Woord vooraf

Deze masterproef vormt de afsluiting van mijn opleiding Master in de Industriële Wetenschappen Informatica aan de UHasselt en KU Leuven. Tijdens deze opleiding nam mijn interesse voor Visual Computing sterk toe. Deze masterproef bood mij de kans om die interesse verder te verdiepen en de opgedane kennis toe te passen binnen een concreet onderzoeksproject. Hierdoor kon ik niet alleen nieuwe technologieën ontdekken, maar ook een tastbaar en bruikbaar eindresultaat realiseren.

Mijn oprechte dank gaat uit naar mijn promotor, Prof. dr. Nick Michiels, en mijn begeleider Joni Vanherck. Ik wil hen bedanken voor hun begeleiding, constructieve feedback en voortdurende ondersteuning gedurende het volledige traject, alsook voor het beschikbaar stellen van de infrastructuur en middelen die nodig waren om dit onderzoek uit te voeren.

Ten slotte ook dank aan iedereen binnen het DFL voor hun waardevolle adviezen en ondersteuning. Hun bereidheid om mee te denken en hun expertise te delen werd steeds gewaardeerd.

Inhoudsopgave

Lijst van tabellen

Lijst van figuren

Verklarende woordenlijst

Term / afkorting	**Omschrijving**
6DoF	Zes vrijheidsgraden (*six degrees of freedom*): drie translatiecomponenten (x, y, z) en drie rotatiecomponenten (pan, tilt, roll) die de pose van een camera volledig beschrijven.
AHRS	*Attitude and Heading Reference System*: inertieel systeem dat de absolute oriëntatie en heading van een object bepaalt door accelerometer-, gyroscoop- en magnetometermetingen te combineren.
Allan-variantie	Statistische methode om ruis en drift in IMU-sensoren te karakteriseren over verschillende tijdsschalen; levert parameters op voor de ruismodellering in een ESKF.
APE	*Absolute Pose Error*: maat voor de absolute positieafwijking van een geschatte pose ten opzichte van een referentiepose, berekend over alle punten in een traject.
AWB	*Automatic White Balance*: automatische witbalansinstelling van een camera op basis van de geschatte kleurtemperatuur van de lichtbron.
BGR	*Blue-Green-Red*: kleurruimte die intern door OpenCV gebruikt wordt, waarbij kleurkanalen in omgekeerde volgorde staan ten opzichte van de gebruikelijke RGB-volgorde.
Bias	Systematische afwijking of constante fout in een sensoruitvoer; bij een gyroscoop manifesteert bias zich als een constante hoekdrift wanneer het systeem stilstaat.
Binair masker	Afbeelding met uitsluitend zwarte en witte pixels die aanduidt welke beeldgebieden als mogelijke markers worden beschouwd na drempelstelling of morfologische bewerking.
Blob	Samenhangende cluster van witte pixels in een binair beeld, gebruikt als kandidaatdetectie voor een retroreflectieve marker.
Broadcast	Uitzenden van audio- of videosignalen; in de context van virtual production ook het versturen van trackingdata naar studioapparatuur.
Bundle adjustment	Gezamenlijke optimalisatie van 3D-puntposities en cameraposes op basis van reprojectiefouten over meerdere beelden, waarbij alle parameters tegelijk worden bijgewerkt.

Term / afkorting	**Omschrijving**
Centroïde	Gewogen middelpunt van een gedetecteerde blob, bepaald via intensiteitsgewogen beeldmomenten voor subpixelnauwkeurigheid.
Confidence	Kwaliteitsmaat (0–1) voor een geschatte camerapose, samengesteld uit de verhouding tussen inliers en detecties en de reprojectiefout.
CPU-affinity	Koppeling van een softwarethread aan een specifieke processorkern, zodat die taak steeds op dezelfde kern blijft draaien.
CSI	*Camera Serial Interface*: seriële verbindingsstandaard voor cameramodules op de Raspberry Pi via een flexibele lintkabel.
Daemon	Achtergrondproces dat zonder directe gebruikersinteractie permanent blijft draaien en de tijdskritische verwerking beheert.
Dead-reckoning	Het schatten van de huidige positie en oriëntatie op basis van eerder gemeten beweging via IMU-integratie, zonder externe absolute correctiemeting.
Delaunay-triangulatie	Geometrisch algoritme dat een verzameling punten groepeert in niet-overlappende driehoeken, waarbij geen enkel punt binnen de omgeschreven cirkel van een driehoek valt.
ESKF	*Error-State Kalman Filter*: Kalman-variant die de niet-lineaire toestand (inclusief oriëntatiequaternion) direct integreert en een kleine fouttoestand rond die nominale toestand schat; geschikt voor IMU-camera-fusie.
Exposure	Belichtingstijd van de camerasensor: de duur waarbinnen het licht wordt verzameld voor één frame.
Feature detection	Detectie van herkenbare beeldkenmerken zoals hoekpunten, blobstructuren of textuurbeschrijvingen.
FPS	*Frames Per Second*: aantal beelden per seconde dat een camera opneemt of een systeem verwerkt.
FreeD D1	Specifiek FreeD-pakkettype dat pan, tilt, roll en positie als 24-bit big-endian integers codeert en via UDP verstuurd wordt.
Genlock	Synchronisatietechniek die meerdere video-apparaten op hetzelfde tijdsraster koppelt, zodat beeldwissels exact gelijktijdig verlopen.
Green screen	Effen groene achtergrond die in nabewerking chromatisch wordt vervangen door een digitale achtergrond.
HAT	*Hardware Attached on Top*: uitbreidingsprint voor de Raspberry Pi die bovenop de basisprint gemonteerd wordt en extra functionaliteit aanbiedt (bv. M.2 NVMe HAT+ voor SSD-opslag).
HSV	*Hue, Saturation, Value*: kleurruimte waarbij tint, verzadiging en helderheid als afzonderlijke kanalen worden voorgesteld; het V-kanaal dient als helderheidskanaal bij de sterdetector.
I2C	*Inter-Integrated Circuit*: seriële communicatiebus voor communicatie over korte afstand tussen componenten op dezelfde print via een kloklijn en een datalijn.
IMU	*Inertial Measurement Unit*: sensor die lineaire versnelling (accelerometer) en hoeksnelheid (gyroscoop) meet; soms uitgebreid met een magnetometer.

Term / afkorting	**Omschrijving**
Inner frustum	Het zichtbare beeldgebied van de opnamecamera dat door de nDisplay-plugin op de LED-wall als hoge-kwaliteitsgebied gerenderd wordt.
IPC	*Inter-Process Communication*: mechanismen waarmee afzonderlijke processen gegevens en commando's uitwisselen, zoals shared memory en Unix domain sockets.
IR	Infrarood: elektromagnetische straling met golflengten net boven het zichtbare spectrum (typisch 780–1000 nm), gebruikt voor markerbelichting.
ISP	*Image Signal Processor*: speciale processor in de Raspberry Pi-camerachain die ruwe sensorbeelden voorverwerkt.
JSON	*JavaScript Object Notation*: lichtgewicht, tekstgebaseerd dataformaat voor gestructureerde gegevensuitwisseling.
LED-wall	Grote wand van LED-panelen waarop realtime digitale achtergronden worden weergegeven zodat acteurs en camera direct in de virtuele omgeving worden geplaatst.
Lens encoder	Sensor op een cameralens die de instellingen van focusring, zoomring en diafragmastand omzet naar digitale waarden voor gebruik in de renderpipeline.
LiDAR	*Light Detection and Ranging*: meetsysteem dat met laserpulsen de afstand tot de omgeving bepaalt en daaruit een driedimensionale puntenwolk opbouwt.
Lookup table	Vooraf berekende tabel voor snelle opzoeking van een waarde; in dit prototype gebruikt voor het opslaan van sterconstellatie-descriptoren.
Motion capture	Techniek waarbij bewegingen van acteurs, props of andere objecten nauwkeurig worden vastgelegd met externe trackingcamera's voor digitale reproductie.
nDisplay	Unreal Engine-plugin voor het aansturen van meerdere schermen of LED-walls in virtual-productie-opstellingen.
Nearest-neighbour matching	Koppelingsmethode waarbij elk gereprojecteerd kaartpunt wordt gekoppeld aan de dichtstbijzijnde detectie binnen een ingestelde maximale pixelafstand.
NN	*Nearest Neighbour*: koppelingsmethode waarbij elk gereprojecteerd kaartpunt wordt gekoppeld aan de dichtstbijzijnde detectie binnen een ingestelde maximale pixelafstand.
NVMe	*Non-Volatile Memory Express*: snel opslagprotocol voor solid-state schijven via de PCI Express-bus.
OpenCV	*Open Source Computer Vision Library*: open-source bibliotheek voor beeldverwerkings- en computervisiealgoritmen.
OpenVR	Open API voor VR-systemen, door SteamVR gebruikt voor communicatie tussen VR-hardware en toepassingen.

Term / afkorting	Omschrijving
One Euro Filter	Snelheidsafhankelijke laagdoorlaatfilter die jitter bij trage bewegingen onderdrukt en bij snellere bewegingen de afsnijfrequentie verhoogt om extra vertraging te beperken.
Parallax	Schijnbare positieverandering van een object ten opzichte van de achtergrond als gevolg van een verandering in kijkrichting of positie van de camera.
PnP	*Perspective-n-Point*: wiskundig probleem waarbij de pose van een camera wordt bepaald uit minstens vier correspondenties tussen 3D-objectpunten en hun 2D-projecties in het beeld.
PnP-RANSAC	Combinatie van PnP-poseberekening en RANSAC waarbij herhaaldelijk kleine subsets worden gebruikt om een posehypothese te genereren en te valideren op de volledige correspondentieset.
Producer-consumer patroon	Softwarearchitectuurpatroon waarbij één component (producer) data aanmaakt en een andere component (consumer) die data verwerkt, via een gedeelde buffer.
Qt/QML	*Qt Quick Markup Language*: declaratief UI-framework voor het bouwen van grafische gebruikersinterfaces in Qt-toepassingen; gebruikt voor de frontend van het prototype.
Quaternion	Wiskundige viervector die een rotatie in de 3D-ruimte voorstelt zonder singulariteiten zoals gimbal lock.
RANSAC	*Random Sample Consensus*: robuust schattingsalgoritme dat herhaaldelijk een minimale subset kiest om een model te hypothetiseren en dat model vervolgens toetst aan alle data om inliers te identificeren.
Rig	Mechanische opstelling waarop de opnamecamera, trackingmodule en eventuele andere apparatuur worden gemonteerd.
RMSE	*Root Mean Square Error*: vierkantswortel van het gemiddelde van de kwadratische afwijkingen; een veelgebruikte samenvattende nauwkeurigheidsmaat.
RPE	*Relative Pose Error*: maat voor de lokale trajectconsistentie over een vaste afgelegde afstand; minder gevoelig voor globale offset dan APE.
Seqlock	Lichtgewicht synchronisatiemechanisme voor gedeeld geheugen: de schrijver verhoogt een sequentieteller voor en na het schrijven; de lezer herhaalt het lezen zolang de teller tijdens het lezen verandert.
Shared memory	Gedeeld geheugengebied dat door meerdere processen tegelijkertijd gelezen of geschreven kan worden; gebruikt voor snelle overdracht van posedata en previewbeelden.
SLAM	*Simultaneous Localisation and Mapping*: klasse van algoritmen die tegelijk een kaart van de omgeving opbouwen en de positie van de sensor daarin schatten, zonder vooraf bekende referentiepunten.
SSD	*Solid State Drive*: opslagmedium zonder bewegende onderdelen, gebaseerd op flashgeheugen.

Term / afkorting	**Omschrijving**
SSH	*Secure Shell*: versleuteld netwerkprotocol voor beveiligde toegang op afstand tot een computer via de opdrachtregel.
SteamVR	VR-platform van Valve dat basisstations, controllers en trackers van het Lighthouse-systeem aanstuurt via de OpenVR API.
Tripletdescriptor	Geometrische descriptor gebaseerd op drie naburige markers waarbij afstandsverhoudingen en een hoekterm als kenmerken worden gebruikt voor schaal- en rotatie-invariante identificatie.
UDP	*User Datagram Protocol*: verbindingsloos netwerkprotocol voor het versturen van datagrampaketten met lage latency, zonder bevestiging van ontvangst.
UML	*Unified Modeling Language*: gestandaardiseerde modelleertaal voor het beschrijven van softwarearchitecturen en -ontwerpen.
UNC	*Unified National Coarse*: standaard voor Amerikaans schroefdraad; de 1/4"-20 UNC-maat is de industriestandaard voor cameramontages.
Undistortie	Correctie van lensdistortie in een camerabeeld op basis van de intrinsieke kalibratieparameters en voorberekende remap-matrices.
Unix domain socket	Lokaal IPC-mechanisme dat communicatie tussen processen op dezelfde machine via een pad in het bestandssysteem mogelijk maakt, zonder gebruik van een netwerkinterface.
Unreal Engine	Realtime render- en productie-engine van Epic Games; in dit prototype gebruikt als renderomgeving voor virtual production.
Virtual production	Productieworkflow waarbij fysieke filmopnames worden gecombineerd met realtime computergraphics, typisch via een LED-wall.
Vive Tracker	Draagbare bewegingsvolger van HTC voor het SteamVR-Lighthouse-systeem; in dit onderzoek gebruikt als onafhankelijk referentiesysteem voor de posevalidatie.
Workerthread	Softwarethread die één specifieke taak uitvoert, in dit prototype vast gekoppeld aan een processorkern via CPU-affinity.
XR	*Extended Reality*: overkoepelende term voor virtual reality (VR), augmented reality (AR) en mixed reality (MR).
YUV420	Kleur- en helderheidscoderingsformaat waarbij het helderheidskanaal (Y) op volle resolutie bewaard wordt en de kleurkanalen (U, V) op halve resolutie; het standaarduitvoerformaat van `rpicam-vid` in dit prototype.

Abstract

Virtual production vereist een nauwkeurige koppeling tussen fysieke en virtuele camera. Commerciële trackingsystemen zijn financieel ontoegankelijk voor kleinere producties en onderwijscontexten. Deze masterproef onderzoekt of een kostenefficiënt markergebaseerd cameratrackingsysteem de kernfunctionaliteit kan benaderen met betaalbare hardware en opensource software.

Een prototype werd gebouwd met een Raspberry Pi 5, infraroodgevoelige camera, retroreflectieve plafondmarkers en een IMU. De softwarepipeline berekent de camerapose en verstuurt die via FreeD naar Unreal Engine. Het systeem werd geëvalueerd in een LED-wallopstelling met een Vive Tracker als referentie.

Het prototype levert circa 98 Hz gefilterde pose-uitvoer. De sterkaart heeft een RMSE van 2,7 mm over 108 markers. Bij stilstand blijft de positiejitter (p95) onder 0,2 mm en de oriëntatiestabiliteit onder 0,01°. Tijdens beweging bedraagt de gemiddelde positieafwijking 1,5 cm (p95: 2,7 cm) en de yawafwijking gemiddeld 0,75° ten opzichte van de Vive Tracker.

Het prototype bevat de kernfunctionaliteit voor een bruikbare virtual-production-opstelling: inside-out tracking met passieve markers, gefilterde realtime pose-uitvoer en visueel stabiele werking op een LED-wall, rechtstreeks inzetbaar in een bestaande renderomgeving via FreeD, voor een hardwarekost van € 313. Beperkingen ten opzichte van commerciële systemen, waaronder ontbrekende *genlock* en lenscodering, blijven bestaan, maar de bevindingen bieden perspectief voor kleinschalige producties, onderwijs en onderzoek.

Abstract in English

Virtual production requires precise synchronization between a physical camera and its virtual counterpart in a real-time rendering environment. Commercial tracking systems provide high performance but are financially inaccessible to smaller productions and educational contexts. This thesis investigates whether a cost-effective marker-based tracker can achieve comparable functionality using affordable hardware and open-source software.

A prototype was built using a Raspberry Pi 5, an infrared-sensitive camera, retroreflective ceiling markers, and an IMU. The software pipeline estimates the camera pose and transmits it to Unreal Engine via the FreeD protocol. The system was evaluated in an LED-wall setup using an HTC Vive Tracker as reference.

The prototype delivers approximately 98 Hz filtered pose output. The star map has an RMSE of 2.7 mm across 108 markers. When stationary, positional jitter (p95) remains below 0.2 mm and orientation stability within 0.01°. During motion, the mean positional deviation from the Vive Tracker is 1.5 cm (p95: 2.7 cm) and mean yaw deviation 0.75°.

The prototype contains the core functionality needed for a viable virtual production setup: inside-out tracking with passive markers, filtered real-time pose output, and visually stable operation on an LED wall, directly deployable in an existing render environment via FreeD, at a hardware cost of € 313. Limitations including absent genlock and lens encoding remain, but the findings offer potential for small-scale productions, education, and research.

Hoofdstuk 1

Inleiding

1.1 Situering

Betrouwbare cameratracking is een noodzakelijke schakel in *virtual production*. Zonder nauwkeurige koppeling tussen fysieke en virtuele camera ontstaat er een zichtbaar verschil tussen de beweging van de camera op de set en de gerenderde omgeving. Deze masterproef onderzoekt daarom hoe een kostenefficiënt trackingsysteem kan worden opgebouwd dat bruikbaar is binnen een kleinschalige virtual-production-opstelling.

1.1.1 Virtual production

Virtual production combineert fysieke filmopnames met realtime computergraphics. In plaats van acteurs of presentatoren voor een klassiek *green screen* te filmen, kan de virtuele omgeving rechtstreeks op een LED-wall of in een gelijkaardige studio-opstelling worden weergegeven. De camera filmt daardoor niet alleen de persoon of het decor op de set, maar ook een reeds zichtbaar deel van de digitale omgeving. Dat maakt het mogelijk om licht, reflecties, compositie en perspectief tijdens de opname beter te beoordelen.

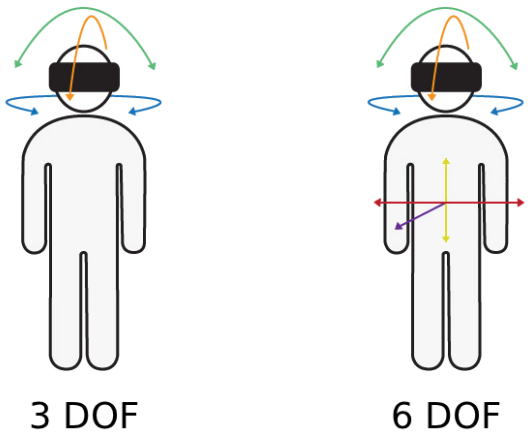
Een belangrijk verschil met traditionele nabewerking is dat de digitale omgeving tijdens de opname mee verandert met de camerabeweging. Wanneer de fysieke camera naar links beweegt, inzoomt of roteert, moet de virtuele camera in de *rendering engine* dezelfde beweging uitvoeren. Alleen dan blijft het perspectief van de LED-wall of de virtuele achtergrond consistent met het perspectief van de fysieke camera. Als die koppeling niet nauwkeurig genoeg is, ontstaan fouten zoals schuivende achtergronden, foutieve parallax of een zichtbaar verschil tussen fysieke en virtuele objecten.

De technische kern van een virtual-production-opstelling bestaat daarom uit een combinatie van cameratracking, realtime *rendering* en synchronisatie met de opnameketen. De rendering engine, bijvoorbeeld Unreal Engine, heeft niet alleen beeldinformatie nodig, maar ook een continue stroom van cameraparameters. Die parameters beschrijven onder meer de positie, oriëntatie en lensinstellingen van de fysieke camera. De kwaliteit van de virtuele productie wordt daardoor rechtstreeks beïnvloed door de nauwkeurigheid, stabiliteit en vertraging van het trackingsysteem. Figuur 1.1 toont een voorbeeld van een dergelijke opstelling.

Figuur 1.1: Voorbeeld van een virtual-production-opstelling waarin een LED-wall, een fysieke camera en een virtuele omgeving samen één opnameketen vormen. Bron: [1].

1.1.2 Cameratracking

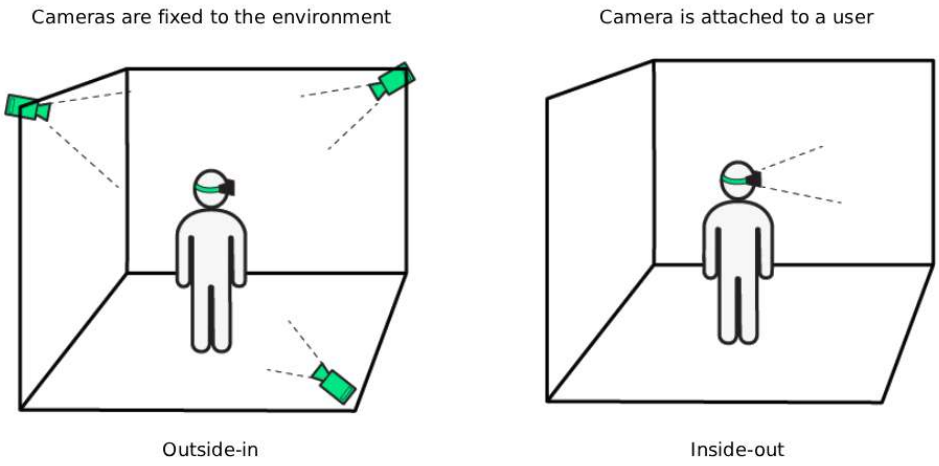
Een cameratrackingsysteem bepaalt de pose van een camera: de positie en oriëntatie van de camera in een gekozen wereldcoördinatenstelsel. Voor virtual production gaat het typisch om zes vrijheidsgraden of *six degrees of freedom* (6DoF): drie translatiecomponenten en drie rotatiecomponenten zoals te zien in figuur 1.2. Die pose moet bovendien met voldoende hoge frequentie en lage vertraging beschikbaar zijn, zodat de virtuele camera vloeiend en synchroon met de fysieke camera beweegt.



Figuur 1.2: Zes vrijheidsgraden van een trackingsysteem: drie translatiecomponenten bepalen de positie en drie rotatiecomponenten bepalen de oriëntatie. Bron: [2].

Trackingsystemen kunnen op verschillende principes gebaseerd zijn. Bij *outside-in tracking* observeren externe camera's of sensoren de positie van de filmcamera of van markers op de camera. Bij *inside-out* tracking bevindt de sensor zich op de camera zelf en kijkt die naar vaste kenmerken

in de omgeving. In beide gevallen moet het systeem herkenbare informatie uit de fysieke ruimte omzetten naar een camerapose die bruikbaar is voor de rendering engine.



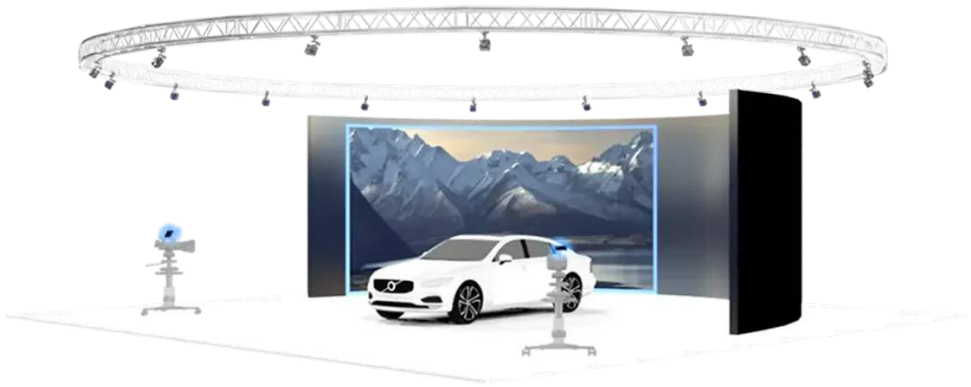
Figuur 1.3: Verschil tussen inside-out en outside-in tracking. Bij inside-out tracking bevindt de sensor zich op de camera en observeert hij vaste referenties in de omgeving; bij outside-in tracking observeren externe sensoren het te volgen object. Bron: [2].

Die herkenbare informatie kan kunstmatig aangebracht zijn, bijvoorbeeld via markers, of natuurlijk aanwezig zijn in de scène, bijvoorbeeld via textuur, randen en vaste objecten. Markertracking maakt de omgeving expliciet meetbaar en reproduceerbaar, terwijl markerloze tracking minder fysieke infrastructuur vraagt maar sterker afhankelijk is van de zichtbare scène. De keuze tussen deze principes bepaalt hoeveel infrastructuur nodig is, hoe voorspelbaar de tracking is en hoe goed het systeem past binnen een studio-opstelling.

1.1.3 Commerciële oplossingen

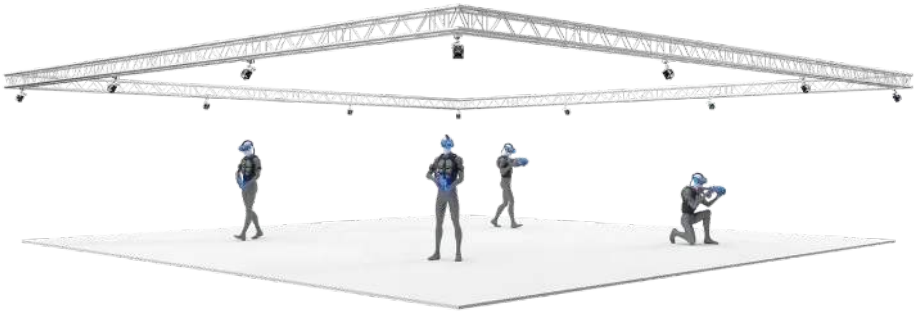
Professionele virtual-productionstudio's gebruiken commerciële trackingsystemen die specifiek ontwikkeld zijn voor camerawerk, *broadcast* en XR-productie. Zulke systemen leveren niet alleen positie- en oriëntatiegegevens, maar moeten ook passen binnen de volledige productieketen. Ze moeten betrouwbaar werken op een set, compatibel zijn met rendering engines, lensdata kunnen verwerken of doorgeven en trackingdata met lage *latency* beschikbaar maken.

