time projectie van de visualisatie zonder significante impact op de tracking.

Een opmerkelijk verschil is dat de SLAM-methode 37 tot 40% minder *frames* per seconde verwerkt dan de ArUco-markertracking. Dat verschil is te verwachten, aangezien SLAM voortdurend een kaart opbouwt en optimaliseert. Dit is computationeel intensiever dan enkel markerherkenning en pose-schatting. De hogere complexiteit verklaart ook de grotere *latency* bij SLAM.

Hoewel SLAM een lagere *framerate* heeft, leverde dit tijdens de evaluatie een consistentere schatting van de camerapose op. Markertracking vertoonde af en toe haperingen wanneer markers

deels buiten beeld raakten, wat bij SLAM niet voorkwam.

Door de hogere *latency* van de SLAM-tracking reageert de projectie trager op bewegingen van het systeem. Bij markergebaseerde tracking was dit nauwelijks merkbaar, omdat de *latency* daar laag genoeg blijft om bewegingen vrijwel direct te volgen. Deze bevinding wordt bevestigd door de laagst gemeten *framerate*: die ligt bij SLAM niet alleen lager dan bij markertracking, maar zelfs de gemiddelde *framerate* van de SLAM-tracking ligt lager dan de minimale waarde van de markertracking.

Hoofdstuk 9

Besluit

In dit onderzoek werd nagegaan welke trackingmethode het meest geschikt is voor toepassing in dynamische projection mapping op schaalmodellen. Aan de hand van een grondige literatuurstudie en experimentele evaluatie werden drie benaderingen onderzocht: marker-gebaseerde tracking, dieptegebaseerde registratie, en SLAM-gebaseerde tracking.

De dieptegebaseerde techniek bleek in de praktijk onvoldoende robuust en werd daarom niet verder opgenomen in de experimentele vergelijking. De resterende methodes, ArUco-markertracking en SLAM-gebaseerde tracking, werden vergeleken op vlak van nauwkeurigheid, stabiliteit, real time prestaties en implementatiecomplexiteit.

De resultaten tonen aan dat marker-gebaseerde tracking zeer betrouwbare en nauwkeurige pose-inschattingen levert, met een gemiddelde positionele fout van slechts 18,79 mm en een stabiele framerate van gemiddeld 29,07 fps, op voorwaarde dat de markers goed zichtbaar blijven. SLAM daarentegen behaalde een gemiddelde fout van 26,48 mm en een framerate van 18,24 fps. SLAM-tracking blijkt ook meer stabiele tracking te bieden dan markergebaseerde tracking, maar heeft een hogere gemiddelde *latency*, wat resulteert in vertraging van de projectie-updates.

Op basis van deze bevindingen kan geconcludeerd worden dat marker-gebaseerde tracking momenteel de meest praktische keuze vormt voor dynamische projection mapping op maquettes, zeker in gecontroleerde omgevingen. Het is een relatief snelle en eenvoudig op te zetten methode. SLAM-gebaseerde tracking blijft echter een veelbelovende techniek, vooral wanneer stabiliteit boven snelheid verkozen wordt. Bovendien maakt SLAM het systeem mobieler. Verdere optimalisaties in robuustheid en filtering van projectie-invloeden kunnen SLAM op termijn evenwaardig of zelfs beter maken dan marker-gebaseerde oplossingen.

Hoewel de resultaten veelbelovend zijn, kent dit onderzoek ook enkele beperkingen. Zo werden de metingen beperkt tot een klein aantal testreeksen en vond de evaluatie plaats in een relatief gecontroleerde omgeving. Verdere studies met uitgebreidere datasets en diverse realistische scenario's zijn nodig om de generaliseerbaarheid van de bevindingen te versterken.