Een eerste benadering is outside-in tracking. Daarbij volgen meerdere vaste camera's een object of rig in de studio. Vicon- en OptiTrack-opstellingen behoren tot deze categorie en gebruiken doorgaans infraroodcamera's in combinatie met passieve of actieve markers. De vaste camera's worden rond het opnamegebied geplaatst en moeten onderling gekalibreerd worden, zodat de positie van markers binnen dat gebied kan worden berekend. Figuur 1.4 toont een voorbeeld van zo'n trackingopstelling.

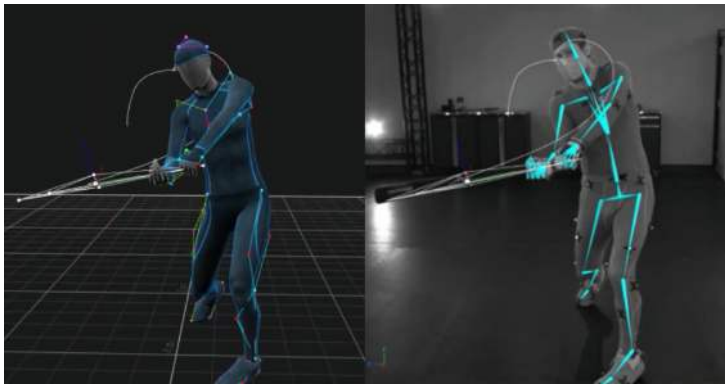


Figuur 1.4: OptiTrack outside-in trackingopstelling voor virtual production met zestien vaste trackingcamera's rond het opnamevolume. Bron: [3].

Doordat het volledige trackingvolume extern wordt waargenomen, kan dezelfde infrastructuur breder worden ingezet dan alleen voor cameratracking binnen virtual production. Ook in andere domeinen, zoals de game-industrie, wordt outside-in tracking gebruikt voor toepassingen zoals *motion capture*. Daarbij worden de bewegingen van acteurs, props of andere objecten nauwkeurig vastgelegd, zodat deze bewegingen digitaal gereproduceerd of verder verwerkt kunnen worden. Figuur 1.6 illustreert zo'n toepassing met een trackingpak [4, 5].



Figuur 1.5: OptiTrack outside-in trackingopstelling voor motion capture in VR met twaalf trackingcamera's.. Bron: [3].



Figuur 1.6: Voorbeeld van motion capture met een trackingpak waarbij lichaamsbewegingen met externe trackingcamera's worden geregistreerd en digitaal gereconstrueerd. Bron: [6].

Die brede inzetbaarheid maakt van outside-in tracking een zeer geschikte oplossing, maar maakt de opstelling tegelijk duur en minder flexibel. De voorbeelden in figuur 1.4 en figuur 1.5 tonen dat een praktijkgerichte configuratie snel uit veel camera's bestaat: de virtual-production-opstelling gebruikt zestien camera's en vertegenwoordigt een kost van meer dan € 60.000, terwijl de VR-opstelling met twaalf camera's al meer dan € 45.000 kost. In beide gevallen gaat het enkel om de trackingcamera's en nodige hardware, dus nog zonder ondersteunende infrastructuur, softwarelicenties of de nodige computer. Daardoor blijven ze vooral geschikt voor professionele producties waarin de kost, voorbereidingstijd en vaste infrastructuur verantwoord zijn.

Een tweede benadering is inside-out tracking. Daarbij draagt de camera zelf een trackingcamera die vaste referenties in de omgeving observeert. Systemen zoals Mo-Sys StarTracker (Fig. 1.7), Stype RedSpy en Antilatency gebruiken hiervoor eveneens infraroodmarkerprincipes, maar verschillen in de keuze tussen passieve retroreflectieve markers en actieve markerinfrastructuur [7, 8, 9, 10]. Deze klasse sluit beter aan bij een compact prototype, omdat de vaste infrastructuur beperkt kan blijven en de trackingmodule op de camera zelf gemonteerd wordt.

Die beperktere infrastructuur betekent echter niet automatisch dat zulke systemen ook goedkoop zijn. Ook de compacte trackingkoppen in figuur 1.8 blijven commerciële systemen met een hoge instapkost: een opstelling met de Mo-Sys StarTracker Max kost meer dan € 28.000, terwijl een Stype RedSpy 5.0 Pro-broadcastpakket boven € 38.000 uitkomt [11, 12]. Inside-out tracking vermindert dus vooral de vaste setinfrastructuur, maar lost het kostenprobleem van professionele cameratracking niet vanzelf op.

Figuur 1.7: Mo-Sys StarTracker gemonteerd op een studiocamera; de trackingkop kijkt omhoog naar retroreflectieve sterren op het plafond en gebruikt die als vaste referentie voor inside-out cameratracking. Bron: [13].

(a) Mo-Sys StarTracker

(b) Stype RedSpy

(c) Antilatency

Figuur 1.8: Detailfoto's van drie commerciële trackingkoppen die de sensorunit op of dicht bij de camera plaatsen: Mo-Sys StarTracker, Stype RedSpy en Antilatency. Bronnen: [14, 15, 16].

Naast markergebaseerde systemen bestaan ook markerloze oplossingen. Een gebruikelijke techniek daarvoor is SLAM, voluit *simultaneous localisation and mapping*. Daarbij bouwt een camera of sensor tijdens het bewegen een kaart van herkenbare beeldkenmerken in de omgeving op en bepaalt het systeem tegelijk zijn eigen beweging ten opzichte van die kaart. Het voordeel is dat er geen nood is aan een vooraf ingerichte trackinginfrastructuur of markers die geplaatst moeten worden.

In een studio- of LED-wallomgeving is dat voordeel echter niet altijd doorslaggevend. Een SLAM-systeem heeft voldoende belichting en stabiele herkenningspunten nodig om betrouwbaar te kunnen werken. In een donkere studio, een egaal ingerichte set of een omgeving met weinig vaste details kan het systeem daardoor te weinig bruikbare kenmerken vinden om een pose te bepalen. Voor klassieke markerloze tracking zijn dergelijke omgevingen dus niet optimaal.

Een LED-wall opent wel een interessante bijkomende mogelijkheid. Doordat de weergegeven beelden volledig controleerbaar zijn, kunnen visuele kenmerken of patronen doelbewust in die content geïntegreerd worden. De LED-wall zou dan zelf kunnen functioneren als een bron van trackbare kenmerken. Dit is technisch interessant, maar vereist verder onderzoek en valt daarom buiten de klassieke SLAM-toepassing die hier als markerloos alternatief wordt besproken.

Tabel 1.1 geeft een vergelijkend overzicht van de systeemklassen en hun relevantie als uitgangspunt voor deze masterproef.

Tabel 1.1: Algemene vergelijking van camera-trackingsystemen voor virtual production, met per systeemklasse de benodigde infrastructuur en de relevantie voor een kostenefficiënt prototype.

Systeemklasse	**Algemene eigenschappen**	**Relevantie voor deze masterproef**
Outside-in tracking	Meerdere externe camera's volgen markers of objecten in een gekalibreerd volume.	Zeer geschikt als professionele referentie, maar minder passend als kostenefficiënt systeem.
Inside-out met passieve markers	De camera draagt zelf een trackingmodule en observeert vaste referenties in de omgeving.	Sluit het best aan bij een compacte tracker met retroreflectieve markers.
Markerloze systemen	De camera gebruikt natuurlijke kenmerken of andere sensordata zonder vaste markers.	Gevoelig voor te weinig stabiele beeldkenmerken.

1.2 Probleemstelling

De voorgaande vergelijking toont dat cameratracking binnen virtual production technisch volwassen is, maar voor kleine producties, onderwijsinstellingen en onderzoeksprojecten blijven de kosten van zowel de tracker als ondersteunende infrastructuur te hoog.

Een educatieve of experimentele opstelling moet niet noodzakelijk dezelfde prestaties halen als een professioneel studiosysteem, maar moet wel aantonen of de kernfunctionaliteit haalbaar is: een fysieke camera volgen, de pose stabiel berekenen en die pose bruikbaar doorsturen naar een realtime renderomgeving. Er is dus behoefte aan een betaalbaar, modulair en reproduceerbaar trackingprototype dat de basisprincipes van commerciële cameratrackingsystemen onderzoekt.

De keuze voor een inside-out systeem met retroreflectieve markers volgt uit de balans tussen technische haalbaarheid, kostprijs en schaalbaarheid. Outside-in systemen bieden sterke prestaties, maar vragen een grotere setinfrastructuur. Markerloze systemen beperken de fysieke infrastructuur, maar zijn afhankelijker van licht en zichtbare omgevingskenmerken. Een inside-out markertracker biedt daarom een bruikbaar middenpunt: de vaste infrastructuur blijft beperkt tot passieve markers, terwijl de cameratracker toch met expliciete referentiepunten kan werken.

De onderzoeksvraag luidt daarom: “Hoe kan een kostenefficiënt cameratrackingsysteem worden ontwikkeld dat de werking van commerciële inside-out startracker-systemen benadert, met behulp van vrij beschikbare hardware en software?”

1.3 Doelstellingen en aanpak

De hoofddoelstelling van deze masterproef is het ontwikkelen van een trackingsysteem als *proof of concept*, gebaseerd op inside-out tracking met passieve markers. Het beoogde resultaat is een prototype dat voldoende nauwkeurigheid en stabiliteit biedt voor realtime virtual production.

Deze hoofddoelstelling wordt vertaald naar de volgende concrete deeldoelstellingen:

1. **Analyse van bestaande systemen en onderliggende principes**

   In een eerste fase wordt de technische werking van gelijkaardige systemen onderzocht aan de hand van technische documentatie en relevante literatuur. Het doel hiervan is inzicht te verkrijgen in de fundamentele principes van optische cameratracking. Daarbij wordt nagegaan welke hardwarecomponenten, softwaremodules en verwerkingsstappen noodzakelijk zijn voor een dergelijk systeem.

2. **Ontwerp en realisatie van een optische trackingmodule**

   Een tweede doelstelling bestaat uit het ontwerpen en implementeren van een eigen optisch trackingsysteem. Hierbij wordt een hardware-opstelling ontwikkeld die compact, draagbaar en kostenefficiënt is, zodat ze praktisch inzetbaar blijft in een kleine virtual-productioncontext. Op softwarevlak omvat deze doelstelling de ontwikkeling van algoritmen voor markerherkenning en cameraposebepaling, aangevuld met IMU-data. Daarnaast moeten ook camerakalibratie, de opbouw van een referentiekaart van de markers en realtime localisatie en tracking gerealiseerd worden.

3. **Integratie met Unreal Engine**

   Het ontwikkelde trackingsysteem moet gekoppeld worden aan Unreal Engine, zodat de berekende positie- en oriëntatiegegevens bruikbaar worden binnen een realtime renderomgeving. De focus ligt hierbij op een stabiele en consistente synchronisatie tussen de fysieke camera-opstelling en de virtuele camera in de 3D-scène.

4. **Validatie in een kleinschalige LED-wall-opstelling**

   Ten slotte worden de prestaties van het systeem geëvalueerd in een kleinschalige virtual-production-opstelling met een LED-wall. Hierbij worden nauwkeurigheid, stabiliteit, detectiesnelheid en end-to-end latency onderzocht. De nauwkeurigheid van de pose tijdens beweging wordt bepaald aan de hand van een HTC Vive Tracker als onafhankelijke referentie. Het doel is niet om commerciële systemen volledig te evenaren, maar om aan te tonen dat een kostenefficiënte oplossing voldoende bruikbaar kan zijn voor experimentele of educatieve toepassingen in virtual production.

1.4 Vooruitblik

De aanpak bestaat uit een combinatie van systeemanalyse, prototypeontwikkeling en experimentele validatie. Na deze inleiding volgt daarom eerst de systeemanalyse. Daarin wordt de keuze voor een inside-out markertracker vertaald naar functionele eisen en naar de noodzakelijke technische onderdelen.

Vervolgens beschrijft het systeemontwerp hoe het prototype op hardware- en softwareniveau is opgebouwd. De nadruk ligt daarbij op de samenwerking tussen de trackingcamera, IMU, backend, *frontend* en uitvoer naar Unreal Engine. De algoritmische uitwerking gaat daarna dieper in op de verwerking van camerabeelden, de opbouw en het gebruik van de sterkaart, de localisatie- en trackingstappen, de poseberekening en de fusie met IMU-data.

Daarna komen de fysieke testopstelling en de evaluatieopzet aan bod. Eerst worden de gebruikte componenten en de virtual-productionomgeving toegelicht, waarna de methodiek voor detectie, localisatie, tracking, integratie en vergelijking met een commercieel systeem wordt beschreven. Tot slot worden de resultaten besproken, de beperkingen van het prototype geduid en aanbevelingen voor verdere ontwikkeling geformuleerd.

Hoofdstuk 2

Systeemanalyse

Dit hoofdstuk analyseert welke functionele onderdelen nodig zijn voor een kostenefficiënt cameratrackingsysteem voor virtual production. In de inleiding is de keuze voor een inside-out systeem met passieve markers gemotiveerd vanuit de vergelijking met andere commerciële en markerloze oplossingen. De systeemanalyse vertrekt daarom vanuit dat systeemtype en bespreekt welke technische eisen daar uit volgen. De concrete toestand van het prototype wordt later beschreven in het systeemontwerp en in het hoofdstuk over de algoritmische uitwerking.

2.1 Inside-out markergebaseerde tracking

Bij een inside-out markertracker draagt de productiecamerarig zelf de trackingcamera. Die camera observeert vaste referentiepunten in de omgeving en leidt daaruit haar pose af. Voor een virtual-production-opstelling zijn vooral retroreflectieve markers relevant, omdat ze passief, goedkoop en visueel sterk onderscheidbaar zijn onder infraroodbelichting. Deze sectie bespreekt daarom eerst welke eigenschappen commerciële inside-out systemen gemeenschappelijk hebben, en leidt daaruit vervolgens de functionele structuur voor de rest van dit hoofdstuk af.

2.1.1 Commerciële inside-out cameratrackingsystemen

Mo-Sys StarTracker, Stype RedSpy en Antilatency vormen de meest directe commerciële referentiesystemen voor deze analyse. Ze plaatsen de trackingmodule op of dicht bij de camera en gebruiken retroreflectieve *stars* die op plafond, vloer of wanden geplaatst worden. De productspecificatie beschrijft een optische trackingkop met infraroodbelichting, een brede beeldhoek en een netwerkuitvoer voor trackinggegevens. De markers vormen samen een constellatie die door het systeem als ruimtelijke referentie wordt gebruikt [7, 8, 9].

Uit de documentatie volgen enkele concrete systeemkenmerken. De Mo-Sys StarTracker gebruikt actieve infraroodbelichting rond de sensor, met 850 nm als vermelde golflengte in de productspecificatie. Het systeem werkt met retroreflectieve stickers en geeft aan dat een voldoende aantal goed gespreide markers zichtbaar moet zijn. De documentatie vermeldt minimaal 11 zichtbare sterren om tracking te behouden en raadt in de praktijk een grotere zichtbare set aan. De markers mogen daarbij niet in een regelmatig raster liggen, omdat een regelmatig patroon geometrische ambiguïteiten veroorzaakt: verschillende deelpatronen kunnen dan sterk op elkaar lijken. Ook de

hoogte tussen sensor en sterren wordt bevestigd in een beschreven kalibratiestap, wat bevestigt dat de markerinfrastructuur niet alleen visueel maar ook metrisch relevant is [7].

De handleiding toont daarnaast dat het systeem meer omvat dan beeldverwerking alleen. Er zijn stappen voor *exposure*-instelling, *star parameters*, *mapping*, referentie aan een wereldcoördinatenstelsel, camera-*offsets*, lenscodering en netwerkuitvoer [17]. In de gebruikersinterface wordt ook een IMU- of gyrofunctie vermeld.

Antilatency is relevant als compacte inside-out referentie, maar gebruikt geen passieve retroreflectieve stickers. De tracker observeert een actieve IR-markerinfrastructuur op plafond of vloer. De trackingkop weegt 12 g en combineert optische tracking met inertiële metingen; de optische frequentie wordt opgegeven als 60–400 fps en de inertiële frequentie als 2000 Hz [10, 18]. Qua integratie ondersteunt Antilatency het FreeD-protocol en biedt het *native plugins* voor Unreal Engine en andere VP-platformen. Dit maakt het systeem architectureel relevant voor een compacte tracker, ook al verschilt het markerprincipe van een passieve startracker.

Stype RedSpy is relevant omdat de documentatie explicieter beschrijft welke sensoren samen worden gebruikt. De StypeBook beschrijft RedSpy als een optisch cameratrackingsysteem dat trackinggegevens afleidt uit een infraroodcamera, een accelerometer en een gyroscoop [9]. De camera detecteert markers op de vloer of het plafond, de accelerometer helpt bij plotselinge bewegingen en de gyroscoop ondersteunt rotatieveranderingen. Dit bevestigt dat commerciële systemen optische absolute tracking combineren met inertiële metingen voor temporele stabiliteit en een hogere poseschattingsfrequentie.

Voor de systeemanalyse betekent dit dat een optische tracker niet als geïsoleerde beeldverwerkingsmodule beschouwd wordt. De visuele meting levert een absolute referentie zonder drift, maar heeft een beperkte beeldfrequentie en kan tijdelijk verstoord worden door occlusie of detectiefouten. Een IMU heeft daarentegen een hoge updatefrequentie, maar integreert ruis en *bias*. De combinatie van beide sensortypes is daarom een terugkerend patroon in systemen die lage latency en stabiele pose-uitvoer vereisen.

2.1.2 Hardware en basisspecificaties per systeem

Elk commercieel systeem vereist naast de trackingmodule ook aanvullende hardware om operationeel te zijn. Tabel 2.1 combineert daarom de markerinfrastructuur, de benodigde hardwaremodules en de gepubliceerde nauwkeurigheids- of resolutiespecificaties. De prijsinformatie wordt pas in de kostenanalyse behandeld; hier is vooral relevant welke modules nodig zijn om tracking technisch mogelijk te maken en welke orde van nauwkeurigheid commerciële systemen bereiken.

Tabel 2.1: Markerprincipe, benodigde hardware en gepubliceerde nauwkeurigheids- of resolutiespecificaties per commercieel cameratrackingsysteem voor virtual production. De tabel vergelijkt enkel specificaties die door de besproken systemen publiek of technisch onderbouwd beschikbaar zijn.

Systeem	**Marker**	**Benodigde hardware**	**Nauwk./res.**
Mo-Sys StarTracker Max [11, 8]	Passief	Sensorunit met processorunit, voeding en monitor, retroreflectieve markers.	Positie: 0,01 mm Hoek: 0,001°
Mo-Sys StarTracker Mini [19, 7]	Passief	Sensorunit, voeding, retroreflectieve markers.	Positie: <1 mm Hoek: <0,1°
Antilatency Basic Kit [10, 18]	Actief	Antilatency-tracker met VP Socket, radio-ontvanger, actieve markerinfrastructuur.	Positie: <1 mm Hoek: <0,1°
Stype RedSpy 5.0 Pro [12, 15]	Passief	RedSpy-camera met *main unit*, voeding en monitor, retroreflectieve markers.	Positie: <0,1 mm Hoek: <0,003°
Stype RedSpy Air / Air Plus [20, 21]	Passief	RedSpy-camera met *controllerboard*, voeding, retroreflectieve markers.	Positie: <0,1 mm Hoek: <0,003°

2.1.3 Functionele opdeling van het systeem

Uit de referentiesystemen volgt dat een inside-out markertracker niet als één beeldverwerkingsalgoritme kan worden geanalyseerd. De commerciële systemen tonen telkens dezelfde functionele keten: de omgeving moet optisch bruikbare markers aanbieden, de camera moet die markers detecteren, de waarnemingen moeten aan een markerkaart gekoppeld worden en de daaruit berekende pose moet gekalibreerd, gestabiliseerd en doorgestuurd worden. Tabel 2.2 maakt die opdeling expliciet en vormt de structuur van dit hoofdstuk.

Tabel 2.2: Functionele opdeling afgeleid uit de commerciële referentiesystemen. Elke rij koppelt een terugkerend onderdeel uit de referentiesystemen aan een systeemfunctie die in de rest van de systeemanalyse verder wordt uitgewerkt.

Functionele laag	**Afleiding uit de referentiesystemen**
Markeromgeving en observatie	Mo-Sys en Stype gebruiken passieve retroreflectieve markers met IR-observatie; Antilatency toont dezelfde inside-out relatie met een actieve markeromgeving.
Kaartopbouw en identificatie	Mo-Sys vermeldt mapping, goed gespreide sterren en een niet-regelmatige markerplaatsing; de markers vormen dus eerst een bruikbare ruimtelijke kaart.
Pose-schatting	De systemen leveren 6DOF-positie en -oriëntatie uit markerwaarnemingen, wat een expliciete 2D-3D-geometrische stap vereist.
Kalibratie en coördinatenstelsels	Documentatie over wereldreferentie, sensorhoogte, camera-offsets, lensgegevens en markerconfiguratie toont dat de pose pas bruikbaar is na correcte kalibratie.
Sensorfusie	Stype en Antilatency combineren optische tracking met accelerometer- en gyroscoopmetingen; ook bij Mo-Sys wordt een IMU- of gyrofunctie vermeld.
Realtime uitvoer	Mo-Sys, Stype en Antilatency voorzien netwerk- of FreeD-uitvoer naar virtual-productionsoftware.

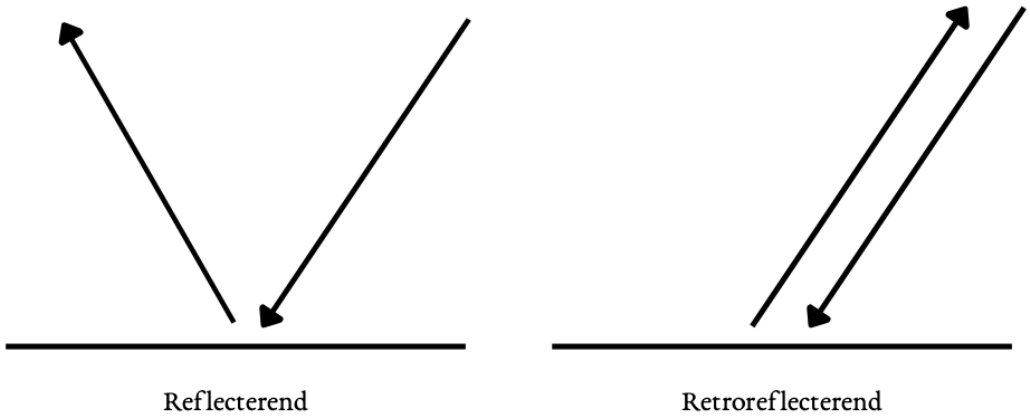
De volgende secties analyseren deze lagen afzonderlijk en leiden per laag de technische eisen af. Aan het einde van het hoofdstuk worden die eisen opnieuw samengevat in tabel 2.3.

2.2 Markerdetectie

Markerdetectie vormt de eerste kritische stap in de verwerkingsketen. Een fout in deze stap werkt rechtstreeks door in kaartopbouw, matching en pose-schatting. De taak is in dit systeem specifieker dan algemene *feature detection*: de relevante objecten zijn heldere, compacte reflecties van passieve retroreflectieve markers onder actieve infraroodbelichting.

2.2.1 Optische eigenschappen van retroreflectieve markers

Een gewone reflecterende marker kaatst licht afhankelijk van materiaal, oppervlak en kijkhoek terug naar de omgeving. Een retroreflectieve marker werkt anders: die stuurt een groot deel van het invallende licht terug in de richting van de lichtbron. In een opstelling waarbij infraroodbelichting dicht bij de trackingcamera zit, verschijnen zulke markers daarom als heldere lokale maxima in het camerabeeld. Figuur 2.1 toont dit principe. Dit is een voordeel ten opzichte van natuurlijke *features*: de detectie kan grotendeels intensiteitsgebaseerd gebeuren en hoeft geen complexe textuurdescriptoren te gebruiken.



Figuur 2.1: Principe van retroreflectieve markerdetectie met IR-belichting. De belichting zit dicht bij de trackingcamera, waardoor retroreflectieve markers licht terugkaatsen naar de sensor en als heldere punten in het beeld verschijnen.

De helderheid en vorm van een marker blijven afhankelijk van afstand, kijkhoek, belichting, lens, sensorrespons en achtergrondreflecties. Kirollos en Povey [22] tonen bijvoorbeeld dat ronde reflectieve markers onder een kijkhoek elliptisch kunnen worden en dat markerdiameter en beeldresolutie invloed hebben op de nauwkeurigheid van het centrum. Wang et al. [23] tonen bovendien dat een infraroodfilter voor de lens extra radiale distortie kan veroorzaken, waardoor het centrum van een marker in ruwe beeldcoördinaten niet zonder correctie als geometrisch zuiver punt mag worden beschouwd.

2.2.2 Snelle detectie via drempeling en contouren

De eenvoudigste detectieklasse vertrekt van intensiteitsdrempeling. IRMA [24] gebruikt een IR-filter, *thresholding*, morfologische bewerkingen, *connected components* en centroidberekening om

markers in meerdere camerabeelden te detecteren. Lin et al. [25] gebruiken eveneens binaire segmentatie en *blob filtering*, maar voegen expliciete vormcriteria toe zoals breedte-hoogteverhouding, oppervlakte en epipolaire consistentie in een stereosysteem. Xing et al. [26] beschrijven een real-time verwerkingsketen waarin na drempeling en connected components een Kalman-voorspelling wordt gebruikt om de zoekregio in volgende beelden te beperken.

Deze klasse van methoden is aantrekkelijk voor *embedded* verwerking omdat de rekenkost laag is. De keerzijde is dat de kwaliteit sterk afhangt van belichting en drempelkeuze. Daarom is een validatiefase noodzakelijk. Typische validatiecriteria zijn minimale en maximale oppervlakte, compactheid, circulariteit, intensiteitscentrum en temporele stabiliteit. Voor dit type tracking-systeem is een kleinere set betrouwbare detecties vaak bruikbaarder dan een grote set met veel valse reflecties.

2.2.3 Robuuste detectie bij variabele markerbeelden

Wanneer markers sterk in grootte verschillen of wanneer de achtergrond minder homogeen is, kan eenvoudige drempeling onvoldoende zijn. Poulopoulos en Psarakis [27] beschrijven een *blob detector* gebaseerd op een vereenvoudigde *Fast Radial Symmetry Transform*. Die methode zoekt radiale symmetrie over meerdere schalen en is daardoor geschikt voor heldere ronde structuren met variabele straal. Ou et al. [28] beschrijven een snelle cirkeldetectiemethode die cirkelinformatie comprimeert en kandidaatcirkels efficiënt verifieert. Zulke methoden zijn robuuster tegen schaalvariatie, maar vragen meer rekentijd dan pure segmentatie.

Voor een camera die veel beelden per seconde moet verwerken is de belangrijkste vergelijking dus niet alleen detectiekwaliteit, maar ook kost per beeld. Een globale schaalruimtedetectie kan nuttig zijn bij initialisatie of moeilijke beelden, terwijl een lichtere drempel- en contourmethode aantrekkelijker is wanneer de markerposities al ongeveer voorspelbaar zijn.

2.2.4 Detectievereisten

Uit de literatuur volgen vier detectievereisten. Ten eerste moet de methode helderheidsinformatie in camerabeelden benutten, maar niet uitsluitend afhankelijk zijn van een globale drempelwaarde. Ten tweede moet de uitvoer bestaan uit geometrisch bruikbare, bij voorkeur subpixelnauwkeurige centra. Ten derde moet de detector andere heldere structuren kunnen onderdrukken via vorm-, grootte- en stabiliteitscriteria. Ten vierde moeten de stickers zo uniform mogelijk zijn in vorm, zodat variatie in de detectie vooral uit projectie, belichting en lensvervorming komt en niet uit verschillen tussen markers zelf.

2.3 Geometrische identificatie en matching

De detectie van een marker resulteerd in een tweedimensionaal coordinaat op het camerabeeld, en is het dus nog niet bekend waar deze zich bevindt in de trackingruimte. Omdat retroreflectieve stickers visueel uniform zijn, kan hun identiteit niet uit individuele eigenschappen worden afgeleid. Het herkennen van sterpunten moet daarom gebeuren op basis van geometrische relaties binnen de constellatie.

2.3.1 Kaartopbouw zonder voorkennis

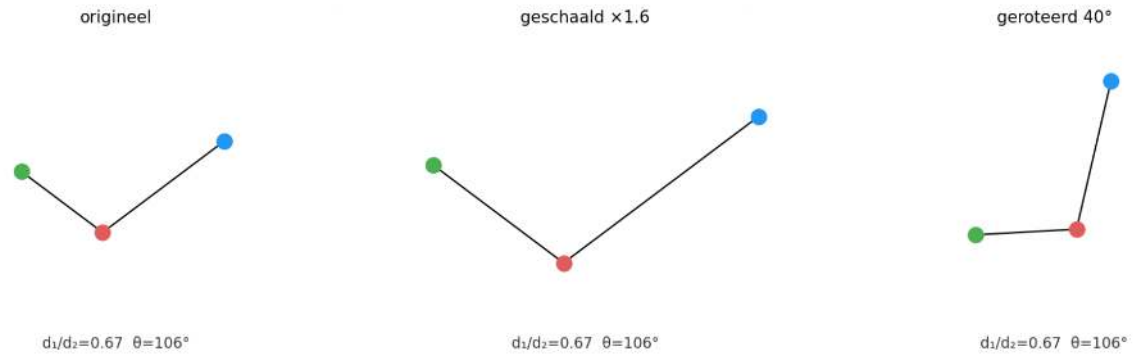
Voordat globale localisatie of lokale matching mogelijk is, moet een sterrenkaart opgebouwd worden. Die kaart moet metrisch correct zijn en de volledige constellatie zo accuraat mogelijk beschrijven, zonder dat de sterposities of cameraposes vooraf gekend zijn. De kaartopbouw vertrekt dus alleen van meerdere overlappende camerabeelden waarin telkens een deel van dezelfde markerstructuur zichtbaar is.

Dit maakt kaartopbouw een gezamenlijk schattingsprobleem. Het systeem moet bepalen welke gedetecteerde punten er in verschillende beelden naar dezelfde fysieke marker verwijzen, terwijl tegelijk de relatieve cameraposes tussen die beelden onbekend zijn. De geometrie van overlappende observaties moet daarom gebruikt worden om zowel de markerposities als de bijbehorende camerastandpunten te verfijnen. Methoden zoals *bundle adjustment* zijn hiervoor relevant, omdat zij 3D-punten en cameraposes gezamenlijk optimaliseren op basis van reprojectiefouten [29, 30].

De nauwkeurigheid van deze kaart is bepalend voor alle latere poses. Een fout in de sterrenkaart werkt systematisch door: de posemodule kan dan een lage reprojectiefout vinden ten opzichte van een verkeerde kaart, terwijl de werkelijke camera in de ruimte toch fout gepositioneerd wordt. De opgebouwde sterrenkaart moet daarom een metrisch consistente referentie zijn, met correcte schaal en een duidelijk wereldcoördinatenstelsel.

2.3.2 Globale localisatie vanuit onbekende pose

Bij globale localisatie is er nog geen voorgaande pose beschikbaar. Het systeem moet dan uit de actuele set 2D-detecties en een reeds bekende sterrenkaart kandidaatidentificaties afleiden. Eenvoudige afstandsverhoudingen en hoeken tussen puntgroepen zijn relatief makkelijk te berekenen en kunnen translatie, rotatie en schaalverandering opvangen. Figuur 2.2 illustreert waarom zulke eigenschappen nuttig zijn: dezelfde lokale constellatie blijft herkenbaar wanneer ze verschoven, gedraaid of geschaald wordt. Ze zijn echter minder robuust tegen perspectiefvervorming wanneer de camera sterk helt ten opzichte van het plafondvlak.



Figuur 2.2: Schaal- en rotatie-invariante eigenschappen van lokale sterconstellaties. De relatieve afstanden en hoeken tussen naburige markers blijven bruikbaar wanneer dezelfde constellatie onder een andere schaal of rotatie in beeld komt.

Kinectrack gebruikt een aanpak uit astronomische *star trackers*: gedetecteerde punten worden via Delaunay-triangulatie gegroepeerd, waarna perspectief-invariante *kite invariants* in een *lookup table* worden opgezocht [30]. Transparent Random Dot Markers gebruikt LLAH en spectrale matching om anonieme willekeurige punten ook bij gedeeltelijke overlap te identificeren [31]. Beide bronnen zijn relevant omdat ze hetzelfde fundamentele probleem behandelen: veel visueel gelijkaardige punten moeten via hun onderlinge geometrie herkend worden.

2.3.3 Lokale matching tijdens tracking

Wanneer er wel een vorige pose beschikbaar is, verandert het probleem. Kaartpunten kunnen dan vanuit de vorige pose naar het nieuwe beeld worden geprojecteerd. Matching kan vervolgens lokaal gebeuren door de dichtstbijzijnde detectie binnen een beperkte zoekradius te kiezen. Figuur 2.3 toont dat principe: de voorspelde projectie bepaalt waar een kaartpunt ongeveer verwacht wordt, waarna alleen detecties in de nabije omgeving kandidaat zijn. Zeynalov et al. [32] gebruiken voor het tracken van infraroodmarkers *nearest-neighbour*-achtige correspondenties tussen opeenvolgende beelden met afstandsdrempels en herstellen verloren correspondenties wanneer markers opnieuw verschijnen. Dit type methode is niet voldoende voor initialisatie, maar is zeer geschikt voor snelle tracking van beeld tot beeld.

Lokale matching kan ook de detectiestap beperken. Wu et al. [33] beschrijven een verwerkingsketen waarbij de eerste detectie over het volledige beeld gebeurt en latere detectie in kleinere regio's rond verwachte markerposities wordt uitgevoerd. Voor een real-time trackingsysteem is dat onderscheid belangrijk: globale detectie en matching mogen zwaarder zijn bij initialisatie, terwijl continue tracking vooral de verwachte markerprojecties rond de vorige pose moet controleren. De zoekradius in figuur 2.3 is daardoor niet alleen een matchingcriterium, maar ook een manier om de beeldverwerking tot relevante regio's te beperken.

Figuur 2.3: Lokale nearest-neighbour matching tussen de gereprojecteerde kaartpunten van de vorige pose en de actuele detecties. Alleen detecties binnen de zoekradius worden als kandidaat beschouwd, waardoor tracking sneller is dan globale identificatie.

De analyse leidt dus tot een duidelijk onderscheid tussen twee matchingmodi. Globale matching moet robuust zijn tegen onbekende pose en gedeeltelijke markerzichtbaarheid. Lokale matching moet snel zijn en gebruikmaken van temporele continuïteit. Een praktisch systeem heeft beide nodig, omdat tracking altijd opnieuw kan wegvallen door occlusie, bewegingsonscherpte of onvoldoende zichtbare markers.

2.3.4 Robuuste validatie

Matching levert enkel kandidaatidentificaties, geen garantie op correcte koppelingen. Foutieve detecties, ontbrekende markers en geometrisch gelijkaardige lokale patronen kunnen foutieve matches veroorzaken. Daarom moet matching gevolgd worden door geometrische validatie. RANSAC-achtige consensus, reprojectiefout en minimumaantal *inliers* zijn hiervoor logische kwaliteitscriteria. In stereo-opstellingen worden ook epipolaire voorwaarden gebruikt om foutieve infraroodmarkerparen te verwerpen [25, 32]. In een inside-out tracker met één camera ligt de nadruk op consistentie met een homografie- of PnP-model.

2.3.5 Matchingvereisten

De identificatie- en matchingmodule moet globale herlocalisatie ondersteunen op basis van geometrische constellaties, lokale tracking op basis van projectievoorspelling en robuuste validatie van kandidaatsets. De descriptoren moeten tolerant zijn voor ontbrekende of vals positieve markers en perspectiefvervorming, terwijl de rekenkost laag genoeg blijft voor embedded verwerking.

2.4 Pose-schatting

Pose-schatting zet gevalideerde 2D-3D-correspondenties om in de positie en oriëntatie van de trackingcamera. In virtual production is deze stap kritisch: geometrisch kleine fouten kunnen zichtbaar worden als verschuiving tussen de fysieke en virtuele camera. De kwaliteit van de pose hangt af van drie zaken: de geometrie van de markerkaart, de betrouwbaarheid van de correspondenties en de gekozen posemethode.

2.4.1 Planariteit van plafondmarkers

In deze toepassing liggen de markers op één vlak: het plafond. Dat heeft gevolgen voor de keuze van de posemethode. Een homografie kan de projectieve relatie tussen een vlakke markerkaart en het beeld beschrijven, maar levert de pose pas na decompositie en is gevoelig voor ambiguïteit en ruis. IPPE is specifiek ontwikkeld voor pose-schatting uit coplanaire punten en behandelt de planariteit rechtstreeks. Collins en Bartoli [34] tonen dat een planaire solver de structuur van het probleem beter benut dan algemene PnP-methoden die bij coplanaire punten numeriek ongunstig kunnen worden.

Algemene PnP-methoden blijven relevant als vergelijkingspunt, omdat ze het algemene 3D-2D-probleem behandelen. EPnP, SQPnP, RPnP, OPnP en recente varianten zoals DEPnP maken verschillende afwegingen tussen snelheid, nauwkeurigheid en robuustheid [35, 36, 37, 38]. Voor een plafondgebaseerde markerkaart volgt uit deze analyse echter dat de posemodule expliciet met coplanaire punten moet kunnen omgaan.

2.4.2 Verschil tussen localisatiepose en trackingpose

Het onderscheid tussen localisatie en tracking gaat verder dan de manier waarop correspondenties worden gevonden. Bij globale localisatie moet de posemodule meerdere posehypothesen kunnen evalueren, omdat de initiële correspondenties onzeker zijn. De pose wordt dan gebruikt als bevestiging van een kandidaatmatch: alleen een hypothese met voldoende inliers, lage reprojectiefout en plausibele geometrie wordt aanvaard.

Tijdens continue tracking is de poseberekening anders opgebouwd. Er is een vorige pose, de beweging tussen twee beelden is beperkt en de pose kan iteratief rond de vorige oplossing verfijnd worden. De posemodule hoeft dan niet elke mogelijke globale hypothese te testen, maar moet wel snel en stabiel reageren op kleine beeldveranderingen. De solverkeuze, initialisatie en validatiecriteria zijn daardoor anders dan bij globale localisatie.

2.4.3 Kwaliteitsmaten

Een pose-oplossing is pas bruikbaar wanneer haar betrouwbaarheid gekend is. De reprojectiefout is daarbij de meest directe maat: kaartpunten worden met de geschatte pose terug naar het beeld geprojecteerd en vergeleken met de gedetecteerde centra. De fout is dus de gemiddelde pixelafstand tussen wat de pose voorspelt en wat de camera werkelijk detecteert. Een lage reprojectiefout betekent dat detecties, correspondenties, markerkaart en posemodel geometrisch goed bij elkaar passen.

Een hoge reprojectiefout betekent daarentegen dat geen enkele geschatte camerapose alle observaties tegelijk goed verklaart. De PnP-oplosser levert dan een compromis, waardoor de pose zichtbaar kan verschuiven. Bij een grotere fout is het eveneens mogelijk dat meerdere oplossingen ongeveer dezelfde afwijking hebben. Reprojectiefouten van minder dan 1 pixel zijn daarom wenselijk: ze tonen dat de pose niet alleen numeriek gevonden is, maar ook nauw aansluit bij de waargenomen markerposities. Grotere drempels, bijvoorbeeld in de orde van meerdere pixels, zijn vooral nuttig als rejectiegrens en niet als kwaliteitsdoel.

Daarnaast zijn het aantal inliers en de ruimtelijke spreiding van die inliers belangrijk. Inliers zijn de correspondenties die na validatie consistent blijven met dezelfde posehypothese. Een grote hoeveelheid markers in een smalle beeldregio kan minder informatief zijn dan minder markers die goed over het beeld verdeeld zijn. De posemodule moet deze kwaliteitsinformatie daarom doorgeven aan de filter- of sensorfusielaag, zodat twijfelachtige visuele metingen gedempt of verworpen kunnen worden.

2.4.4 Posevereisten

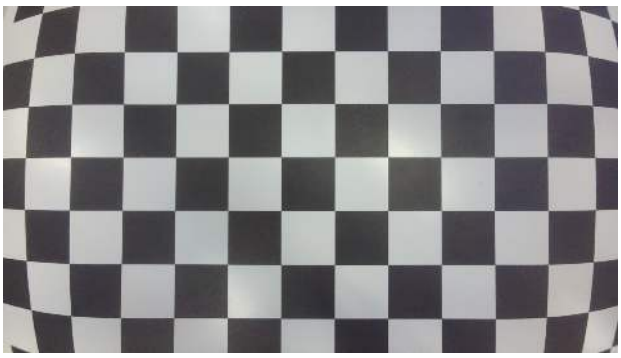
Uit de analyse volgen drie eisen. De posemodule moet coplanaire markerconfiguraties correct kunnen behandelen, meerdere hypothesen kunnen beoordelen bij globale localisatie en een snelle pose-update kunnen uitvoeren tijdens continue tracking. Daarnaast moet elke oplossing kwaliteitsmaten leveren, zoals reprojectiefout, aantal inliers en spreiding van de gebruikte markers.

2.5 Camerakalibratie en coördinatenstelsels

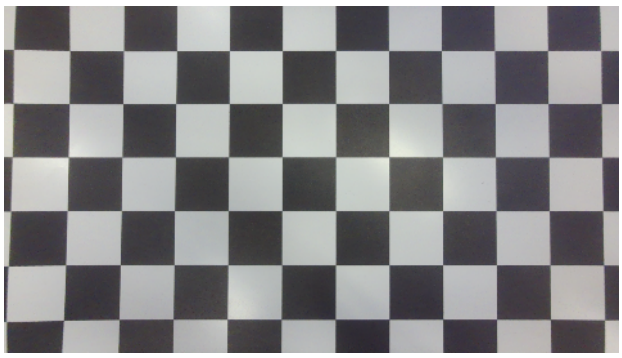
Camerakalibratie is een randvoorwaarde voor alle geometrische stappen. Zonder gekende intrinsieke parameters en distortie kan een gedetecteerd beeldpunt niet betrouwbaar als projectiestraal worden geïnterpreteerd. Zonder duidelijke coördinatenstelsels kan een correcte trackingpose bovendien niet eenduidig naar een renderomgeving gestuurd worden.

2.5.1 Intrinsieke kalibratie en infraroodlensdistortie

De intrinsieke kalibratie beschrijft brandpuntsafstand, hoofdpunt en lensdistortie. Een brede beeldhoek is nuttig omdat er dan meer plafondmarkers tegelijk zichtbaar zijn, maar brede lenzen vergroten vaak de radiale vervorming aan de beeldranden. Zie figuur 2.4. Voor markertracking is die distortie geen cosmetisch probleem: ze verplaatst de gemeten markercentra en beïnvloedt matching en pose-schatting.



(a) Ruw camerabeeld met zichtbare radiale vervorming aan de beeldranden.



(b) Gecorrigeerd beeld na toepassing van de intrinsieke kalibratieparameters.

Figuur 2.4: Checkerboardafbeelding voor (links) en na (rechts) correctie van de radiale lensdistortie; de rechte rasterlijnen in het gecorrigeerde beeld bevestigen de werking van de intrinsieke camerakalibratie.

Bij infraroodgebaseerde systemen komt daar een extra aandachtspunt bij. Wang et al. [23] tonen dat een infraroodfilter voor de lens bijkomende radiale vervorming kan introduceren en beschrijven daarom een kalibratieprocedure in deze condities. Dit betekent dat een kalibratie met zichtbaar licht niet automatisch volstaat wanneer de uiteindelijke tracking met een infraroodfilter en infraroodbelichting gebeurt.

2.5.2 Coördinatenstelsels en sensoroffsets

Voor een cameratrackingsysteem zijn er minstens drie relevante coördinatenstelsels: het camerakader van de trackingcamera, het wereldkader van de markerkaart en het kader van de opnamecamera of virtuele camera. Wanneer de trackingcamera niet exact samenvalt met de opnamecamera, is ook een rig-*offset* nodig. De analyse van commerciële systemen toont dat zulke offsets en lensgegevens expliciet deel uitmaken van het kalibratieproces [17, 39].

Ook de IMU heeft een eigen coördinatenstelsel. Om inertiële voorspellingen met visuele poses te combineren moet de rotatie en translatie tussen IMU en trackingcamera gekend zijn. Daarnaast

moet de tijdsrelatie tussen beide sensoren duidelijk zijn, omdat een kleine tijdsverschuiving bij snelle bewegingen tot een verkeerde vergelijking tussen voorspelde en visuele pose leidt.

2.5.3 Kalibratievereisten

De kalibratielaag moet de intrinsieke camera-parameters, lensdistortie, markerkaart, wereldreferentie, rig-offsets en IMU-camera-extrinsieken van elkaar onderscheiden. Elk van deze parameters heeft een andere functie en een andere foutimpact. De kalibratie moet eenmalig zeer nauwkeurig gebeuren voor de gekozen opstelling, omdat systematische fouten in kalibratie niet door latere filtering verdwijnen.

2.6 Sensorfusie

2.6.1 Rol van IMU-data

De Stype-documentatie beschrijft een commercieel systeem waarin een infraroodcamera, een accelerometer en een gyroscoop samen worden gebruikt [9]. Voor een inside-out markertracker is de IMU ook nuttig als voorspellende sensor. Op basis van recente beweging kan het systeem een verwachte pose en daaruit verwachte markerprojecties bepalen. Die projecties verkleinen de zoekruimte voor detectie en matching tijdens continue tracking.

Die rol is vooral praktisch bij snelle camerabewegingen en bij een beperkte camerafrequentie. De visuele verwerking hoeft dan niet telkens het volledige beeld als onafhankelijk probleem te behandelen, maar kan rond voorspelde markerposities zoeken. Xing et al. [26] tonen voor infraroodmarkertracking dat een Kalman-voorspelling zulke zoekregio's kan beperken en daardoor de verwerking versnelt. Daarnaast verhoogt IMU-integratie de frequentie waarmee een pose aan de renderomgeving kan worden aangeboden. De optische pose blijft de absolute referentie, maar komt slechts binnen aan de camerafrequentie. Tussen twee camerabeelden kan de IMU de pose vooruit voorspellen, zodat de virtuele camera vaker bijgewerkt wordt en de gerenderde beweging vloeiender blijft. Agistri [40] beschrijft in een andere context hetzelfde basisprincipe op systeemniveau: gyroscoopinformatie ondersteunt de dynamische voorspelling, terwijl *star tracker*-metingen de absolute correctie leveren.

2.6.2 Filterarchitecturen

Voor IMU-camera-integratie is een niet-lineaire filter nodig, omdat rotaties, acceleraties en biascorrectie niet lineair zijn. Een *Error-State Kalman Filter* is hiervoor een gangbare architectuur. De nominale toestand wordt met IMU-metingen vooruit voorspeld. Die toestand bevat minstens de oriëntatie en eventueel ook positie, snelheid en sensorbias. De filter schat daarnaast een kleine fout op die nominale toestand.

Wanneer een absolute visuele pose beschikbaar is, wordt die niet zomaar als nieuwe toestand overgenomen. De visuele pose wordt gebruikt als correctiemeting: het verschil tussen voorspelde en gemeten pose corrigeert de fouttoestand, waarna de nominale toestand wordt bijgewerkt. Daarvoor moeten de tijdstempels van camera en IMU, de assenconventies en de extrinsieke relatie tussen IMU en trackingcamera consistent zijn. Als die voorwaarden niet kloppen, vergelijkt de

filter fysisch verschillende grootheden en kan de uitvoer numeriek stabiel lijken maar geometrisch fout zijn.

In de analysefase is vooral de architecturale implicatie belangrijk. De optische verwerkingsketen moet tijdstempels, kwaliteitsmaten en absolute posemetingen leveren die bruikbaar zijn als correctiemetingen. De IMU-verwerkingsketen moet frequent genoeg meten, bias kunnen modelleren en een consistente conventie voor oriëntatie en assen gebruiken.

2.6.3 Fusievereisten

Sensorfusie vereist consistente tijdstempels, gekende assenconventies, een gekende extrinsieke relatie tussen IMU en trackingcamera, en kwaliteitsmaten op de visuele pose. Daarnaast is een afzonderlijk ruisonderzoek van de IMU nodig om gyroscoopruis, accelerometerruis, biasstabiliteit en driftgedrag te karakteriseren. Die parameters bepalen de procesruis en biasmodellen van de filter. Ook een camera-IMU-kalibratie is noodzakelijk, omdat de rotatie, translatie en eventuele tijdsvertraging tussen beide sensoren bepalen hoe inertiële voorspellingen met visuele posemetingen vergeleken kunnen worden.

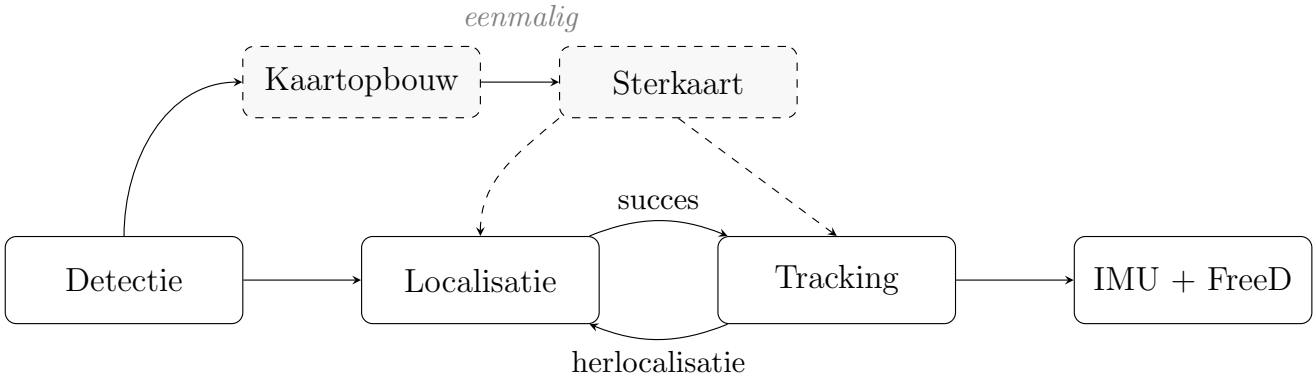
2.7 Realtime integratie

De laatste systeemlaag is de overdracht naar de renderomgeving. In virtual production is poseinformatie pas bruikbaar wanneer zij met voldoende lage latency, juiste assenconventie en consistente schaal bij de virtuele camera aankomt. Kadner beschrijft FreeD als een gangbaar protocol voor trackinggegevens zoals *pan*, *tilt*, *roll*, positie, *zoom* en focus [39]. Ook commerciële systemen zoals Mo-Sys en Stype bieden netwerkuitvoer voor integratie met render- en broadcastsoftware [17, 9].

Voor de systeemanalyse volgen hieruit drie eisen. De uitvoer moet een vast updatetempo kunnen halen dat past bij de beeldfrequentie van de renderomgeving. De conversie tussen trackingcoördinaten en engine-coördinaten moet expliciet gedefinieerd zijn. Ten slotte moet de communicatielaag licht genoeg zijn om de beeldverwerking en pose-schatting niet te blokkeren. Communicatie is dus geen aparte nabewerking, maar een real-time onderdeel van de totale trackingketen.