Voor toekomstig onderzoek is het interessant om SLAM-tracking volledig markerloos te maken, waarbij de focus ligt op het verbeteren van de robuustheid en nauwkeurigheid, met speciale aandacht voor het verlagen van *latency*. Daarnaast zijn er binnen dieptegebaseerde tracking nog diverse mogelijkheden voor verbetering, zoals het ontwikkelen van snellere en nauwkeurigere matchingmethoden. Ook kan het proces van het genereren van point clouds uit dieptebeelden

geoptimaliseerd worden met de focus op snelheid en ruisreductie. Deze ontwikkelingen kunnen gezamenlijk bijdragen aan een flexibeler dynamisch projection mapping-systeem dat beter geschikt is voor complexe en veranderlijke omgevingen.

Dankwoord

Een taalmodel van OpenAI, ChatGPT [21] versie 4o, werd gebruikt om grammaticale correctheid en tekstuele helderheid te verbeteren. Alle inhoud en context werden ontwikkeld door de auteur.

Literatuurlijst

- [1] Disney Campus, “The science behind the magic: Disney enchantment.” <https://www.disneycampus.com/blog/the-science-behind-the-magic-disney-enchantment/>, 2022. Accessed: 2025-06-04.
- [2] Wikipedia contributors, “Artag — wikipedia, the free encyclopedia.” <https://en.wikipedia.org/wiki/ARTag>, 2025. Accessed: 2025-06-04.
- [3] S. Sumikura, M. Shibuya, and K. Sakurada, “OpenVSLAM: A Versatile Visual SLAM Framework,” in *Proceedings of the 27th ACM International Conference on Multimedia*, MM ’19, (New York, NY, USA), pp. 2292–2295, ACM, 2019.
- [4] camcalib.io, “What is the reprojection error?,” 2022. Accessed: 2025-06-04.
- [5] OpenCV, “Camera calibration and 3d reconstruction — opencv documentation.” https://docs.opencv.org/4.x/d9/d0c/group_calib3d.html, 2024. Accessed: 2025-06-06.
- [6] B. Miller, “Camera modeling: Exploring distortion and distortion models — part i.” <https://www.tangramvision.com/blog/camera-modeling-exploring-distortion-and-distortion-models-part-i>, 2022. Accessed: 2025-06-06.
- [7] OpenCV, “solvepnp - opencv documentation,” 2023. Accessed: 2025-06-04.
- [8] E. Luta, “Hololens 2: Ar applications for more efficiency in companies.” <https://augment-it.com/en/hololens-2-ar-applications-for-more-efficiency-in-companies/>, 2020. Accessed: 2025-06-04.
- [9] I. Wright, “Bench-top coordinate measuring machine brings ar to qa.” <https://www.engineering.com/bench-top-coordinate-measuring-machine-brings-ar-to-qa/>, 2017. Accessed: 2025-06-04.
- [10] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.
- [11] M. T. Ibrahim, M. Gopi, and A. Majumder, “Projector-camera calibration on dynamic, deformable surfaces,” in *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 905–906, 2023.
- [12] G. Narita, Y. Watanabe, and M. Ishikawa, “Dynamic projection mapping onto deforming non-rigid surface using deformable dot cluster marker,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1235–1248, 2017.

- [13] H. Lim, D. Kim, G. Shin, J. Shi, I. Vizzo, H. Myung, J. Park, and L. Carlone, “Kiss-matcher: Fast and robust point cloud registration revisited,” *arXiv preprint arXiv:2409.15615*, 2024.
- [14] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, “Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1530–1538, 2017.
- [15] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [16] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 834–849, Springer International Publishing, 2014.
- [17] R. Muñoz-Salinas and R. Medina-Carnicer, “Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers,” *Pattern Recognition*, p. 107193, 2020.
- [18] J. Vanherck, L. Jorissen, B. Zoomers, and N. Michiels, “Projector-camera calibration with non-overlapping fields of view using a planar mirror,” *Virtual Reality*, vol. 29, p. 19, Jan 2025.
- [19] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.
- [20] G. Falcao, N. Hurtos, and J. Massich, “Plane-based calibration of a projector-camera system,” *VIBOT master*, vol. 9, no. 1, pp. 1–12, 2008.
- [21] OpenAI, “Chatgpt,” June 2025. AI-tool gebruikt ter ondersteuning van grammaticale controle en tekstuele helderheid.