2.8 Conclusie

De systeemanalyse toont dat een inside-out markergebaseerde cameratracker uit meerdere gekoppelde deelproblemen bestaat. De commerciële referentiesystemen tonen de noodzakelijke systeemlagen: infraroodmarkerobservatie, kaartopbouw, identificatie en matching, pose-schatting, kalibratie, sensorfusie en realtime uitvoer. De literatuur vult deze lagen technisch in met detectiemethoden voor heldere infraroodmarkers, geometrische identificatie van anonieme puntconstellaties, planaire pose-schatting en filterarchitecturen voor visueel-inertiële stabilisatie. De belangrijkste ontwerpimplicatie is dat het systeem twee verwerkingsmodi nodig heeft: globale localisatie wanneer geen vorige pose beschikbaar is, en een lichtere continue trackingmodus daarna. Figuur 2.5 geeft een schematisch overzicht van de verwerkingsketen; tabel 2.3 werkt de afgeleide eisen per domein verder uit.



Figuur 2.5: Schematisch overzicht van de verwerkingsketen. Kaartopbouw gebruikt detectie om een sterkaart op te bouwen. Tijdens verwerking wisselt het systeem tussen localisatie en tracking op basis van de beschikbare posekwaliteit. De geschatte pose wordt gecombineerd met sensordata en via FreeD verzonden.

Tabel 2.3: Samenvatting van de systeemeisen voor een kostenefficiënte inside-out markertracker. De eisen volgen uit de commerciële referentiesystemen en uit de literatuur rond detectie, kaartopbouw, pose-schatting, kalibratie en sensorfusie.

Domein	**Afgeleide systeemeis**
Markerinfrastructuur	De markers moeten goedkoop, passief, mechanisch stabiel, uniform van vorm en voldoende zichtbaar zijn. Hun plaatsing moet onregelmatig maar goed gespreid zijn, zodat lokale constellaties herkenbaar blijven.
Optische observatie	De sensor moet voldoende markers tegelijk waarnemen. Een brede beeldhoek is nuttig, maar vereist expliciete correctie van lensdistortie. IR-belichting en optische filtering moeten markerreflecties met hoog contrast opleveren.
Detectie	De beeldverwerking moet heldere compacte IR-reflecties robuust detecteren, valse reflecties verwerpen en geometrisch bruikbare markercentra leveren.
Kaartopbouw	Het systeem heeft een accurate metrische markerkaart nodig die wordt opgebouwd zonder vooraf gekende sterposities of cameraposes. Kaartfouten moeten vermeden worden, omdat ze systematisch doorwerken in elke pose.
Identificatie en matching	Het systeem moet globale herlocalisatie uit anonieme markerconstellaties en snelle lokale matching tijdens continue tracking ondersteunen. Beide modi moeten gevalideerd worden met geometrische kwaliteitsmaten.
Pose-schatting	De posemodule moet met coplanaire plafondmarkers kunnen omgaan, meerdere hypothesen kunnen beoordelen bij initialisatie en snelle pose-updates kunnen leveren tijdens tracking.
Kalibratie	Intrinsieke camera-parameters, IR-distortie, wereldreferentie, rig-offsets en IMU-camera-extrinsieken moeten nauwkeurig bepaald worden voor de gekozen opstelling.
Sensorfusie	De architectuur moet visuele posemetingen kunnen combineren met IMU-voorspellingen. Daarvoor zijn tijdstempels, assenconventies, IMU-ruiskarakterisatie en camera-IMU-kalibratie nodig.
Real-time integratie	De pose-uitvoer moet met lage latency, correcte schaal en expliciete assenconversie beschikbaar zijn voor een renderomgeving, bijvoorbeeld via FreeD.

Hoofdstuk 3

Systeemontwerp

Dit hoofdstuk beschrijft het ontwerp van het ontwikkelde prototype. De systeemanalyse toont dat een inside-out markertracker niet alleen afhankelijk is van een correct softwarematig algoritme. Het algoritme moet ondersteund worden door geschikte hardware, een stabiele mechanische opbouw en een efficiënte softwarearchitectuur.

Het systeemontwerp legt daarom vast hoe de trackingmodule fysiek en softwarematig is opgebouwd. Eerst wordt het hardwareconcept besproken, omdat de keuze van rekeneenheid, camera, IMU, infraroodbelichting en mechanische montage de verdere architectuur bepaalt. Daarna volgt de softwarearchitectuur met de procesgrenzen, de IPC-laag en de taakverdeling over de processorkernen. De exacte werking van de software wordt in het volgende hoofdstuk behandeld.

3.1 Hardwareconcept

De hardware is ontworpen als een compacte trackingmodule die vast op de camerarig wordt gemonteerd. Dit bootst het formaat, gewicht en werking van commerciële cameratrackers na. Figuur 3.1 toont het finale prototype van boven- en onderzijde.

Figuur 3.1: Finaal prototype van de trackingmodule. Links is de bovenzijde met trackingcamera en IR-ledring zichtbaar; rechts is de onderzijde met IMU en bevestiging zichtbaar.

3.1.1 Raspberry Pi 5 als rekeneenheid

De centrale rekeneenheid is een Raspberry Pi 5. Die verwerkt de camerabeelden lokaal, leest de IMU uit, draait de grafische interface en verstuurt de pose naar de renderomgeving. De keuze voor lokale verwerking past bij de doelstelling van een reproduceerbaar en kostenefficiënt prototype: de trackingmodule heeft geen externe trackingcomputer nodig om de basiswerking uit te voeren.

De Raspberry Pi 5 wordt gecombineerd met een koeler-HAT voor thermische stabiliteit en optioneel een SSD voor extra opslag. De koeling is nodig omdat beeldverwerking en poseberekening de processor langdurig belasten. Zonder actieve of voldoende efficiënte koeling kan de rekeneenheid thermisch terugklokken, wat rechtstreeks invloed heeft op de beeldfrequentie en latency. Figuur 3.2 toont de Raspberry Pi 5 met de actieve koeler en M.2 HAT+.

Figuur 3.2: Raspberry Pi 5 met actieve koeler voor thermische stabiliteit (links) en M.2 HAT+ voor SSD-opslag (rechts). Bron: [41, 42].

3.1.2 Trackingcamera en lens

De opwaarts gerichte trackingcamera is een IMX708 *wide/noir*-sensor. De brede lens is functioneel belangrijk omdat er in één beeld voldoende markers zichtbaar moeten zijn om localisatie en tracking mogelijk te maken. Een smallere beeldhoek zou de detectie geometrisch nauwkeuriger kunnen maken, maar verkleint het zichtbare plafondgebied en verhoogt het risico dat te weinig markers zichtbaar zijn.

De opwaarts gerichte trackingcamera is een Raspberry Pi Camera Module 3 NoIR Wide met IMX708-sensor. Deze camera is gekozen binnen het kader van goedkope en gemakkelijk verkrijgbare hardware. Voor het prototype moest de camera een brede beeldhoek hebben, infrarood licht kunnen waarnemen en via een CSI-ribbonkabel rechtstreeks op de Raspberry Pi aangesloten kunnen worden. De NoIR-uitvoering bevat geen standaard infraroodfilter, waardoor infraroodreflecties van de markers zichtbaar blijven. De brede lens is functioneel belangrijk omdat er in één beeld voldoende markers zichtbaar moeten zijn om localisatie en tracking mogelijk te maken. Een smallere beeldhoek zou de detectie geometrisch nauwkeuriger kunnen maken, maar verkleint de zichtbare oppervlakte en daarom ook de grenzen van het tracking gebied.

De camerakeuze heeft ook een keerzijde. De brede lens introduceert merkbare lensdistortie, zeker aan de beeldranden. Daarom is de intrinsieke camerakalibratie een belangrijk onderdeel van het systeemontwerp. De *daemon* verwerkt de beelden in een ontdistorteerde beeldruimte, zodat de gedetecteerde markercentra geometrisch bruikbaar zijn voor matching en pose-schatting. Figuur 3.3 toont de gebruikte cameramodule.

Figuur 3.3: Raspberry Pi Camera Module 3 NoIR Wide met IMX708-sensor en brede lens. De brede beeldhoek vergroot het zichtbare plafondgebied en verhoogt de kans dat voldoende markers in één *frame* aanwezig zijn. Bron: [43].

3.1.3 infraroodbelichting en retroreflectieve markers

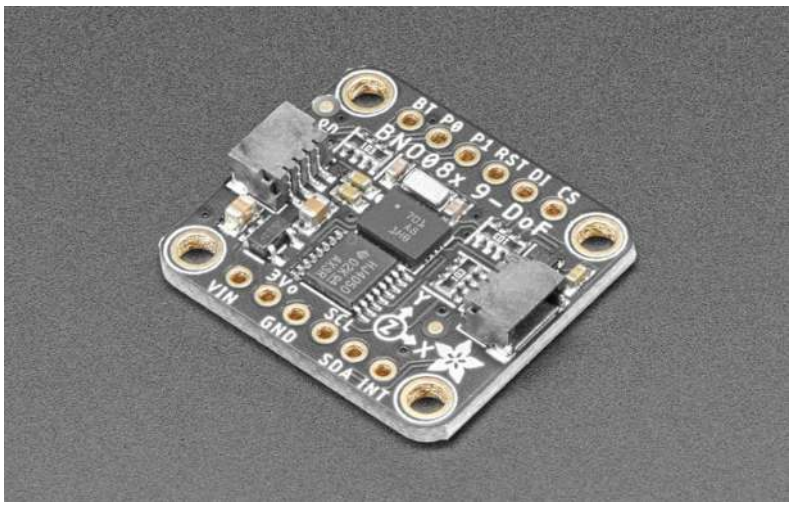
De trackingcamera wordt gecombineerd met infraroodbelichting rond de lens. Retroreflectieve stickers sturen het licht grotendeels terug naar de lichtbron en verschijnen daardoor als heldere punten in het camerabeeld. De infraroodbelichting moet fysiek dicht genoeg bij de camera zitten om retroreflectie goed te benutten. Tegelijk mag de belichting de sensor niet verzadigen. De positie van de leds, de camera-exposure, de *gain* en de detectiedrempels vormen daarom één samenhangende ontwerpkeuze. Figuur 3.4 toont de gebruikte IR-ledring.

Figuur 3.4: IR-ledring met infraroodleds, geplaatst rond de trackingcamera voor retroreflectieve belichting van de plafondmarkers.

3.1.4 IMU

De trackingmodule bevat een BNO085-IMU. Die levert gyroscoopgegevens, accelerometergegevens en oriëntatiegegevens via I2C. I2C is een seriële bus voor communicatie over korte afstand tussen componenten op dezelfde print of module. De Raspberry Pi fungeert daarbij als controller en leest de IMU via twee signaallijnen uit: een kloklijn en een datalijn. De IMU is vast gekoppeld aan de trackingcamera, zodat de relatieve oriëntatie tussen beide sensoren niet verandert tijdens gebruik. Die mechanische koppeling is belangrijk omdat de sensorfusie alleen betekenisvol is wanneer de IMU-camera-extrinsieken stabiel blijven.

De IMU vervangt de visuele pose niet. De visuele pose blijft de absolute referentie ten opzichte van de markerkaart. De IMU ondersteunt het systeem vooral door snellere bewegingen tussen twee camerabeelden te voorspellen en door een hogere updatefrequentie beschikbaar te maken voor de filterlaag. Figuur 3.5 toont het gebruikte IMU-breakout-board.



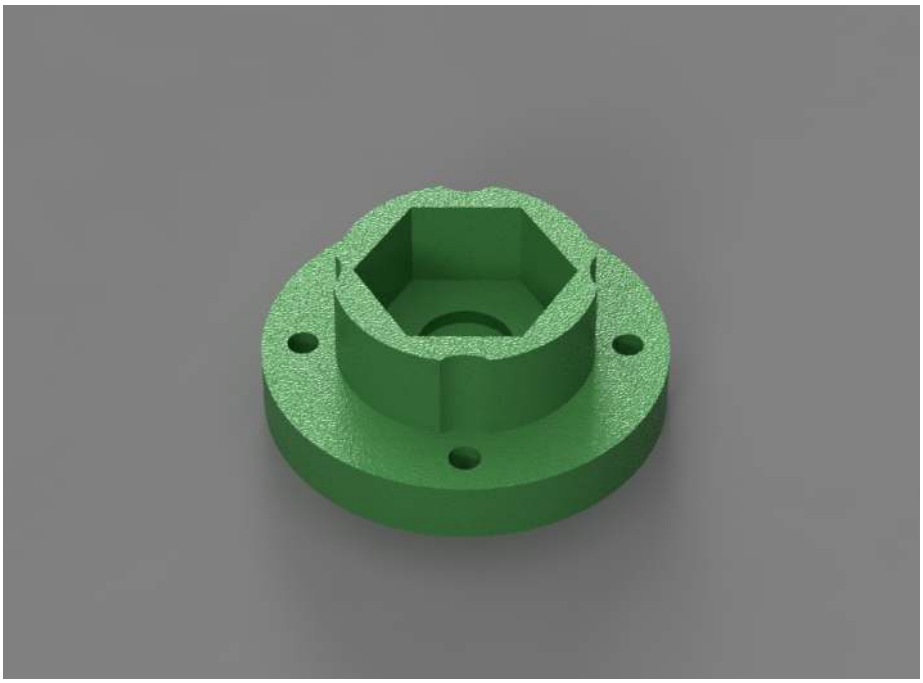
Figuur 3.5: Adafruit BNO085 9-DoF IMU breakout-board voor gyroscoop-, accelerometer- en oriëntatiemetingen via I2C. In het prototype wordt de IMU gebruikt als snelle inertiële sensor tussen twee visuele posemetingen. Bron: [44].

3.1.5 Behuizing, 3D-geprinte tussenstukken en touchscreen

Figuur 3.6 toont de KKSB metalen behuizing voor Raspberry Pi 5 en M.2 NVMe HAT [45], deze vormt de fysieke kern van de trackingmodule. Alle onderdelen worden, rechtstreeks of via tussenstukken, aan deze behuizing bevestigd. De ventilatieopeningen worden benut door het actieve koelingssysteem en alle nodige poorten zijn nog steeds beschikbaar.

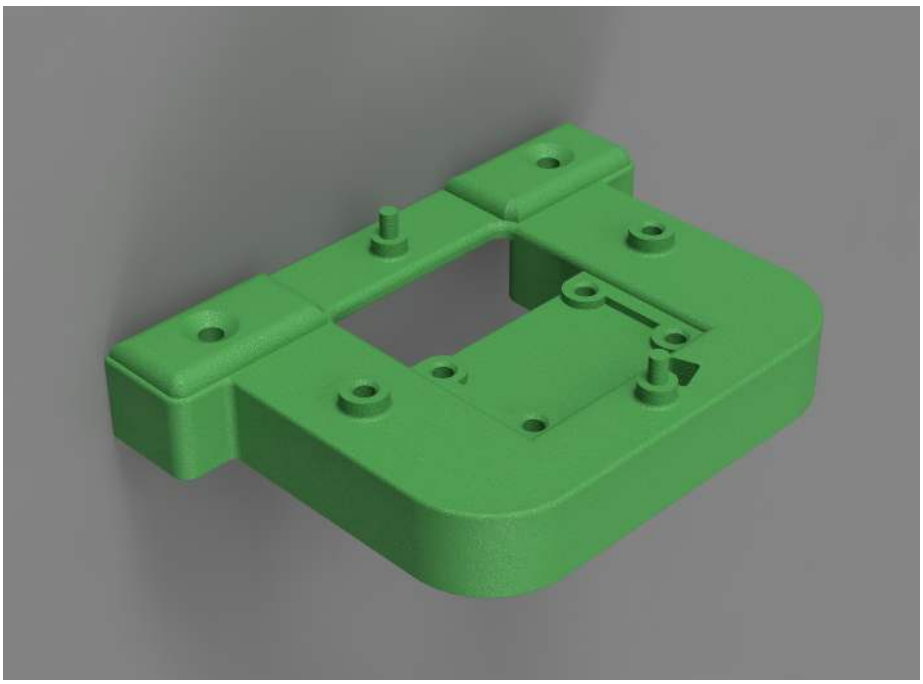
Figuur 3.6: KKSB metalen behuizing voor Raspberry Pi 5 en M.2 NVMe HAT+. De behuizing beschermt de rekeneenheid en vormt tegelijk het mechanische basispunt voor de 3D-geprinte bevestigingen. Bron: [45].

Om het prototype op een verstelbare arm of rechtstreeks op de camera te bevestigen, is een 3D-geprinte adapter ontworpen. Die beugel sluit aan op de onderkant van de metalen behuizing en biedt een opening om de nodige moer in te bevestigen die standaard 1/4”-20 UNC-schroefdraad heeft. Die maat is de industriestandaard voor cameramontages en maakt de trackingmodule compatibel met gangbaar fotografie- en filmmateriaal. Figuur 3.7 toont de 3D-geprinte console-beugel.

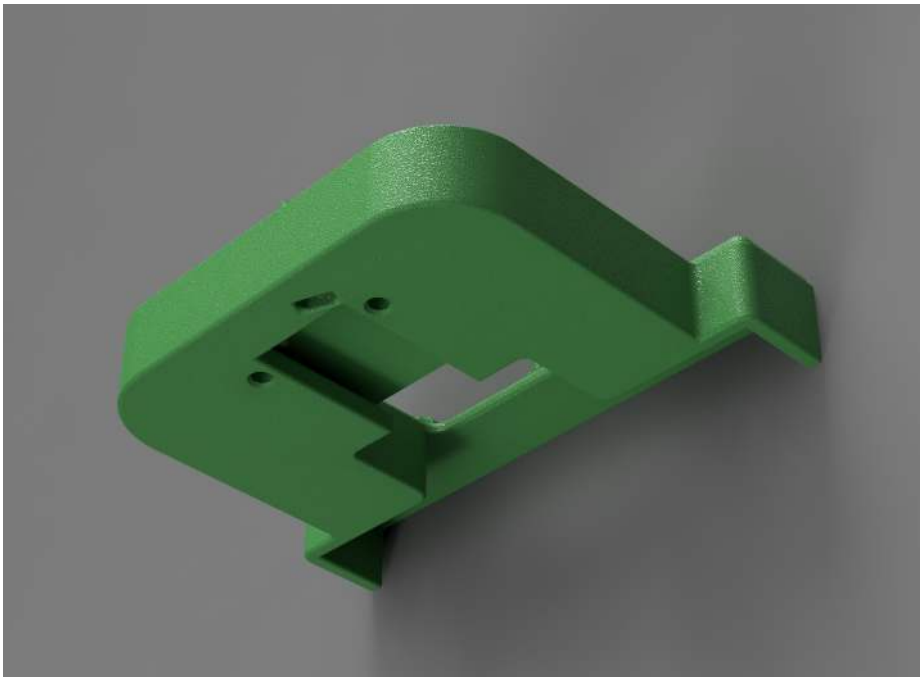


Figuur 3.7: 3D-geprinte adapter die de onderkant van de KKSB-behuizing verbindt met een 1/4”-20 UNC-schroefdraad voor montage op standaard cameramateriaal.

De camerahouder is een 3D-geprint onderdeel dat aan de zijkant van de behuizing vasthangt. Aan de bovenzijde is er plaats voor de camerasensor en de IR-ledring, aan de onderkant kan de IMU bevestigd worden. De geometrie zorgt ervoor dat de bekabeling van camera en IMU netjes via de zijkant van de behuizing naar de aansluitingen van de Raspberry Pi loopt. Figuur 3.8 toont het bovenaanzicht van de camerahouder en figuur 3.9 het onderaanzicht.



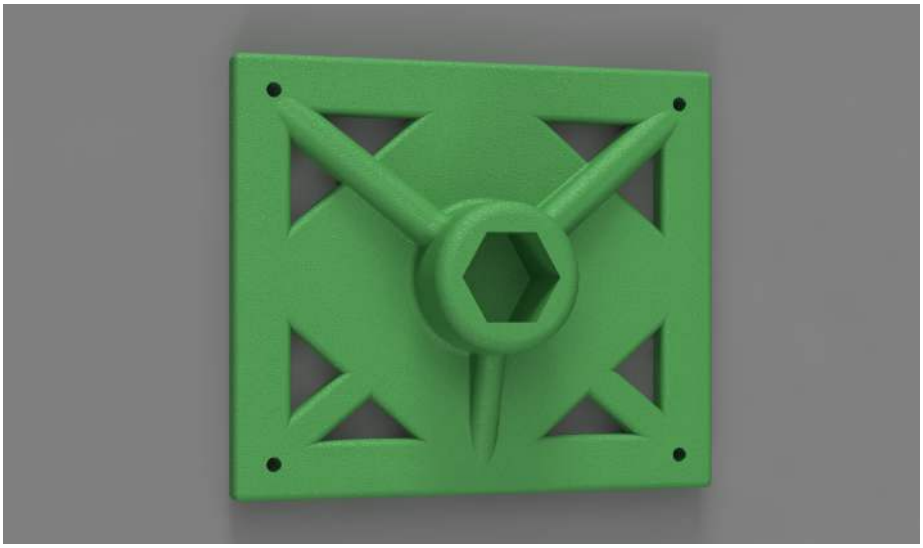
Figuur 3.8: Bovenaanzicht van de 3D-geprinte camerahouder met de bevestigingsvlakken aan de case en de uitsparing voor camera en IR-ledring.



Figuur 3.9: Onderaanzicht van de 3D-geprinte camerahouder met de houder voor de IMU. Deze positie houdt de IMU mechanisch gekoppeld aan de trackingcamera, wat nodig is voor een vaste camera-IMU-extrinsiek.

Op het touchscreen draait de Qt/QML-frontend. Het gebruikte scherm is een Elecrow 8 inch IPS-touchscreen met een resolutie van 1280×800 pixels, dat compatibel is met de Raspberry Pi [46]. De Raspberry Pi kan rechtstreeks op de achterkant van dit scherm gemonteerd worden, maar voor dit prototype is gekozen voor een algemenere bevestiging via de 3D-geprinte schermbracket. Die bracket zet de Raspberry Pi bevestigingspunten van het scherm opnieuw om naar opening voor een moer die de 1/4"-20 UNC-schroefdraad bevat, zodat het scherm via dezelfde montagestandaard als de overige componenten op de opstelling bevestigd kan worden. Het scherm wordt gebruikt voor bediening, *preview*, statusweergave, kalibratie en kaartbeheer. Daardoor kan het prototype optioneel als zelfstandige module gebruikt worden zonder externe computer. Figuur 3.10 toont het touchscreen en figuur 3.11 de bijbehorende schermbracket.

Figuur 3.10: Elecrow 8 inch IPS-touchscreen (1280×800) voor lokale bediening en visualisatie van de trackingstatus. Het scherm is een gebruiksgemakuitbreiding; de basistracker kan ook zonder touchscreen via SSH gebruikt worden [46].

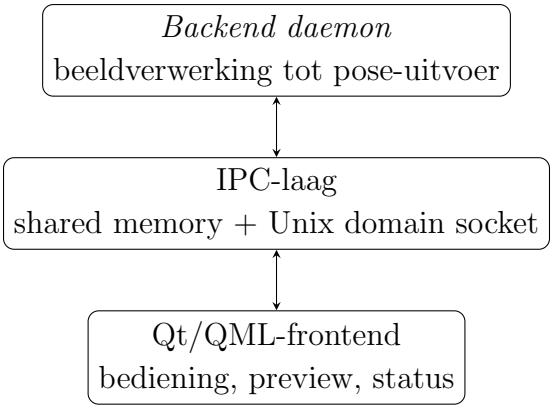


Figuur 3.11: 3D-geprinte schermbracket die de bevestigingspunten van het touchscreen omzet naar een 1/4"-20 UNC-schroefdraad voor montage op de opstelling.

3.2 Globale systeemarchitectuur

Het prototype bestaat uit twee softwareprocessen. Het eerste proces is een daemon: een achtergrondprogramma dat zonder directe gebruikersinteractie blijft draaien en de tijdskritische verwerking beheert. In dit systeem verwerkt de daemon de beeldverwerking tot pose-uitvoer. Het tweede proces is de grafische frontend, waarmee de gebruiker het systeem bedient en de status controleert.

Tussen beide processen ligt een IPC-laag. IPC staat voor *inter-process communication*: de mechanismen waarmee afzonderlijke processen gegevens en commando's uitwisselen. Voor continue status- en beelddata gebruikt het prototype *shared memory*. Voor discrete commando's en antwoorden gebruikt het een *Unix domain socket*. Die scheiding voorkomt dat grote previewbeelden of snel wijzigende posedata de bediening van het systeem blokkeren.



Figuur 3.12: Globale opsplitsing tussen daemon, IPC-laag en frontend. De daemon verwerkt camerabeelden en IMU-data, de IPC-laag wisselt status en commando's uit, en de frontend toont bediening en diagnose.

Deze opsplitsing beschermt de tracking tegen vertragingen in de gebruikersinterface. De frontend leest alleen de meest recente status, pose en preview uit. Omgekeerd kan de gebruiker via commando's acties starten, zoals kalibratie, kaartopbouw of tracking, zonder dat de frontend zelf beeldverwerking uitvoert.

3.3 Interprocescommunicatie

De communicatie tussen daemon en frontend bestaat uit twee complementaire IPC-mechanismen. Continue datastromen gaan via shared memory. Discrete commando's, antwoorden en *events* gaan via een Unix domain socket. Die scheiding voorkomt dat grote of frequente datastromen de commandostructuur blokkeren.

3.3.1 Shared memory

Shared memory wordt gebruikt voor gegevens die continu veranderen en alleen in hun meest recente toestand relevant zijn. Voorbeelden zijn de actuele pose, trackingstatus, IMU-status, FreeD-statistieken en previewbeelden. De daemon schrijft deze gegevens naar een vaste gedeelde geheugenstructuur. De frontend leest die structuur periodiek uit.

Om te vermijden dat de frontend halfgeschreven data leest, wordt een *seqlock*patroon gebruikt. De schrijver markeert een veld tijdelijk als instabiel, schrijft de data en markeert het daarna opnieuw als stabiel. De lezer accepteert alleen data waarvan de sequentiewaarde tijdens het lezen niet verandert. Dit is lichter dan klassieke synchronisatie met *locks* en past goed bij realtime statusdata.

3.3.2 Unix domain socket

Een Unix domain socket is een lokaal IPC-mechanisme voor communicatie tussen processen op dezelfde machine. In tegenstelling tot een netwerksocket gebruikt ze geen IP-adres of netwerkinterface, maar een pad in het bestandssysteem. Dat past bij dit prototype, omdat daemon en frontend op dezelfde Raspberry Pi draaien.

De Unix domain socket wordt gebruikt voor discrete interacties. De frontend stuurt JSON-commando's zoals configuratie opvragen, tracking starten, kaart bouwen of systeem stoppen. De daemon antwoordt met events zoals statusupdates, foutmeldingen, bestandslijsten of voltooiing van een kalibratiestap. Omdat deze berichten tekstueel en gestructureerd zijn, blijft de interface uitbreidbaar zonder de shared-memory-structuur telkens te wijzigen.

3.4 Taakverdeling op de quad-core processor

De Raspberry Pi 5 beschikt over vier processorkernen. De software gebruikt die niet als vier volledig gescheiden subsystemen, maar als een combinatie van vaste *workerthreads* en door Linux geplande neventaken. De daemon pint drie tijdskritische workerthreads met *CPU-affinity* op vaste processorkernen. De hoofdthread van de daemon, de frontend, de QThreads van de frontend, tijdelijke kalibratie- of kaartbouwtaken en besturingssysteemtaken hebben geen vaste processorkern in de huidige configuratie en worden door de Linux-*scheduler* beheerd.

3.4.1 Hoofdthread van de daemon

De hoofdthread van de daemon is het besturingspunt van het backendproces. Deze beheert de Unix domain socket, verwerkt commando's van de frontend en start langere procedures zoals kalibratie of kaartopbouw als afzonderlijke taak. Omdat deze thread in de code geen expliciete CPU-affinity krijgt, plaatst de Linux-scheduler hem op een beschikbare processorkern.

3.4.2 Core 1: camera, beeldverwerking en preview

De eerste tijdskritische workerthread is met CPU-affinity aan core 1 gekoppeld. Deze thread leest beelden van de trackingcamera, corrigeert lensdistortie en voert sterdetectie uit. Voor de frontend wordt een verkleind previewbeeld met annotaties naar shared memory geschreven. Voor de visuele poseberekening worden de gedetecteerde stercentra, het bijbehorende beeld en een tijdstempel in een dubbelgebufferd *frameslot* geplaatst.

Het ontwerp gebruikt hier een *producer-consumer*-patroon. De camerataak mag nieuwe beelden blijven aanleveren zonder te wachten op localisatie of tracking. De volgende verwerkingstaak leest telkens het meest recente stabiele frameslot. Daardoor blijft de beeldinvoer bruikbaar wanneer een enkele poseberekening langer duurt dan verwacht.

3.4.3 Core 2: localisatie en tracking

De tweede tijdskritische workerthread is met CPU-affinity aan core 2 gekoppeld. Deze thread leest de meest recente detecties en koppelt die aan de ingeladen sterkaart. In localisatiemodus is er nog geen vorige pose beschikbaar en wordt een globale posehypothese gezocht. Na succesvolle localisatie schakelt het systeem over naar trackingmodus. In die modus wordt de vorige pose gebruikt als basis voor een snellere update.

De taak schrijft een visuele posemeting weg met kwaliteitsinformatie, zoals het aantal gebruikte punten en de reprojectiefout. Die informatie wordt gebruikt voor statusweergave en als correctiemeting voor de IMU-fusie. Wanneer de trackingkwaliteit te laag wordt, valt het systeem terug naar localisatiemodus.

3.4.4 Core 3: IMU, filter en FreeD-uitvoer

De derde tijdskritische workerthread is met CPU-affinity aan core 3 gekoppeld. Deze thread leest de IMU uit en beheert de uiteindelijke pose-uitvoer. De IMU-metingen worden gebruikt in een ESKF-laag die inertiële voorspelling combineert met de laatst beschikbare visuele posemeting. Daarna wordt op de samengestelde uitvoerpose nog een *One Euro Filter* toegepast om hoogfrequente schommelingen in de doorgestuurde pose te dempen. De uitvoer van deze taak is de pose die door de rest van het systeem als finale trackingpose wordt beschouwd.

Deze finale pose wordt naar shared memory geschreven, zodat de frontend de actuele positie en oriëntatie kan tonen. Daarnaast wordt de pose als FreeD-pakket over UDP verstuurd naar de virtual-productionomgeving. Wanneer nog geen gefilterde pose beschikbaar is, kan de uitvoerlaag terugvallen op de laatst bekende visuele pose.

3.4.5 Frontendproces

De Qt/QML-frontend draait als afzonderlijk proces en is niet vast aan een processorkern gekoppeld. De code voorziet wel een optionele compileervlag waarmee de frontend op core 0 gepind kan worden, maar standaard laat het systeem de planning van dit proces over aan de Linux-scheduler. Binnen de frontend draaien onder meer de QML-interface, een shared-memorylezer en een *socketclient*.

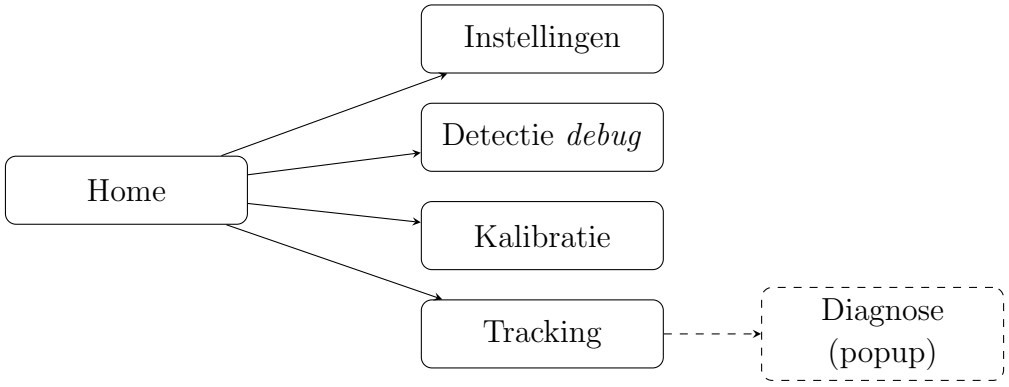
De frontend bevat geen eigen trackinglogica, maar leest status, preview en pose uit shared memory. Commando's worden via de Unix domain socket naar de daemon gestuurd. De schermen volgen de operationele stappen van het prototype: detectie en preview, kalibratie, kaartbeheer, tracking, instellingen en statuscontrole.

3.5 Procesgrenzen

De procesgrens tussen daemon en frontend is vooral een grens tussen trackingkern en bediening. De daemon blijft eigenaar van sensoren, timing, poseberekening en uitvoer; de frontend krijgt alleen toegang via de formele IPC-laag. Daardoor kan dezelfde trackingkern bijvoorbeeld via het touchscreen of *remote* met SSH bestuurd worden. Een tragere gebruikersinterface hoeft de trackingketen dan niet rechtstreeks te blokkeren, terwijl de interface wel beschikbaar blijft voor statusweergave en diagnose.

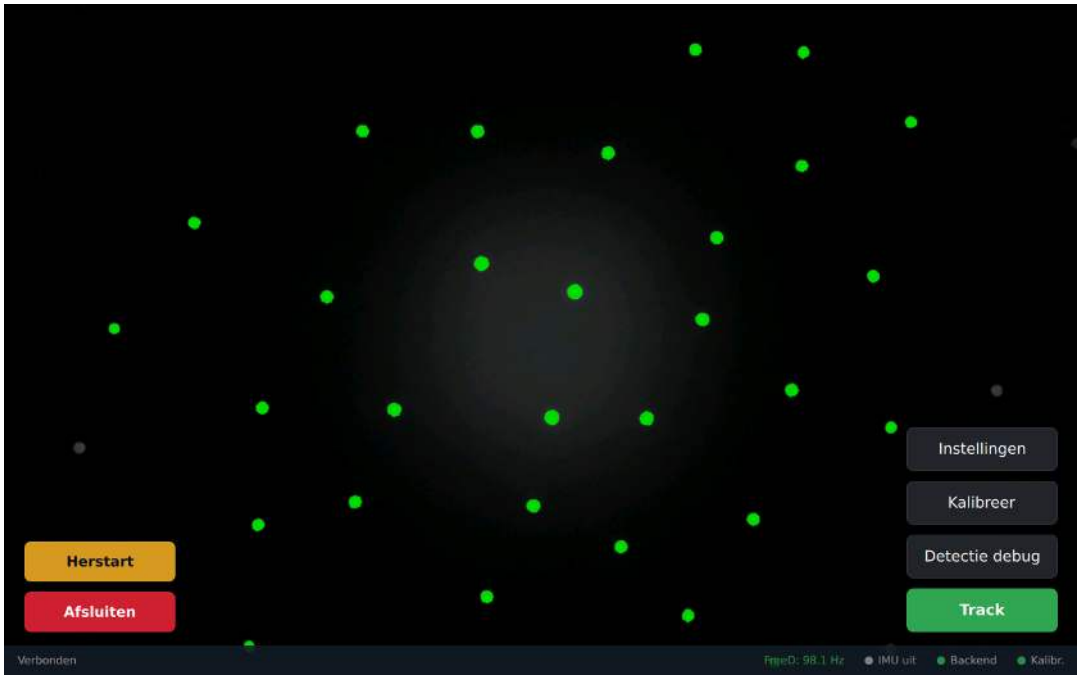
3.6 Workflow en frontendtoestanden

De frontend organiseert de bediening van het prototype rond een beperkt aantal toestanden. Die toestanden beschrijven niet de volledige interne algoritmische toestand van de daemon, maar de stappen die de gebruiker nodig heeft om het systeem voor te bereiden, te controleren en te gebruiken.



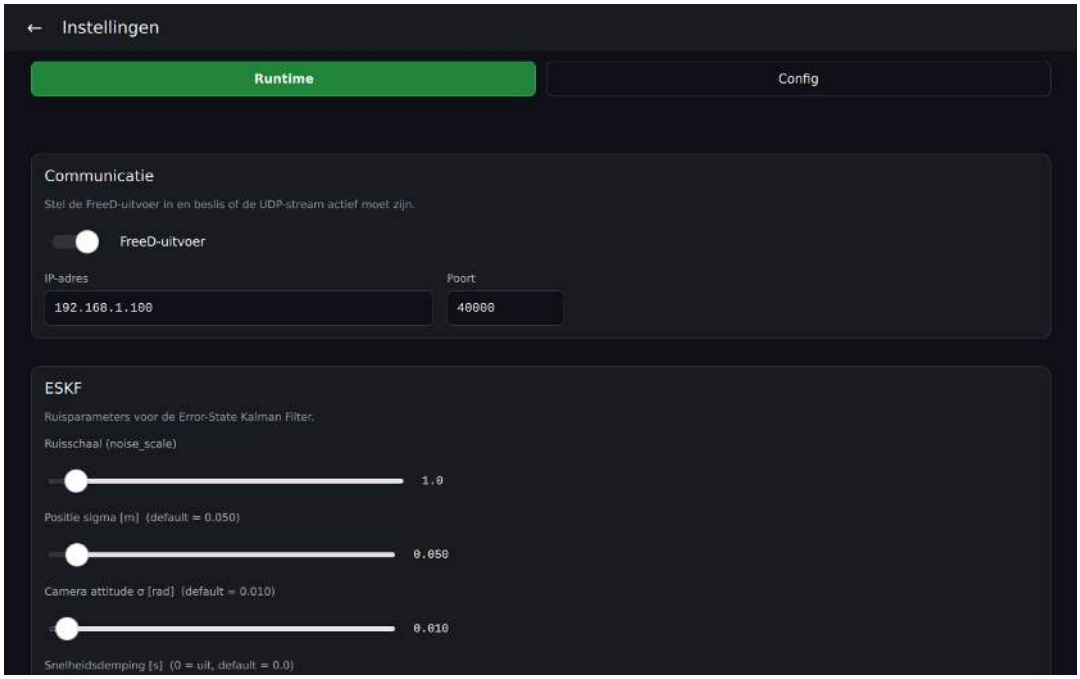
Figuur 3.13: Workflow van de frontendtoestanden: vanuit het homescherm zijn instellingen, detectie debug, kalibratie en tracking direct bereikbaar. Bij verlies van tracking verschijnt een diagnosepopup.

De normale workflow start bij het homescherm. Figuur 3.14 toont dit scherm. Het geeft aan of de daemon bereikbaar is, of de IMU beschikbaar is, of een kalibratiebestand geladen is en of de FreeD-uitvoer actief is. Het scherm fungeert daardoor als controlepunt en is zo ook het vertrekpunt naar alle andere toestanden: instellingen, detectie debug, kalibratie en tracking zijn allemaal rechtstreeks vanuit het homescherm bereikbaar.



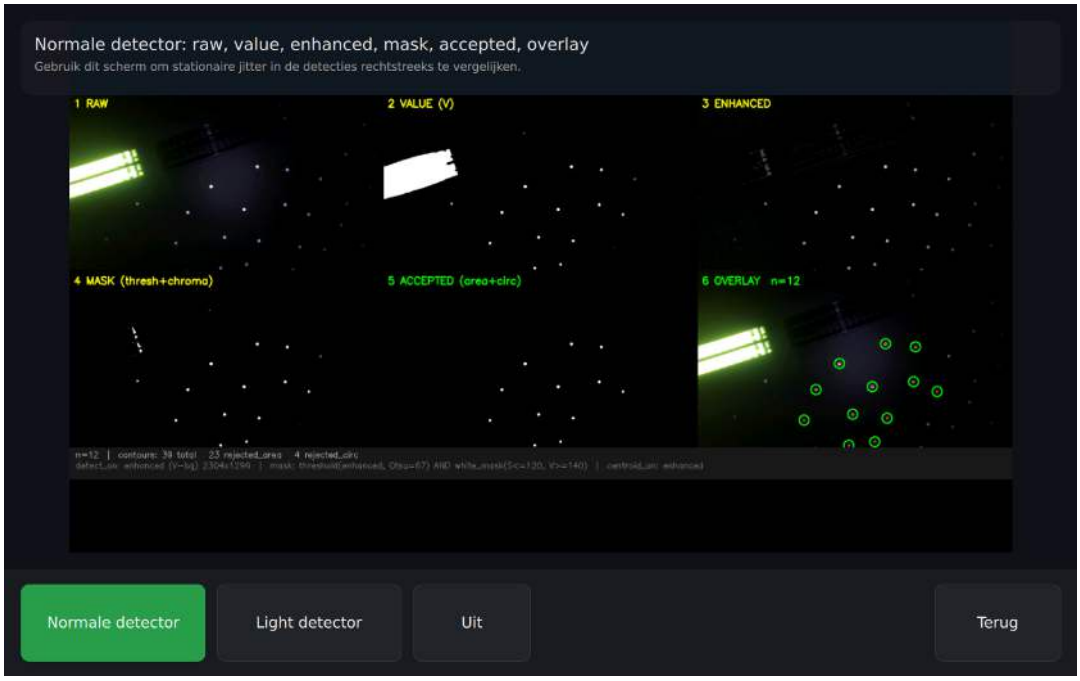
Figuur 3.14: Startscherm van de Qt/QML-frontend met de actuele systeemstatus en toegang tot de belangrijkste bedieningsschermen.

Via het instellingenscherm kan de gebruiker systeemparameters aanpassen zoals netwerkinstellingen voor de FreeD-uitvoer, cameraparameters en filterinstellingen. Deze toestand is niet noodzakelijk voor elke sessie, maar biedt toegang tot de configuratie zonder de trackingkern opnieuw te hoeven starten.



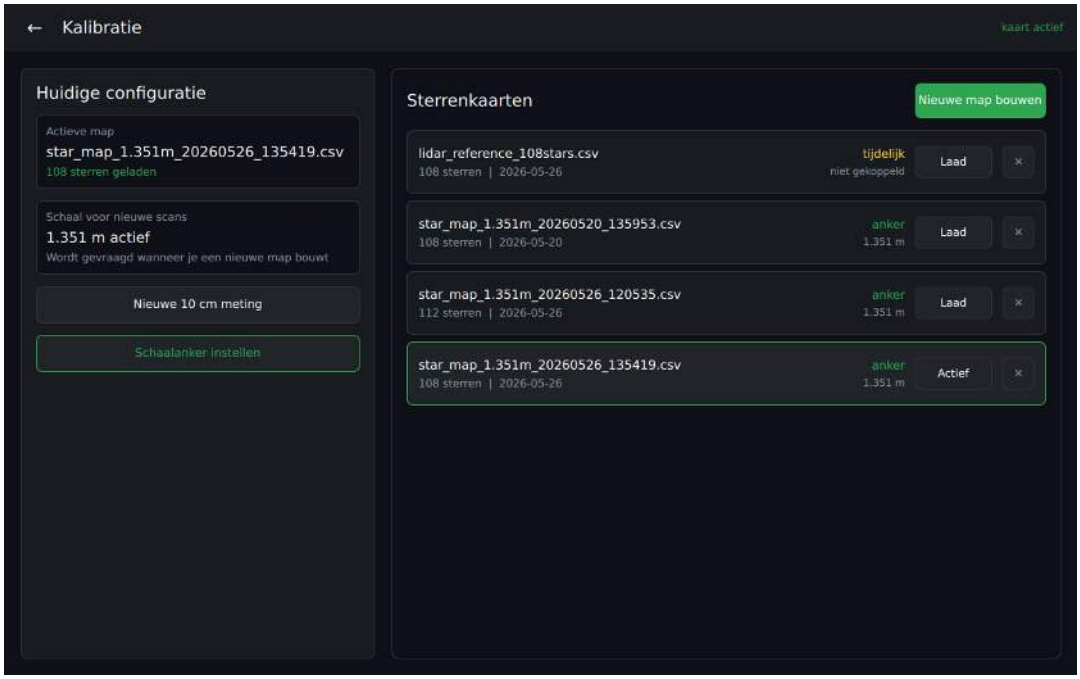
Figuur 3.15: Instellingenscherm van de Qt/QML-frontend met systeemconfiguratie, camera-instellingen en netwerkparameters.

Figuur 3.16 toont het detectie-debugscherm, hier ziet de gebruiker elk belangrijk beeld van de voorverwerking en als laatst een *overlay* van gedetecteerde markers. Deze toestand is belangrijk om belichting, markerzichtbaarheid en lenscorrectie te confirmeren.

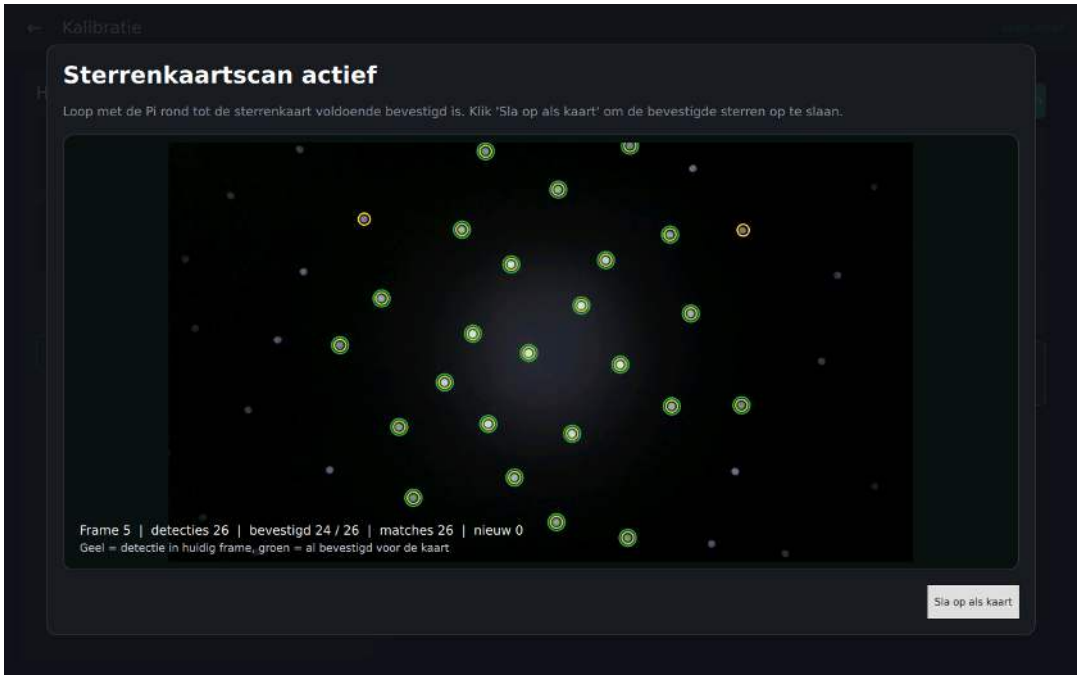


Figuur 3.16: Detectie-debugscherm met camerabeelden en marker-overlay.

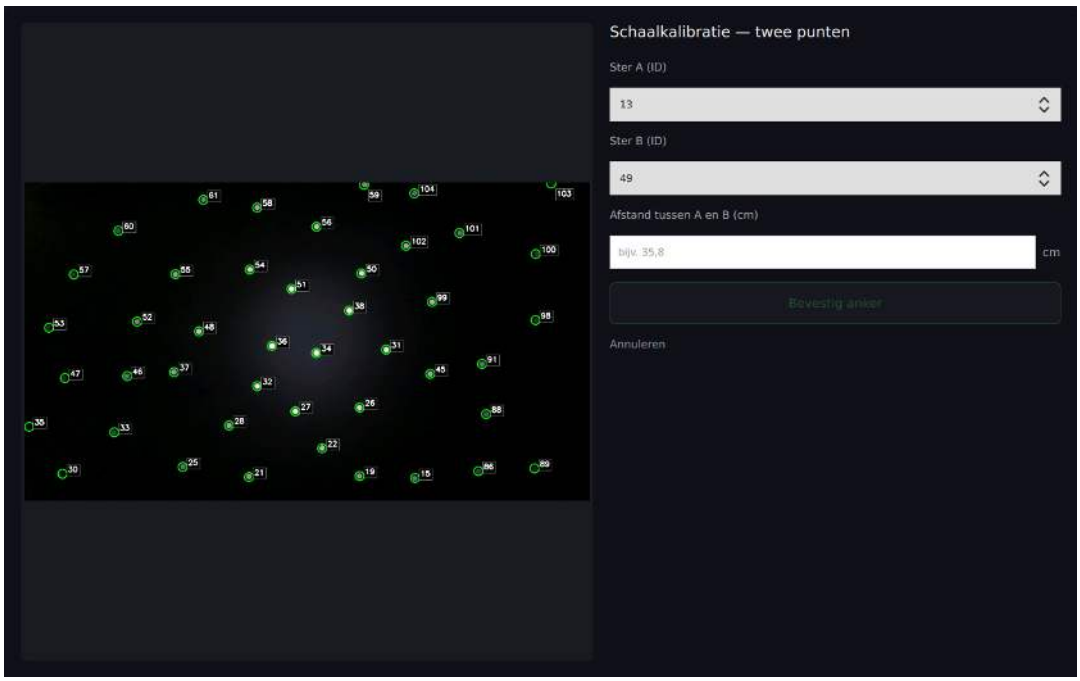
De kalibratie- en kaartbeheertoestand bundelt de voorbereidende acties. De gebruiker kan hier kalibratiegegevens controleren, een bestaande markerkaart laden, een nieuwe kaartopbouw starten of een schaalanker toepassen. De frontend voert deze berekeningen niet zelf uit, maar start de overeenkomstige taken in de daemon en toont de voortgang en resultaten. Figuur 3.17 toont het kalibratieoverzicht. Figuur 3.18 toont de kaartopbouwtoestand en figuur 3.19 de toestand waarin een schaalanker gekozen wordt.



Figuur 3.17: Kalibratie- en kaartbeheerscherm van de frontend.

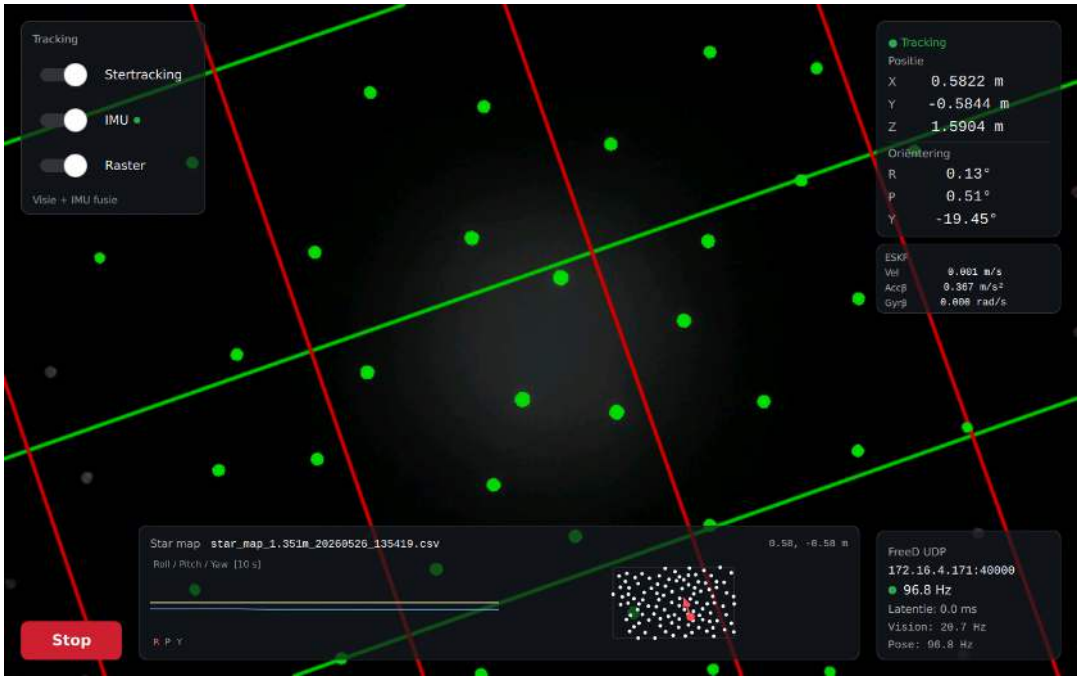


Figuur 3.18: Kaartopbouwscherm van de frontend. Tijdens deze toestand beweegt de gebruiker de opstelling door de ruimte zodat de zichtbare markers aan de sterkaart kunnen worden toegevoegd.



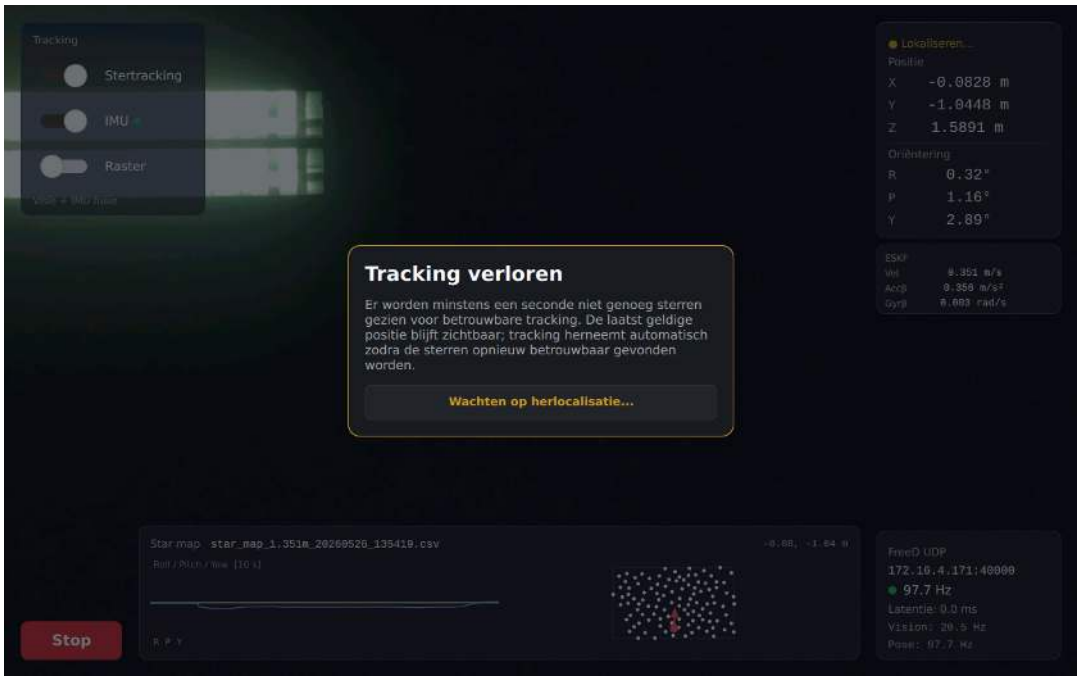
Figuur 3.19: Schaalankerscherm waarin de gebruiker twee markers uit de opgebouwde sterkaart kan kiezen. De fysieke afstand tussen deze markers wordt gebruikt om de ruwe kaart naar een nauwkeurigere metrische schaal te herschalen.

Wanneer de voorbereiding volledig is, kan de gebruiker trackingscherm starten dat te zien is in figuur 3.20. In deze toestand toont de frontend de actuele pose, trackingstatus en FreeD-status. Stertracking en IMU-gebruik kunnen hier afzonderlijk aan- of uitgezet worden: wanneer stertracking uitstaat, valt het systeem terug op puur inertiële voorspelling; wanneer de IMU uitstaat, gebruikt het systeem enkel de visuele posemetingen zonder filterondersteuning.



Figuur 3.20: Trackingscherm met preview, status, pose-informatie en afzonderlijke toggles voor stertracking en IMU.

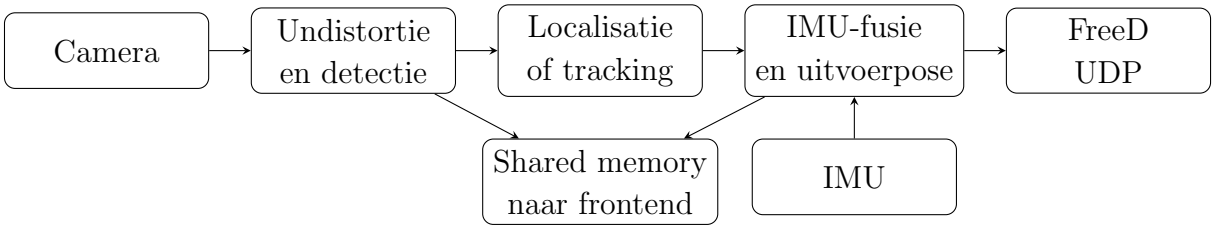
Wanneer er één seconde lang geen enkele ster gedetecteerd wordt verschijnt het diagnosevenster dat te zien is in figuur 3.21. Als de pose-uitvoer van te lage kwaliteit is vraagt het systeem terug een herlocalisatie aan zonder onderbreking.



Figuur 3.21: Diagnosevenster van de frontend bij trackingverlies. Het venster verschijnt wanneer één seconde geen sterren gezien worden.

3.7 *Runtime pipeline*

Tijdens *runtime* loopt de verwerking als een continue keten van sensordata naar pose-uitvoer. De camera levert beelden aan de detectietaak. De detecties worden via frameslots doorgegeven aan de localisatie- of trackingtaak. Die taak produceert een visuele posemeting. Parallel leest de IMU-taak inertiële metingen in. De finale pose wordt door de uitvoertaak naar de frontend en naar de renderomgeving gestuurd. Figuur 3.22 toont de volledige dataflow.



Figuur 3.22: Runtime dataflow van camerabeeld en IMU-meting naar FreeD-uitvoer en frontend. De figuur toont hoe detectie, localisatie, tracking, ESKF-fusie en netwerkuitvoer in de softwareketen op elkaar aansluiten.

Deze pipeline maakt onderscheid tussen previewdata en posedata. Previewbeelden zijn bedoeld voor diagnose en gebruikersfeedback. Posegegevens zijn tijdskritisch en worden door de daemon beheerd. De frontend kan daardoor vertraagd of tijdelijk niet actief zijn zonder dat de kernverwerking noodzakelijk stopt.

3.8 Configuratie en opslag

Het systeem gebruikt persistente configuratiebestanden voor camera-instellingen, filterparameters, FreeD-instellingen en algemene runtimevlaggen. Sterkaarten en kalibratiegegevens worden afzonderlijk opgeslagen met bijhorende metadata. Die opsplitsing maakt het mogelijk om dezelfde software met verschillende fysieke opstellingen of markerkaarten te gebruiken.

Het systeemontwerp behandelt deze opslag als onderdeel van de architectuur, omdat dit een belangrijke functionaliteit is voor het efficiënt gebruik van het prototype.

3.9 Motivatie van de ontwerpkeuzes

De gekozen architectuur volgt rechtstreeks uit de eisen van een realtime cameratracker. De scheiding tussen daemon en frontend beschermt de tracking tegen vertragingen in de gebruikersinterface. De verdeling over camera, visuele poseberekening en IMU/FreeD-uitvoer sluit aan bij de verschillende frequenties en foutmodellen van de sensoren. Shared memory is geschikt voor snelle status- en previewdata, terwijl een Unix domain socket een eenvoudige commandolaag biedt.

Het ontwerp is ook uitbreidbaar. Alternatieve detectoren, posemethoden of filters kunnen binnen dezelfde systeemstructuur worden vervangen zolang ze dezelfde tussenproducten leveren: detectiepunten, visuele posemetingen, kwaliteitsmaten en finale pose-uitvoer. Daardoor blijft het systeem geschikt voor verdere experimenten zonder dat de volledige applicatiearchitectuur telkens opnieuw ontworpen moet worden.

Hoofdstuk 4

Algoritmische uitwerking

Dit hoofdstuk beschrijft de algoritmische werking van het prototype. Waar het systeemontwerp de systeemarchitectuur opdeelt in processen, *threads* en communicatiekanalen, beschrijft dit hoofdstuk de inhoudelijke verwerking van de data. De pipeline vertrekt van een camerabeeld met retroreflectieve plafondmarkers en eindigt bij een pose die via shared memory en FreeD beschikbaar wordt gemaakt.

4.1 Coördinatenstelsels en poseconventies

De pipeline gebruikt drie belangrijke coördinatenstelsels. Het eerste is de pixelruimte van de trackingcamera. Omdat het ruwe camerabeeld een merkbare lensdistortie heeft, wordt elk frame eerst met een vooraf berekende *remap* naar een undistorted beeld omgezet. De gedetecteerde stercentra bevinden zich daarna in undistorted pixelcoördinaten.

De tweede ruimte is het wereldframe van de sterkaart. Een sterkaart bestaat uit markers met een uniek ID en een positie

$$p_i^W = (x_i, y_i, z_i), \tag{4.1}$$

uitgedrukt in meter. Voor plafondmarkers ligt het markeroppervlak in de kaartvoorstelling op $z = 0$. De camera bevindt zich onder dat vlak, waardoor de camerapositie in het wereldframe typisch een negatieve z-component heeft. De wereldpose zelf blijft dus fysisch consistent: de camera ligt onder het plafondvlak. Alleen de uitvoerlaag zet deze negatieve wereld-z om naar een positieve hoogte voor de gebruikersinterface en FreeD.

De derde ruimte is het cameraframe dat OpenCV gebruikt bij PnP. OpenCV geeft een transformatie T_{CW} terug waarvoor geldt

$$p^C = R_{CW} p^W + t_{CW}. \tag{4.2}$$

De software gebruikt de inverse T_{WC}: de pose van de camera in het wereldframe. Deze omzetting wordt centraal uitgevoerd zodat localisatie, tracking, filtering en uitvoer dezelfde conventie gebruiken.

4.2 Cameravoorverwerking en detectie

De visuele verwerking start bij het camerabeeld. Op dat moment is er nog geen ster-ID, geen kaartpositie en geen pose beschikbaar: het systeem ziet enkel heldere vlekken in een breedhoekbeeld. De eerste taak van de pipeline is daarom om elk camerabeeld om te zetten naar een compacte lijst met 2D-centra van mogelijke plafondmarkers. Pas daarna kunnen kaartopbouw, identificatie en posebepaling volgen.

De trackingcamera levert *frames* aan via `rpicam-vid`, dat bij het opstarten van de camerataak via `popen()` als subproces wordt gestart. Het programma streamt het ruwe beeldmateriaal als YUV420-planar via *stdout* rechtstreeks naar het werkgeheugen van het prototype; er wordt geen ruw beeld naar schijf geschreven. Het YUV420-formaat bestaat uit een volledig Y-vlak van $W \times H$ bytes gevolgd door twee gekrompen kleurkanalen van elk $\frac{W}{2} \times \frac{H}{2}$ bytes, wat neerkomt op $\frac{3}{2} \cdot W \cdot H$ bytes per frame. De camerataak leest telkens exact zoveel bytes en converteert het resultaat via `cv::COLOR_YUV2BGR_I420` naar een BGR-beeld.

De standaardconfiguratie gebruikt een resolutie van 2304×1296 pixels bij 20 frames per seconde en een sluitertijd van 500 μs. De gain staat in de configuratie op 0, waardoor de optie niet expliciet aan `rpicam-vid` wordt doorgegeven. Automatisch scherpstellen is uitgeschakeld; de vaste lensstand is ingesteld op positie 10. Ook voor de witbalans worden geen vaste *AWB gains* meegegeven.

Binnenkomende frames worden via een wachtrij verwerkt. Wanneer de wachtrij vol is, wordt het oudste frame verwijderd zodat de pipeline altijd het meest recente beeld verwerkt en geen achterstand opbouwt. Elk frame doorloopt vervolgens dezelfde keten: correctie van het camerabeeld, detectie van markercentra en overdracht naar de poseberekening via een `FrameSlot`. De details van die stappen worden in de volgende secties uitgewerkt.

Foutieve detecties worden niet in de camerataak zelf gefilterd. De beoordeling vindt later in de pose-estimatiestap plaats. Een detectieset die onvoldoende betrouwbare matches oplevert, krijgt het verdict Doubtful en wordt verworpen als geldige posemeting.

4.2.1 Correctie van lensdistortie per frame

Bij het opstarten laadt de camerataak de intrinsieke cameramatrix K en de distortiecoëfficiënten. Daarmee maakt de software eenmalig remap-matrices. Per frame past OpenCV die tabellen toe met `cv::remap`. Dit is sneller dan per frame een volledige correctie opnieuw te berekenen, omdat de projectiemapping niet telkens opnieuw opgesteld hoeft te worden.

Na deze stap werkt de rest van de verwerking met het undistorted beeld. Omdat het beeld zelf al gecorrigeerd is, wordt in de PnP-stappen een nulvector voor de distortie gebruikt. Dit voorkomt dat punten twee keer gecorrigeerd worden.

4.2.2 Detectie tijdens tracking en kaartopbouw

Niet elk gebruik van detectie heeft dezelfde eis. Tijdens continue tracking moet de camerataak snel genoeg blijven om de poseketen te voeden. Tijdens kaartopbouw en debug is de verwerkingssnelheid minder kritisch, maar moet de detector sterker bestand zijn tegen ongelijkmatige belichting en vals-positieve reflecties. De implementatie gebruikt daarom twee detectoren met

dezelfde outputvorm: een lijst met `cv::Point2f`-centra in undistorted pixelcoördinaten.

Voor continue tracking gebruikt de standaardconfiguratie `StarDetectorLight`. Deze detector werkt rechtstreeks op grijswaarden en houdt de beeldbewerking bewust beperkt. Voor kaartopbouw en uitgebreide debug gebruikt het systeem `StarDetector`. Ook daar wordt de eigenlijke helderheid op een grijswaardenbeeld beoordeeld, maar de detector gebruikt extra beeldinformatie om ongelijkmatige belichting en vals-positieve reflecties beter te onderdrukken. Het onderscheid zit dus niet in wat de rest van het algoritme ontvangt, maar in hoeveel beeldbewerking nodig is om de sterren betrouwbaar uit het beeld te halen.

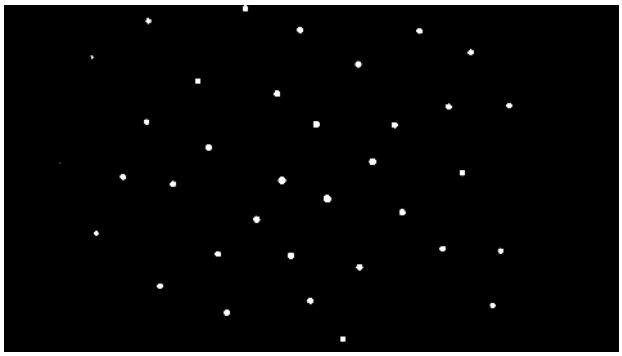
4.2.3 Van beeld naar detectiemasker

Sterdetectie tijdens tracking

De detector voor continue tracking zet het BGR-beeld eerst om naar een enkelkanaals grijswaardenbeeld via `cv::cvtColor`. Dat beeld blijft op volle resolutie bewaard in `gray_`, zodat de uiteindelijke centrumbepaling later op het originele pixelraster gebeurt. Voor het masker wordt het grijswaardenbeeld licht vervaagd met een Gaussische *blur*, waarvan de kernelgrootte een instelbare detectorparameter is. Deze stap dempt geïsoleerde heldere ruispixels en stabiliseert de rand van de blobs vóór de drempelstelling.

Na deze voorbewerking wordt het beeld naar een binair masker omgezet. Een binair masker bevat alleen zwarte en witte pixels: zwart betekent achtergrond, wit is een mogelijke marker. De detector kan hiervoor een ingestelde drempel gebruiken of, wanneer die waarde nul is, Otsu-binarisatie toepassen. Otsu bepaalt automatisch een drempel door de pixelintensiteiten in twee groepen te splitsen en de spreiding binnen die groepen zo klein mogelijk te maken. Figuur 4.1 en figuur 4.2 tonen hoe het ruwe beeld via drempelstelling tot een detectiemasker wordt teruggebracht bij respectievelijk goede en moeilijkere belichting.

(a) Ruw beeld



(b) Binair drempelmasker

Figuur 4.1: Goed belicht frame en het bijhorende drempelmasker van `StarDetectorLight`. De detector kan in zulke omstandigheden met beperkte beeldbewerking een duidelijk masker vormen.

(a) Ruw beeld

(b) Binair drempelmasker

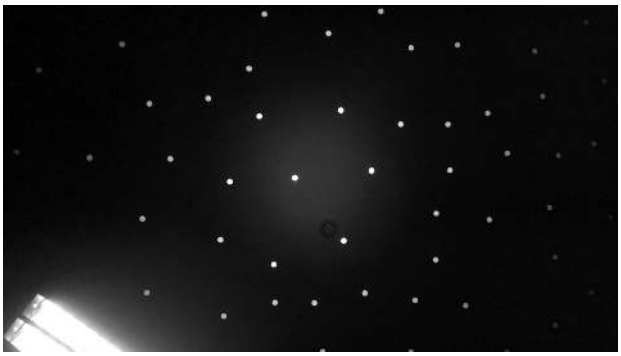
Figuur 4.2: Slechter belicht frame en het bijhorende drempelmasker van `StarDetectorLight`. Het resultaat blijft bruikbaar, maar toont tegelijk dat de kwaliteit van het masker rechtstreeks afhangt van de lokale belichting en de gekozen drempel.

Sterdetectie tijdens kaartopbouw

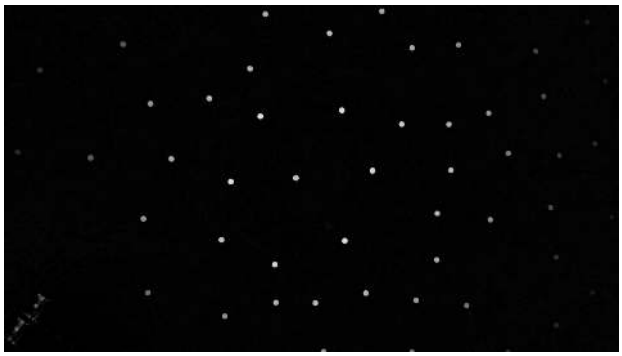
Voor kaartopbouw wordt meer informatie uit het beeld gebruikt. Eerst wordt het BGR-beeld omgezet naar HSV, voluit *hue*, *saturation* en *value*. Het V-kanaal is daarbij het helderheidsbeeld waarop de markerpieken gezocht worden; dit speelt in de praktijk dezelfde rol als een grijswaardenbeeld. Het S-kanaal wordt alleen gebruikt om gekleurde reflecties te onderdrukken. Een eerste witheidsmasker behoudt daarom pixels met lage saturatie en hoge helderheid. Met een bitsgewijze *and*-operatie blijven pixels over die tegelijk weinig kleur en veel helderheid hebben. Dit past bij retroreflectieve markers, die in de opname als neutrale heldere vlekken verschijnen.

Daarna volgt achtergrondonderdrukking op het V-kanaal. Het helderheidsbeeld wordt eerst met factor 2 verkleind via `INTER_AREA`. Op deze half-resolutieversie wordt een morfologische opening toegepast. Een morfologische opening bestaat uit erosie gevolgd door dilatie: kleine heldere structuren verdwijnen bij de erosie en de dilatie herstelt vooral de grotere, traag variërende achtergrond. Het resultaat is een glad achtergrondbeeld zonder lokale markerpieken. Door dat achtergrondbeeld terug naar volle resolutie te schalen en af te trekken van het oorspronkelijke V-kanaal blijven vooral lokale helderheidspieken over. Die tussenstap wordt in de code als `enhanced_` bewaard.

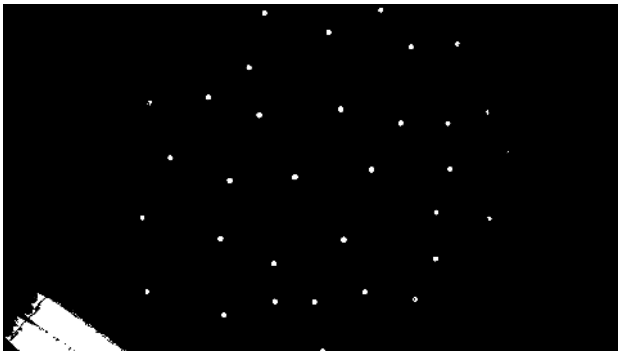
Op `enhanced_` wordt opnieuw een binair masker gemaakt. Hier is Otsu nuttig omdat de absolute helderheid tijdens kaartopbouw sterker kan variëren: de methode kiest per beeld een drempel die de donkere achtergrond en de heldere lokale pieken zo goed mogelijk scheidt. De gevonden waarde wordt wel begrensd met `peakFloor` en `peakCeil`. Daardoor leidt een uitzonderlijk lage drempel niet tot ruisdetecties en kan een zeer felle lichtbron de drempel niet zo hoog duwen dat zwakkere markers verdwijnen. Het piekmasker wordt daarna gecombineerd met het witheidsmasker. Het resultaat is het definitieve detectiemasker voor de volgende stap. Figuur 4.3 toont deze keten voor de detector tijdens kaartopbouw.



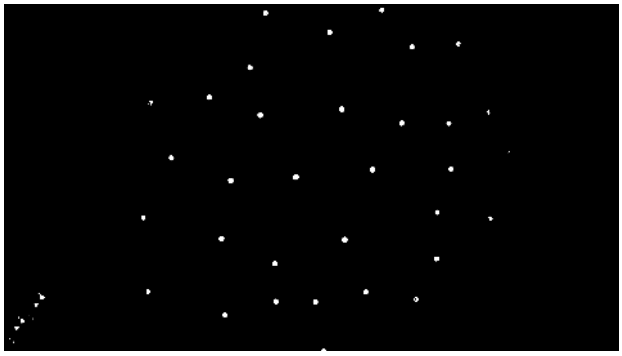
(a) Helderheidsbeeld V



(b) Achtergrond-gecorrigeerd beeld `enhanced_`



(c) Witheidsmasker



(d) Gecombineerd detectiemasker

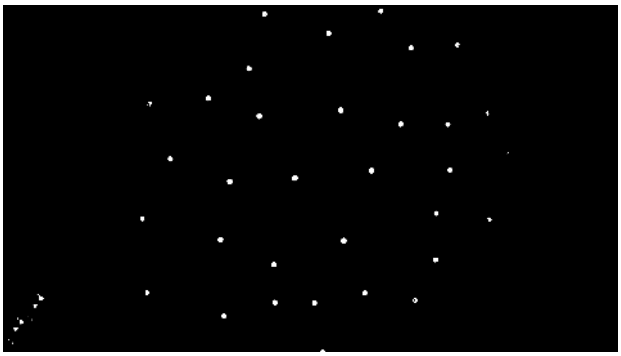
Figuur 4.3: Tussenbeelden van `StarDetector` op een slechter belicht frame. Linksboven staat het V-kanaal als helderheidsbeeld; rechtsboven het achtergrond-gecorrigeerde beeld `enhanced_`. Linksonder staat het witheidsmasker en rechtsonder het uiteindelijke gecombineerde detectiemasker. Deze extra stappen onderdrukken ongelijkmatige belichting en behouden tegelijk de heldere markerpieken.

4.2.4 Contourfiltering en centroïdebepaling

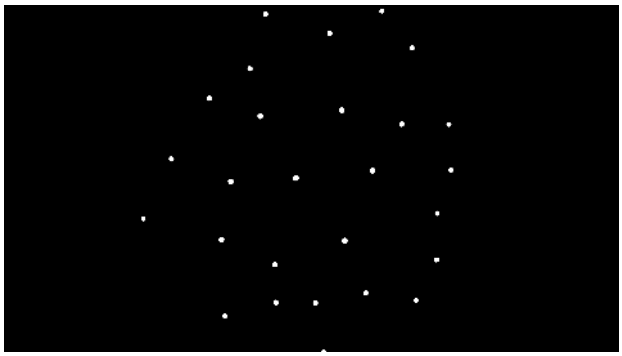
Zodra een binair detectiemasker beschikbaar is, verloopt de verwerking voor beide detectoren op dezelfde manier. Uit het masker worden met `cv::findContours` de buitencontouren gehaald. Elke contour wordt gefilterd op oppervlakte en circulariteit. De oppervlaktedrempels zijn detectorparameters en dienen alleen om zeer kleine ruisblobs en te grote reflecties te verwerpen. De circulariteit wordt berekend als

$$C = \frac{4\pi A}{P^2}, \tag{4.3}$$

waarbij A de contouroppervlakte is en P de contouromtrek. Een perfecte cirkel geeft $C = 1$; langgerekte of rafelige blobs geven een lagere waarde. Alleen compacte, voldoende ronde blobs blijven over.



(a) Gecombineerd detectiemasker van `StarDetector`



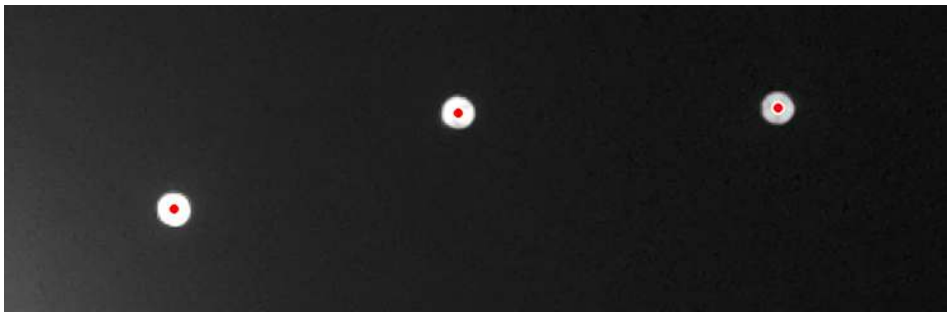
(b) Blobs na vormfiltering

Figuur 4.4: Links het gecombineerde detectiemasker voor een moeilijk belicht frame. Rechts de blobs die na filtering op oppervlakte en circulariteit overblijven. Op die manier worden ruisblobs en onregelmatige reflecties verworpen voordat een centroïde wordt bepaald.

Het centrum van elke geldige blob wordt niet geometrisch uit de contour afgeleid, maar via intensiteitsgewogen beeldmomenten. Bij `StarDetectorLight` worden die momenten berekend op het volle-resolutiebeeld `gray_`. Bij `StarDetector` gebeurt dit op het helderheidskanaal V. Het achtergrond-gecorrigeerde beeld `enhanced_` dient dus voor de drempelstelling, terwijl de centroïde zelf uit de oorspronkelijke helderheidswaarden volgt. De beeldmomenten worden

$$x_c = \frac{m_{10}}{m_{00}}, \qquad y_c = \frac{m_{01}}{m_{00}}. \tag{4.4}$$

met m_{00} de som van alle intensiteiten in de blob, m_{10} de som van $x \cdot I(x, y)$ en m_{01} de som van $y \cdot I(x, y)$. Deze gewogen centra liggen dichter bij het werkelijke optische midden van de marker dan zuiver geometrische contourmomenten, omdat helderdere pixels zwaarder meetellen.



Figuur 4.5: Ingezoomd voorbeeld van drie gedetecteerde markers. De rode markeringen tonen de intensiteitsgewogen centroïdes die uit de aanvaarde blobs worden berekend. Omdat de centroïde op basis van de helderheidsverdeling binnen elke blob wordt bepaald, ligt die niet noodzakelijk exact samen met het geometrische midden van de contour.

Een ellipsfit via `cv::fitEllipse` wordt bij `StarDetectorLight` berekend wanneer de contour voldoende punten bevat, maar enkel voor de *preview-overlay*. De ellips bepaalt dus niet het uiteindelijke stercentrum. De effectieve output blijft een lijst met subpixelcentra.

4.2.5 FrameSlot

Na detectie schrijft de camerataak het resultaat naar een dubbel gebufferd `FrameSlot`. Dubbel gebufferd betekent dat er twee slots beschikbaar zijn. Terwijl de camerataak een nieuw slot vult, kan de localisatie- en trackingtaak het andere stabiele slot lezen. Elk slot bevat het undistorted frame, de gedetecteerde punten en een tijdstempel. Een seqlockmarkering geeft aan of een slot stabiel gelezen kan worden. De localisatie- en trackingtaak leest telkens het meest recente stabiele slot. Hierdoor hoeft de camerataak niet te wachten op poseberekening en kan het systeem frames overslaan in plaats van op te stapelen.

4.3 Sterkaart en schaal

Na de detectorstap zijn per camerabeeld alleen afzonderlijke 2D-punten bekend. Voor een cameratracker is dat onvoldoende: dezelfde fysieke marker moet over meerdere beelden heen herkend worden en uiteindelijk een vaste positie in de ruimte krijgen. De volgende stap brengt de detecties uit meerdere frames daarom samen in één sterkaart. Die kaart vormt de geometrische referentie waartegen latere localisatie en tracking worden uitgevoerd.

4.3.1 StarMap3D en initiële schaal

De sterkaart wordt opgeslagen als `StarMap3D`. Dat datatype bevat een lijst van markerposities die later door localisatie en tracking wordt ingelezen. Een kaartpunt heeft de vorm

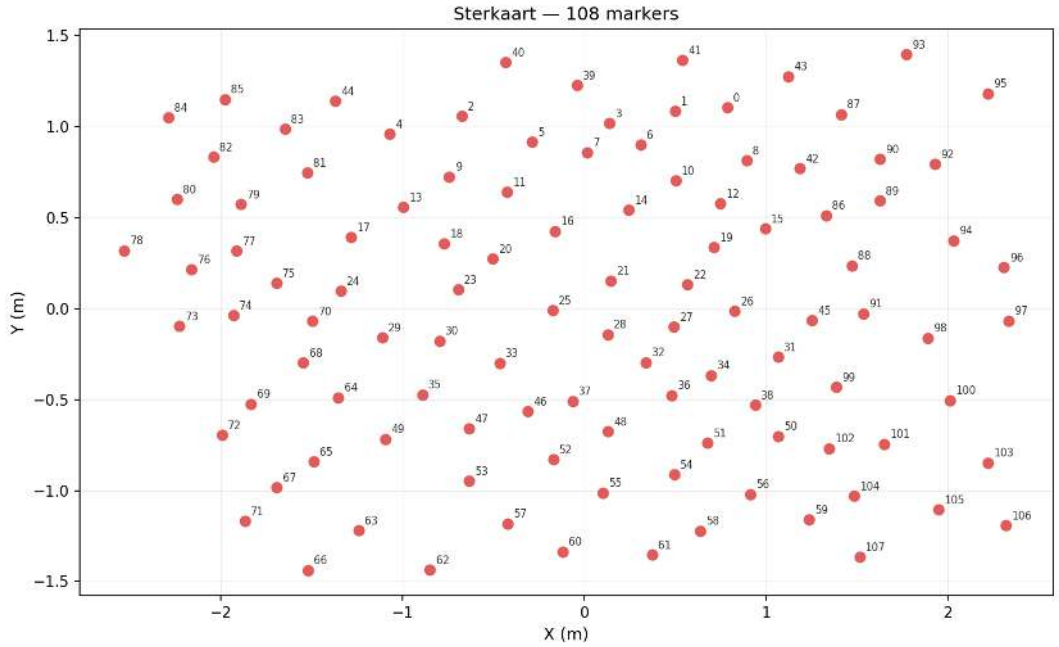
$$M_i = \{id_i, p_i^W\}, \tag{4.5}$$

waarbij id_i het marker-ID is en p_i^W de positie in meter. Omdat de markers op het plafond liggen, wordt de kaart als vlakke kaart gebruikt en geldt $z_i = 0$ voor alle markers. De positie p_i^W is dus een 3D-positie in het wereldstelsel, maar de kaart zelf ligt in één vlak.

De initiële schaal van de kaart komt uit de hoogtekalibratie. De camera wordt op een vaste positie geplaatst, een eerste frame wordt vastgelegd, daarna wordt de camera precies 10 cm verplaatst en een tweede frame wordt vastgelegd. Op basis van de parallaxverschuiving tussen de twee frames berekent het systeem de afstand van de camera tot het plafond. Die berekende plafondhoogte wordt als kalibratieparameter opgeslagen en dient als metrische schaal bij de kaartopbouw. De nauwkeurigheid hangt af van hoe precies de 10 cm-verplaatsing wordt uitgevoerd.

4.3.2 Kaartopbouw

De kaartopbouw gebeurt buiten de normale *runtimetracking*. Tijdens een scan wordt de opstelling door de ruimte bewogen zodat verschillende delen van het markerplafond zichtbaar worden. De afzonderlijke frames worden verwerkt met `StarDetector`, omdat deze detector beter omgaat met ongelijkmatige belichting tijdens een kaartscan. Detecties uit opeenvolgende frames worden aan elkaar gerelateerd en samengevoegd tot één kaart met marker-ID's en posities. Die kaart wordt rechtstreeks als `StarMap3D` opgeslagen en nadien opnieuw geladen door de localisatie- en *trackingpipeline*. Figuur 4.6 toont een opgebouwde sterkaart met marker-ID's en posities.



Figuur 4.6: Opgebouwde sterkaart met marker-ID's en metrische posities in het wereldframe. De kaart vormt de vaste referentie waartegen latere 2D-detecties worden geïdentificeerd en waaruit de camerapose wordt afgeleid.

4.3.3 Schaalanker

Nadat de kaart is opgebouwd, kan de schaal verfijnd worden met een twee-punts schaalanker. De gebruiker kiest twee marker-ID's a en b en geeft de fysiek opgemeten afstand d_{ab} tussen die markers op. De kaartafstand is

$$\hat{d}_{ab} = \|p_b - p_a\|. \tag{4.6}$$

De schaalfactor wordt dan

$$s = \frac{d_{ab}}{\hat{d}_{ab}}. \tag{4.7}$$

Alle kaartpunten worden met deze factor vermenigvuldigd. De implementatie schaalt daarbij ook de kalibratiehoogte mee, zodat kaart en hoogteconventie consistent blijven. Daarna wordt de kaart opnieuw opgeslagen samen met metadata over het gebruikte anker. Deze optionele stap compenseert fouten die zijn veroorzaakt door een onnauwkeurige 10 cm-verplaatsing tijdens de hoogtekalibratie en beperkt daarmee de resterende schaalfouten in de pose-output.

4.4 Geometrische indexering

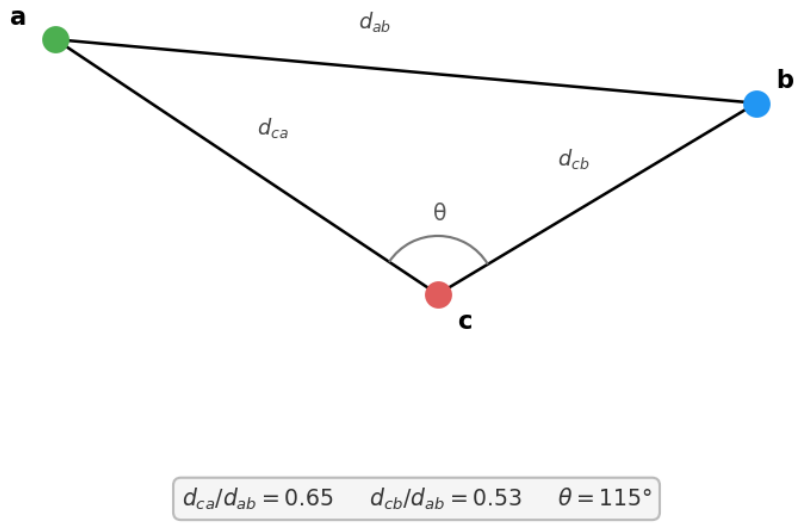
De sterkaart bevat na de kaartopbouw metrische markerposities, maar de markers zelf blijven visueel identiek. Een heldere blob in het beeld draagt dus geen eigen ID. De koppeling tussen beeldpunt en kaartmarker moet daarom uit de lokale geometrie van de markerposities komen. De geometrische index maakt die stap efficiënt: nadat een kaart is opgebouwd of geladen, worden de lokale patronen rond elke kaartmarker berekend. Een nieuw camerabeeld hoeft daardoor niet telkens brute-force met de volledige kaart vergeleken te worden.

4.4.1 Tripletdescriptoren

Omdat de markers visueel uniform zijn, kan een gedetecteerde blob niet op uiterlijk aan een kaartmarker gekoppeld worden. Daarom bouwt `StarIndex3D` een geometrische index op basis van lokale triplets. Voor elke centrale kaartmarker c worden de dichtstbijzijnde buren gezocht. Voor elk buurpaar a, b wordt een descriptor berekend:

$$\phi(c, a, b) = \left(\frac{d_{ca}}{d_{ab}}, \frac{d_{cb}}{d_{ab}}, \cos \angle acb \right). \tag{4.8}$$

Deze descriptor gebruikt afstandsverhoudingen en een hoekterm. Daardoor is hij invariant voor translatie, rotatie en uniforme schaal. Dat is nuttig omdat dezelfde markerconstellatie vanuit verschillende hoogten en rotaties gezien kan worden. Figuur 4.7 illustreert de tripletdescriptor.



Figuur 4.7: Tripletdescriptor voor globale identificatie van lokale markerconstellaties. De descriptor combineert afstandsverhoudingen en een hoekterm, zodat een groep naburige markers ook bij schaal- en rotatieverschillen herkenbaar blijft.

4.4.2 Voting tijdens query

Tijdens localisatie worden voor de detecties in het frame dezelfde tripletdescriptoren opgebouwd. Elke detectiedescriptor wordt vergeleken met de kaartdescriptoren binnen een tolerantie. Een overeenkomst levert een stem op voor een kandidaatpaar

$$(d_i, m_j), \tag{4.9}$$

waarbij d_i een detectie-index is en m_j een kaartmarker-index. De paren worden gerangschikt op stemgewicht. Daarna wordt een *greedy* selectie gemaakt waarbij een detectie en een kaartmarker elk maar een keer gebruikt mogen worden.

De votingstap is nuttig omdat één tripletovereenkomst ambigu kan zijn. Bij uniforme markers en repetitieve lokale patronen kunnen meerdere kaarttriplets ongeveer dezelfde afstandsverhoudingen hebben. Een correct detectie-markerpaar komt echter in meerdere overlappende triplets terug en verzamelt daardoor meerdere stemmen. Door eerst de stemmen te aggregeren en daarna pas unieke paren te kiezen, wordt niet elke losse overeenkomst even zwaar behandeld. Dat is robuuster dan alle ruwe overeenkomsten rechtstreeks te sorteren, omdat toevallige matches dan gemakkelijker meerdere keren dezelfde detectie of kaartmarker zouden kunnen opeisen.

De output van `StarIndex3D` is dus geen definitieve identificatie, maar een kandidaatset voor de poseberekening. In de volgende stap wordt nagegaan welke kandidaatparen samen één fysisch mogelijke camerapose opleveren. Een foutief paar kan lokaal nog plausibel lijken, maar zal meestal niet passen bij dezelfde projectie als de andere correcte paren. Daarom hoeft de indexeringsstap foutieve kandidaten nog niet volledig uit te sluiten.

4.5 Globale localisatie

Bij het opstarten heeft de runtime nog geen voorgaande pose. De eerste geldige pose moet daarom globaal uit één camerabeeld en de sterkaart worden afgeleid. Dezelfde stap wordt opnieuw gebruikt wanneer continue tracking later te weinig betrouwbare matches oplevert. Globale localisatie vertrekt van de kandidaatparen uit de geometrische index en zoekt rechtstreeks een absolute camerapose ten opzichte van de sterkaart.

4.5.1 Doel

`Localiser3D` verwerkt frames waarvoor geen betrouwbare voorgaande pose beschikbaar is. De localiser krijgt een detectieset, de sterkaart en de geometrische index. Het doel is een absolute pose T_{WC} te vinden op basis van één enkel frame.

4.5.2 Kandidaatcorrespondenties

Eerst vraagt `Localiser3D` kandidaatparen op bij `StarIndex3D`. Als de index minder dan vier kandidaten oplevert, wordt als fallback de combinatie van alle detecties met alle kaartmarkers gebruikt. Die fallback is computationeel duurder, maar laat RANSAC alsnog proberen een consistente subset te vinden.

Uit de kandidaatparen worden twee lijsten opgebouwd:

- 3D-objectpunten uit de sterkaart, in meter;
- 2D-beeldpunten uit de detectielijst, in undistorted pixelcoördinaten.

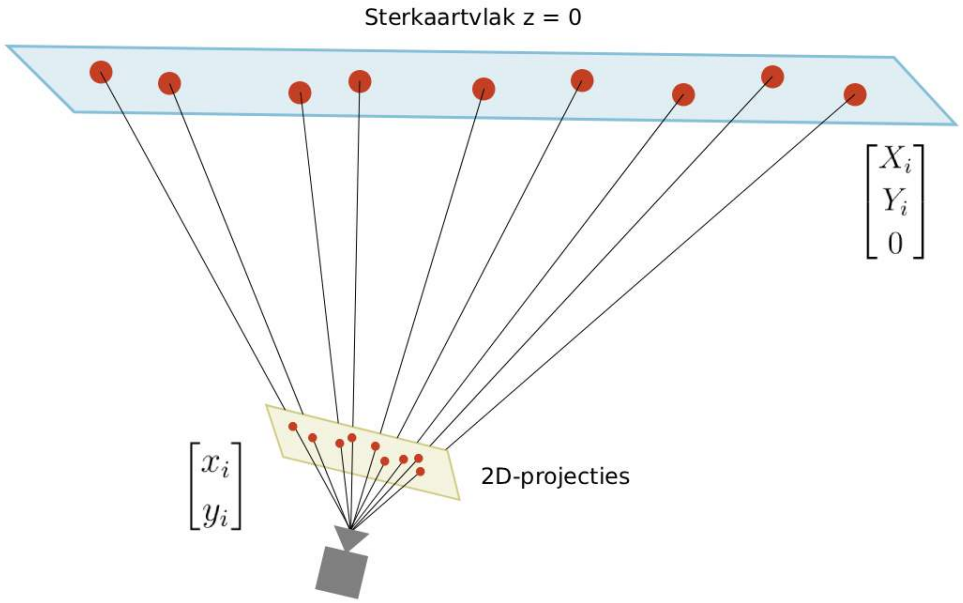
Minstens vier corresponderende punten zijn nodig voor de PnP-stap.

4.5.3 PnP-RANSAC

De globale pose wordt berekend met `cv::solvePnPRansac`. PnP, voluit *Perspective-n-Point*, bepaalt de camerapose uit correspondenties tussen 3D-objectpunten en hun 2D-projecties. In deze toepassing zijn de 3D-objectpunten de markerposities p_i^W uit de sterkaart en zijn de 2D-punten de gedetecteerde centroïden (x_i, y_i) in het undistorted beeld, dit is geïllustreerd in figuur 4.8. PnP zoekt dus de rotatie R_{CW} en translatie t_{CW} die wereldpunten naar het camerastelsel brengen en vervolgens via de intrinsieke cameramatrix K op het beeld projecteren:

$$w_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = K \begin{bmatrix} R_{CW} & t_{CW} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{bmatrix}. \tag{4.10}$$

Omdat de beeldpunten al uit het undistorted beeld komen, gebruikt de PnP-stap een nulvector voor de distortiecoëfficiënten. De projectie vergelijkt de voorspelde beeldpunten dus rechtstreeks met de gedetecteerde centroïden.



Figuur 4.8: PnP-posebepaling uit 3D-wereldpunten in het sterkaartvlak en hun 2D-projecties in het camerabeeld. De rotatie en translatie van de camera worden gezocht zodat de geprojecteerde kaartpunten overeenkomen met de gedetecteerde beeldpunten.

Binnen OpenCV worden de posehypothesen opgelost met `SOLVEPNP_SQPNP`. SQPnP wordt hier gebruikt omdat de markerpunten in de plafondkaart coplanair zijn en omdat globale localisatie

geen initiële pose heeft. In tegenstelling tot EPnP, dat gevoeliger is voor ruis en lokale minima bij zulke configuraties, is SQPnP ontworpen als globale PnP-oplosser voor willekeurige puntconfiguraties, inclusief coplanaire punten [35].

De solver levert een directe PnP-oplossing, maar de kandidaatset uit de vorige stap kan nog foutieve paren bevatten. RANSAC kiest daarom herhaaldelijk een kleine subset, berekent een posehypothese en projecteert daarna alle kandidaatpunten opnieuw in het beeld. Paren waarvan de projectie binnen het ingestelde venster (5 px) van het gemeten beeldpunt valt, worden inliers. Deze 5 px-drempel is dus de consensusdrempel binnen RANSAC; dit geldt niet tijdens tracking en daarom is de latere 8 px-grens is een algemene geldigheidscontrole op de uiteindelijke posemeting. De hypothese met de beste inlierset wordt behouden.

De gevonden inliers worden daarna met Levenberg-Marquardt verfijnd, dit is een iteratieve niet-lineaire optimalisatie. In deze stap blijven alleen de inliercorrespondenties over en worden R_{CW} en t_{CW} verder aangepast om de totale reprojectiefout te minimaliseren.

4.5.4 Plausibiliteitscontrole

Na PnP voert `Localiser3D` nog twee controles uit. Eerst moet `PoseEstimator` de meting als geldig markeren: de gemiddelde reprojectiefout moet kleiner zijn dan 8 pixels en er moeten minstens drie inliers overblijven. Daarna wordt de geschatte camerapositie getoetst aan de kaartgrenzen. Ze moet binnen 1,0 m rond de bekende kaart liggen. Dit voorkomt dat een mathematisch geldige maar fysisch onwaarschijnlijke RANSAC-oplossing de tracking start.

Bij succes wordt de pose gepubliceerd als `CameraPoseMeasurement`. Die bevat onder meer de geldige transformatie T_{WC}, het aantal detecties, het aantal inliers, de reprojectiefout en een *confidence*waarde. Daarna schakelt de systeemtoestand over naar trackingmodus.

4.6 Continue tracking

4.6.1 Projectie vanuit vorige pose

Tijdens tracking is er in normale omstandigheden een geldige voorgaande pose beschikbaar. `Tracker3D` gebruikt die pose om alle kaartmarkers opnieuw naar het inkomende beeld te projecteren:

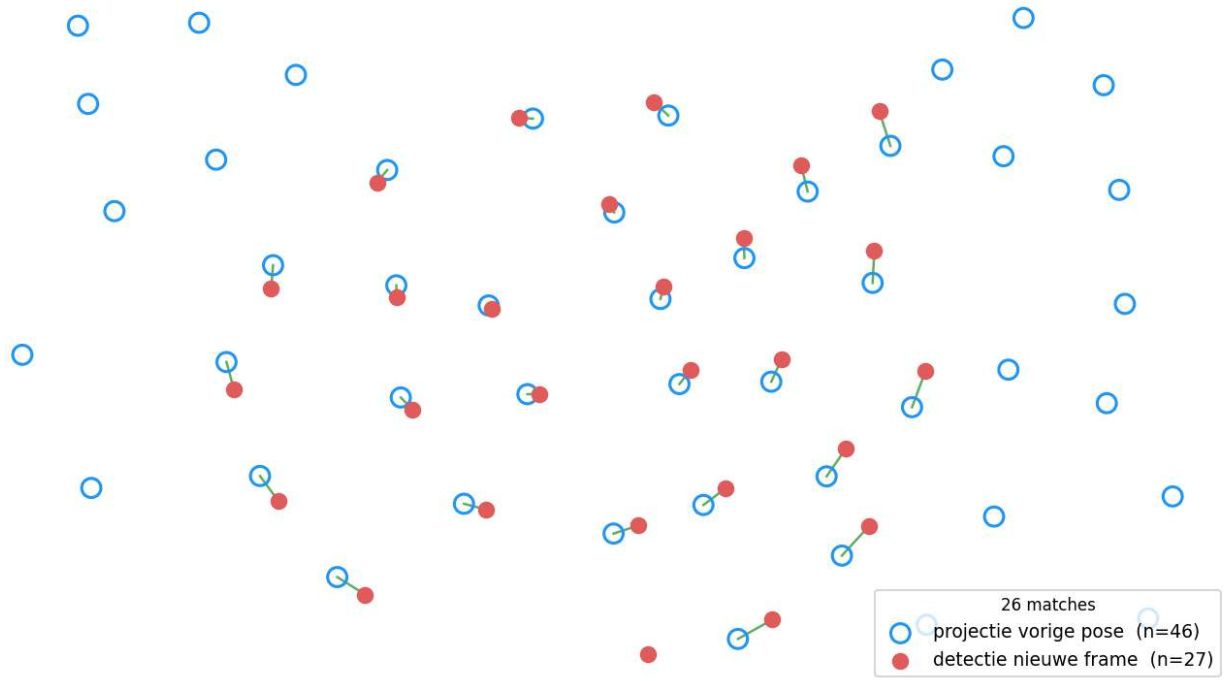
$$\hat{u}_j = \pi(K, T_{CW}^{prev}, p_j^W). \tag{4.11}$$

Markers waarvan de projectie meer dan 50 pixels buiten de beeldgrens valt, worden genegeerd. Voor de overige projecties wordt lokaal gezocht naar de dichtstbijzijnde detectie binnen de ingestelde maximale pixelafstand van 50 pixels. De code knipt geen afzonderlijke beeld-ROI uit voor detectie; de lokale beperking gebeurt in de matchingstap op basis van de gereprojecteerde kaartpunten. Wanneer minder dan het ingestelde minimumaantal matches overblijft, standaard vijf sterren met een absolute ondergrens van drie, wordt geen pose verfijnd en vraagt de tracker opnieuw globale localisatie aan.

4.6.2 Lokale nearest-neighbour matching

Voor elke geprojecteerde kaartmarker zoekt de tracker de dichtstbijzijnde detectie binnen een maximale pixelafstand. Een detectie mag slechts eenmaal gebruikt worden, zodat dezelfde blob niet aan meerdere kaartmarkers gekoppeld wordt. Hierdoor ontstaat een lokale set 2D-3D-correspondenties. Dit principe is te zien in figuur 4.9.

Deze aanpak is veel lichter dan globale tripletmatching, maar vereist een voldoende betrouwbare voorgaande pose. Wanneer te weinig matches gevonden worden, vraagt de tracker herlocalisatie aan. De systeemtoestand gaat dan terug naar localisatiemodus.



Figuur 4.9: Lokale nearest-neighbour matching tijdens tracking bij rotatie van de camera. De vorige pose projecteert kaartmarkers naar verwachte beeldposities; de dichtstbijzijnde actuele detectie binnen het zoekvenster wordt als kandidaatcorrespondentie gebruikt.

4.6.3 Poseverfijning

Voor de eigenlijke pose-update roept `Tracker3D` `PoseEstimator::refinePose` aan. Deze functie probeert eerst IPPE via `cv::solvePnPGeneric` met `SOLVEPNP_IPPE`. IPPE, voluit *Infinitesimal Plane-Based Pose Estimation*, is een PnP-variant die specifiek gebruikmaakt van coplanaire objectpunten. Dat past bij de plafondkaart, omdat alle markerpunten in hetzelfde vlak liggen. In tegenstelling tot een algemene PnP-solver behandelt IPPE de vlakke geometrie expliciet en geeft de methode de twee projectief mogelijke vlakposes terug.

Omdat IPPE twee oplossingen kan teruggeven, kiest de implementatie eerst fysisch geldige oplossingen waarbij de markers voor de camera liggen. Als meerdere oplossingen geldig zijn, wordt de oplossing gekozen waarvan de rotatie het dichtst bij de voorgaande pose ligt. Dat voorkomt dat de pose tijdens continue tracking naar de spiegeloplossing van het vlak springt.

Als IPPE niet bruikbaar is, gebruikt de code een iteratieve `solvePnP` met de voorgaande pose als initiële schatting. Daarna wordt de oplossing opnieuw verfijnd met Levenberg-Marquardt via `cv::solvePnPRefineLM`.

De trackingstap wordt pas aanvaard na twee bijkomende controles. Eerst moet de reprojectiefout onder de ingestelde drempel blijven. In de configuratie heet deze variabele `maxReprojM`; de naam duidt op de maximale toegelaten reprojectie-afwijking in meters. Voor de controle in beeldruimte zet de implementatie deze waarde om naar pixels met $\texttt{maxReprojM} \cdot f_x$. Daarna vergelijkt `Tracker3D` de rotatie met de vorige pose. Een rotatieverandering groter dan 10 graden per frame wordt als onwaarschijnlijk beschouwd en veroorzaakt herlocalisatie. Hierdoor worden vooral spiegeloplossingen of plots fout gekoppelde markerparen afgewezen.

4.7 PoseEstimator

De globale localisatie en de continue tracking gebruiken niet dezelfde invoer, maar ze eindigen wel allebei bij hetzelfde type posemeting. Daarom is de OpenCV-PnP-logica geconcentreerd in `PoseEstimator`. Deze klasse vormt de grens tussen de correspondenties uit de vorige stappen en de posemeting die later door de ESKF verwerkt wordt. De klasse biedt twee hoofdoperaties:

- `estimatePoseRansac` voor globale localisatie;
- `refinePose` voor tracking rond een vorige pose.

Beide operaties leveren een `CameraPoseMeasurement`. Deze meting bevat de pose T_{WC}, de tijdstempel, het aantal detecties, het aantal inliers, de gemiddelde reprojectiefout en een confidencewaarde. De meting krijgt alleen `valid=true` wanneer de gemiddelde reprojectiefout kleiner is dan 8 pixels en minstens drie inliers beschikbaar zijn.

De confidence combineert twee signalen. Het eerste signaal is de verhouding tussen inliers en detecties. Het tweede signaal is de reprojectiekwaliteit, genormaliseerd ten opzichte van de 8-pixelgrens. In vereenvoudigde vorm:

$$c = \frac{1}{2}\frac{N_{\text{inliers}}}{N_{\text{detecties}}} + \frac{1}{2}\left(1 - \min\left(\frac{e_{\text{reproj}}}{e_{\max}}, 1\right)\right). \tag{4.12}$$

Deze confidence is geen statistische waarschijnlijkheid, maar een praktische kwaliteitsmaat voor statusweergave, filtercorrectie en trackingbeslissingen.

Na de poseberekening wordt de meting omgezet naar een `PoseResult` voor de systeemstatus en de verdere publicatie. Daarbij wordt op basis van de matchrate en reprojectiefout een kwaliteitsvonnis toegekend. Er zijn vier categorieën: Correct (matchpercentage $\geq 70\,\%$ en reprojectiefout < 3 pixels), Probable ($\geq 50\,\%$ en < 5 pixels), Partial ($\geq 30\,\%$ en < 8 pixels) en Doubtful voor alle overige gevallen. Enkel poses met verdict Correct, Probable of Partial worden geaccepteerd door de *trackingworker*. Bij Doubtful wordt `pose.valid` op `false` gezet en wordt de meting niet gepubliceerd.

Rond de PnP-uitkomst worden nog aanvullende controles toegepast. De Z-controle interpreteert $-z$ als de afstand tot het plafond. Wanneer die afstand niet eindig is, kleiner dan of gelijk aan nul is, of groter is dan de geconfigureerde vloer-tot-plafondhoogte, wordt z teruggezet naar

de gekalibreerde plafondafstand en krijgt de pose een confidence van maximaal 0,69. Daarnaast wordt de horizontale positie getoetst aan de bekende kaartgrenzen. Bij globale localisatie gebruikt `Localiser3D` hiervoor een marge van 1,0 m; tijdens de verdere runtimeverwerking wordt een conservatievere marge van 0,75 m gebruikt rond het gekende kaartgebied.

4.8 IMU en Error-State Kalman Filter

De visuele pipeline levert absolute poses wanneer voldoende markers correct gedetecteerd en geïdentificeerd zijn. Die poses komen echter maar binnen op de camerafrequentie en kunnen tijdelijk ontbreken bij afdekking, bewegingsonscherpte of te weinig zichtbare sterren. De IMU vult die tussenliggende tijdstappen op. De ESKF combineert daarom een snelle inertiële propagatie met visuele correcties telkens wanneer een nieuwe betrouwbare camerapose beschikbaar is.

De eerste ESKF-initialisatie gebruikt een strengere cameragate dan de gewone posepublicatie. Naast `cam_snap.valid` vereist de implementatie minstens vier inliers en een reprojectiefout kleiner dan 5 pixels. Voor latere correcties volstaat een nieuwe visuele meting met `cam_snap.valid`. De meetruis van die correctie wordt wel opgeschaald wanneer de reprojectiefout boven 2,5 pixels ligt. Zo krijgt een pose met hogere reprojectiefout minder gewicht in de filtercorrectie, zonder dat ze onmiddellijk volledig verworpen wordt.

4.8.1 IMU-metingen

De BNO085 levert via I2C drie datatypes: een interne AHRS-quaternion (rotatiemeting), ruwe accelerometermetingen en gekalibreerde gyroscoopmetingen. De interne AHRS-quaternion combineert accelerometer, gyroscoop en magnetometer tot een absolute oriëntatieschatting inclusief magnetische *heading*. Tijdens de ontwikkeling bleek dat het gebruik van deze interne heading problemen veroorzaakte voor de stabiliteit van de uitvoerpose. Daardoor is de BNO085-quaternion niet als directe correctiemeting in de ESKF opgenomen.

De ESKF gebruikt uitsluitend de ruwe accelerometer- en gyroscoopmetingen voor de IMU-propagatie. Tussen twee opeenvolgende camerametingen draait de ESKF op 100 Hz als *dead-reckoning* systeem: de gyroscoop integreert de oriëntatieverandering en de accelerometer integreert de positieverandering. De attitudecorrectie en de positiecorrectie komen niet van de IMU, maar van de visuele camerapose.

Op basis van die keuze wordt de IMU niet als tweede absolute tracker behandeld. De inertiële metingen voorspellen hoe de pose tussen twee cameracorrecties verandert; de visuele meting blijft de bron die de absolute positie en heading opnieuw vastlegt.

4.8.2 Nominale toestand en fouttoestand

De ESKF gebruikt een nominale toestand met positie, snelheid, attitude en biastermen:

$$x = \{p, v, q, b_a, b_g\}. \tag{4.13}$$

De fouttoestand is 15-dimensionaal:

$$\delta x = \begin{bmatrix} \delta p \\ \delta v \\ \delta \theta \\ \delta b_a \\ \delta b_g \end{bmatrix}. \tag{4.14}$$

De nominale toestand bevat dus een quaternion voor de attitude, terwijl de fout op de attitude als kleine hoek $\delta\theta$ wordt voorgesteld. Dit is het basisprincipe van een Error-State Kalman Filter: de niet-lineaire toestand wordt rechtstreeks geïntegreerd, terwijl de filter kleine fouten rond die toestand schat.

4.8.3 Ruis- en kalibratieparameters

De ESKF bevat parameters voor accelerometerruis, gyroscoopruis en *bias random walk*. Deze waarden bepalen hoe sterk het filter de IMU-propagatie vertrouwt ten opzichte van visuele correctiemetingen. Voor het prototype is een IMU-ruisonderzoek uitgevoerd en zijn de resulterende ruiswaarden in de kalibratieconfiguratie opgenomen. De implementatie voorziet daarnaast instelbare sigma's voor attitudecorrectie en camerapositiecorrectie, zodat het filtergedrag verder kan worden afgestemd.

Correcte sensorfusie vereist ook dat de ruimtelijke en temporele relatie tussen trackingcamera en IMU gekend is. Voor het prototype is een camera-IMU-kalibratie uitgevoerd met Kalibr [47]. De kalibratie levert de rotatie, translatie en tijdsverschuiving tussen beide sensoren. Die waarden worden gebruikt om de visuele camerapose naar het IMU-frame te brengen voordat de positiemeting in de ESKF wordt verwerkt.

4.8.4 IMU-propagatie

Met de ruis- en extrinsieke kalibratie vastgelegd kan de filterlus per IMU-tick uitgevoerd worden. De propagatiestap gebruikt alleen de inertiële metingen en loopt dus ook door wanneer er tijdelijk geen nieuwe visuele pose beschikbaar is.

Bij elke IMU-tick worden acceleratie en hoeksnelheid eerst gecorrigeerd met de geschatte bias:

$$a = a_m - b_a, \qquad \omega = \omega_m - b_g. \tag{4.15}$$

De positie en snelheid worden gepropageerd met

$$p_{k+1} = p_k + v_k \Delta t + \frac{1}{2}(R(q_k)a + g)\Delta t^2, \tag{4.16}$$

$$v_{k+1} = v_k + (R(q_k)a + g)\Delta t. \tag{4.17}$$

De attitude wordt gepropageerd door de gyroscoopmeting als kleine quaternionrotatie te integreren. Tegelijk wordt de covariantie van de fouttoestand gepropageerd met een eerste-orde

overgangsmatrix en de ruisparameters uit de kalibratie.

Naast de IMU-propagatie ontvangt de ESKF correctiemetingen telkens wanneer een nieuwe geldige visuele pose beschikbaar is. Die visuele pose levert twee correcties tegelijk. De positiecomponent corrigeert de cumulatieve positiedrift. De camera-afgeleide oriëntatie, verkregen door de visuele camerarotatie samen te stellen met de camera-IMU-extrinsieken, corrigeert de geaccumuleerde gyroscopische oriëntatiefout inclusief heading. Na elke correctiestap wordt de fouttoestand opnieuw rond nul gecentreerd en geïnjecteerd in de nominale toestand.

4.9 Outputpose en FreeD

4.9.1 Samenstellen van de uitvoerpose

Na de ESKF-stap wordt de uitvoerpose samengesteld. Wanneer de ESKF geïnitialiseerd is, wordt eerst de IMU-positie terug naar de camerapositie omgerekend met de camera-IMU-extrinsieken. De interne wereldpose behoudt daarbij de conventie dat het plafondvlak op $z = 0$ ligt en de camera eronder een negatieve z-waarde heeft. In de uitvoerlaag wordt de oorsprong verschoven van het plafond naar de vloer. Met de geconfigureerde vloer-tot-plafondafstand h_{ceiling} wordt de hoogte boven de vloer berekend als

$$z_{\text{out}} = z_{\text{tracker}} + h_{\text{ceiling}}. \tag{4.18}$$

Zolang de ESKF niet geïnitialiseerd is, valt de uitvoer terug op de laatst beschikbare visuele pose. De confidence in shared memory geeft aan of de gefilterde pose actief is.

4.9.2 Nabewerking van de uitvoerpose

Na het samenstellen van de uitvoerpose wordt nog een *One Euro Filter* toegepast op de zes uitvoerkanalen: positie x, y, z en oriëntatie roll, pitch en yaw. De filter is een snelheidsafhankelijke laagdoorlaatfilter: bij trage of stilstaande bewegingen wordt sterker afgevlakt om jitter te verminderen, terwijl de afsnijfrequentie bij snellere bewegingen verhoogt om bijkomende vertraging te beperken [48].

In de implementatie worden de filterparameters `smooth_min_cutoff` en `smooth_beta` via de configuratie ingesteld. Wanneer `smooth_min_cutoff` nul is, wordt de filter overgeslagen. De afvlakking wordt toegepast nadat de ESKF- of terugvalpose naar de uitvoerconventie is gebracht en voordat de pose verder naar FreeD en shared memory wordt gepubliceerd.

4.9.3 Oriëntatieconventie

De trackingcamera is naar boven gericht, terwijl een studiocamera naar voren kijkt. Dit 90° verschil in montagepositie heeft gevolgen voor de betekenis van de ZYX-Eulerhoeken uit het interne cameraframe. De uitvoerlaag brengt een expliciete assenmapping aan zodat alle ontvangende systemen de oriëntatie in het render-cameraframe ontvangen.

De interne camerarotatie levert drie hoeken op via ZYX-decompositie. Yaw (ψ) beschrijft rotatie rond de opwaartse as van de camera en stemt overeen met de pan-beweging van de cameramast. FreeD gebruikt een CW-positieve conventie, wat een tekenomkering vereist:

$$\text{pan}_{\text{out}} = -\psi. \tag{4.19}$$

Omdat de camera naar boven kijkt, correspondeert de interne roll (ϕ) met een fysische voorwaartse of achterwaartse kanteling van de camera, het equivalent van FreeD-tilt. De interne pitch (θ) correspondeert met een zijdelingse kanteling, het equivalent van FreeD-roll:

$$\text{tilt}_{\text{out}} = \phi_{\text{cam}}, \tag{4.20}$$

$$\text{roll}_{\text{out}} = \theta_{\text{cam}}. \tag{4.21}$$

Deze omzetting is expliciet in de uitvoerlaag geconcentreerd zodat localisatie, tracking en ESKF intern consistent blijven en de render-cameraconventie enkel op het eindpunt wordt toegepast.

4.9.4 FreeD D1-codering

De finale pose wordt als FreeD D1-pakket via UDP verstuurd. Het pakket bevat pan, tilt, roll en de drie positiecomponenten. Hoeken worden in graden vermenigvuldigd met 32768 en als *signed* 24-bit *big-endian integers* opgeslagen:

$$\theta_{\text{enc}} = \text{round}(\theta_{\text{deg}} \cdot 32768). \tag{4.22}$$

Posities worden van meter naar millimeter omgezet en daarna met factor 64 geschaald:

$$p_{\text{enc}} = \text{round}(p_m \cdot 1000 \cdot 64). \tag{4.23}$$

Zoom en focus worden als nulwaarden ingevuld. De FreeD-doelhost en poort kunnen via de configuratie of frontendinstellingen aangepast worden.

Hoofdstuk 5

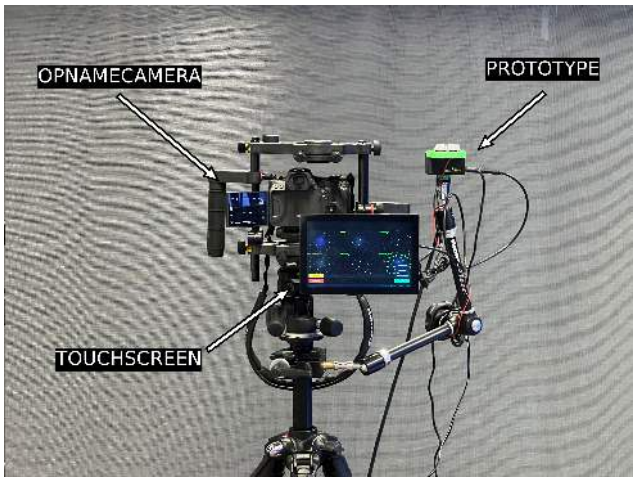
Opstelling en evaluatie

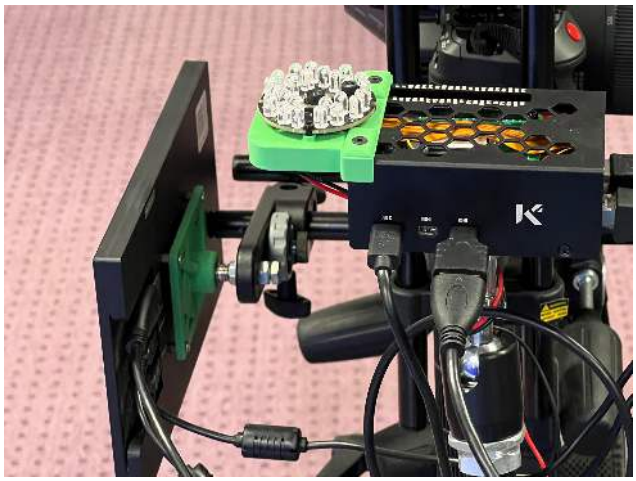
Dit hoofdstuk beschrijft hoe het prototype in een functionele toestand wordt gebracht en hoe de evaluatie wordt opgebouwd. Eerst wordt de gemeenschappelijke voorbereiding beschreven. Vervolgens worden de afzonderlijke evaluaties afgebakend: de kwaliteit van de sterkaart, een detectiestabiliteits- en localisatiebenchmark, de posevergelijking met een Vive Tracker en de prestatie- en integratie-evaluatie via FreeD, Unreal Engine en de LED-wall.

5.1 Voorbereiding van het prototype

5.1.1 Trackingrig en markeromgeving

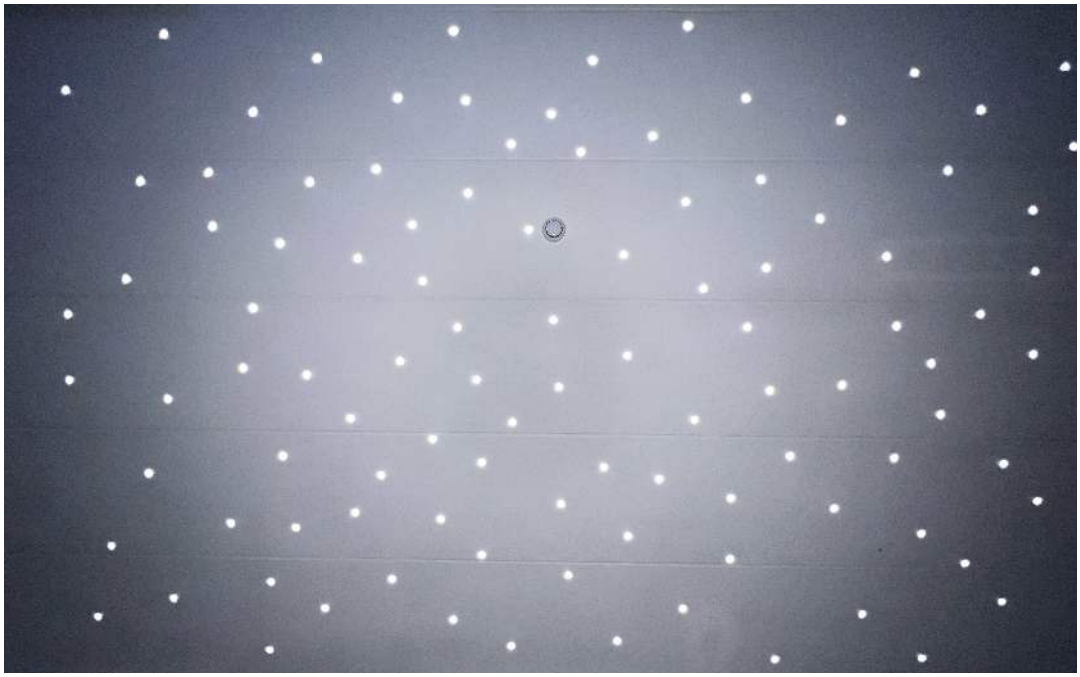
De fysieke basis van de evaluatie is een rig waarop de trackingmodule vast gemonteerd wordt, met de camera naar boven gericht. De trackingcamera bevindt zich op een afstand van circa 1,4 m van het plafond, zodat de retroreflectieve markers als sterren in beeld verschijnen. In de ruimte wordt bewust laag omgevingslicht voorzien voor optimale tracking, zoals ook in echte virtual-productionstudio's gebruikelijk is. De rig bestaat uit een statief dat op een verrijdbare basis met wieltjes geplaatst is. Daardoor kan de opstelling rond een vaste positie roteren en doorheen de ruimte verplaatst worden. Figuur 5.1 toont het prototype en touchscreen gemonteerd op de rig.





Figuur 5.1: Trackingrig met de opwaarts gerichte trackingmodule op een statief en verrijdbare basis. De verrijdbare basis bootst een eenvoudige camerabeweging op een dolly na tijdens de posevalidatie.

De markeromgeving blijft dezelfde voor de prestatie-evaluatie en de nauwkeurigheidsvalidatie. De retroreflectieve markers worden onregelmatig en voldoende gespreid aangebracht, zodat er binnen het meetgebied genoeg markers tegelijk zichtbaar zijn voor localisatie en continue tracking. Figuur 5.2 toont de retroreflectieve markers aan het plafond.



Figuur 5.2: Retroreflectieve markeromgeving aan het plafond van de evaluatieruimte. De markers vormen de vaste ruimtelijke referentie die zowel voor kaartopbouw als voor latere tracking gebruikt wordt.

5.1.2 Camerakalibratie en IMU-voorbereiding

Voor een functioneel prototype moet de trackingcamera intrinsiek gekalibreerd zijn. De cameramatrix en distortiecoëfficiënten worden gebruikt om beelden te corrigeren voordat markercentra voor identificatie, matching en poseberekening worden gebruikt. Zonder deze stap zou een gedetecteerd beeldpunt niet overeenkomen met de projectiestraal die in de poseberekening wordt verondersteld. Er werden geen fysieke infraroodfilters op de camera gemonteerd.

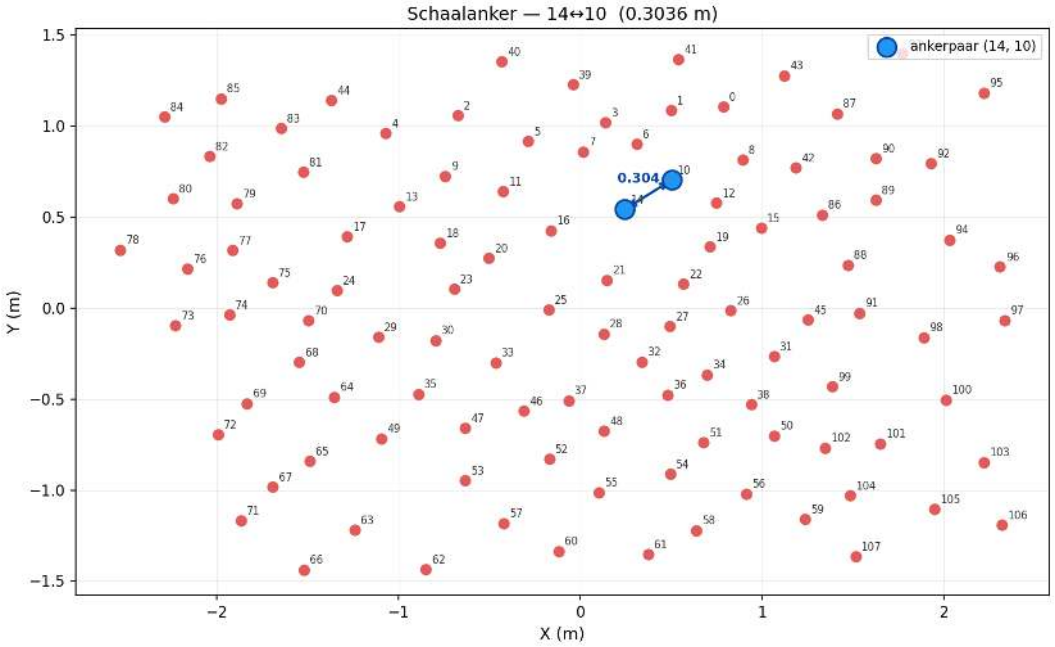
Ook de IMU wordt voorbereid voordat evaluaties uitgevoerd worden. De camera-IMU-kalibratie legt de vaste rotatie, translatie en tijdsoffset tussen beide sensoren vast. Het IMU-ruisonderzoek levert de parameters waarmee de filterlaag de inertiële metingen weegt.

5.1.3 Sterkaart, schaal en configuratie

Na de sensorkalibraties wordt een sterkaart van de markeromgeving opgebouwd. Hiervoor wordt de verrijdbare opstelling door het meetgebied bewogen totdat alle aangebrachte markers visueel geconfirmeerd zijn. De kaartopbouw levert daarna een eerste metrische markerkaart op met de geschatte posities van de retroreflectieve sterren.

Na de kaartopbouw wordt een ankerkalibratie uitgevoerd. Het systeem projecteert de opgebouwde sterkaart op het actuele camerabeeld, zodat de gebruiker twee herkenbare markers kan

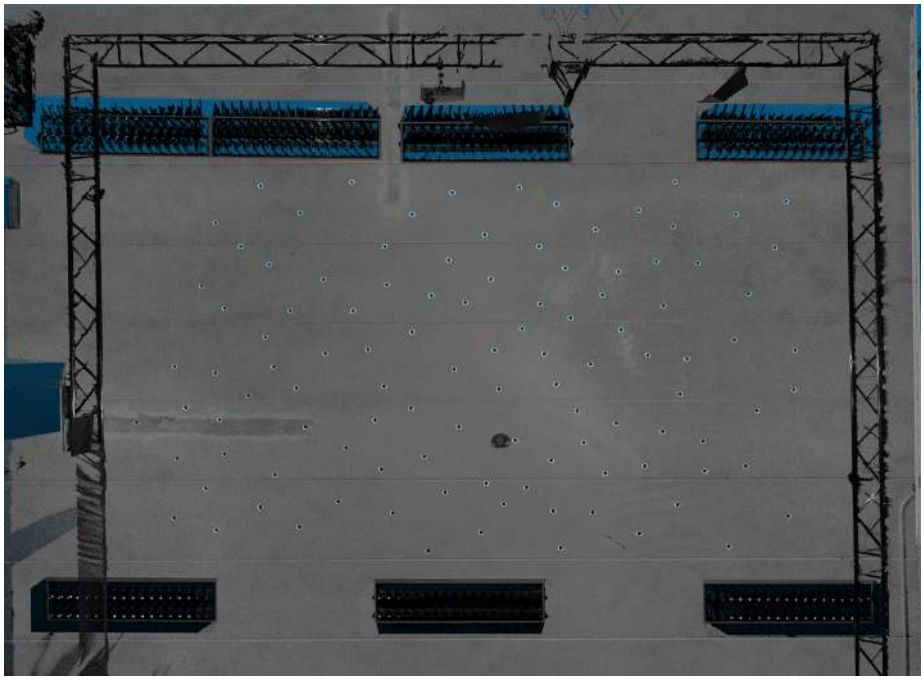
selecteren. De fysieke afstand tussen die twee markers wordt vervolgens opgemeten en als schaalanker ingevoerd. Op basis daarvan wordt de volledige sterkaart herschaald naar haar eindvorm. Deze stap is nodig omdat een kleine schaalfout in de kaart rechtstreeks doorwerkt in de berekende camerapositie. Figuur 5.3 toont de finale sterkaart na de schaalankerkalibratie.



Figuur 5.3: Schaalcorrectie op basis van twee fysiek opgemeten markers. Het systeem projecteert de opgebouwde sterkaart op het actuele camerabeeld, waarna de gebruiker twee herkenbare markers kiest en hun fysieke afstand invoert.

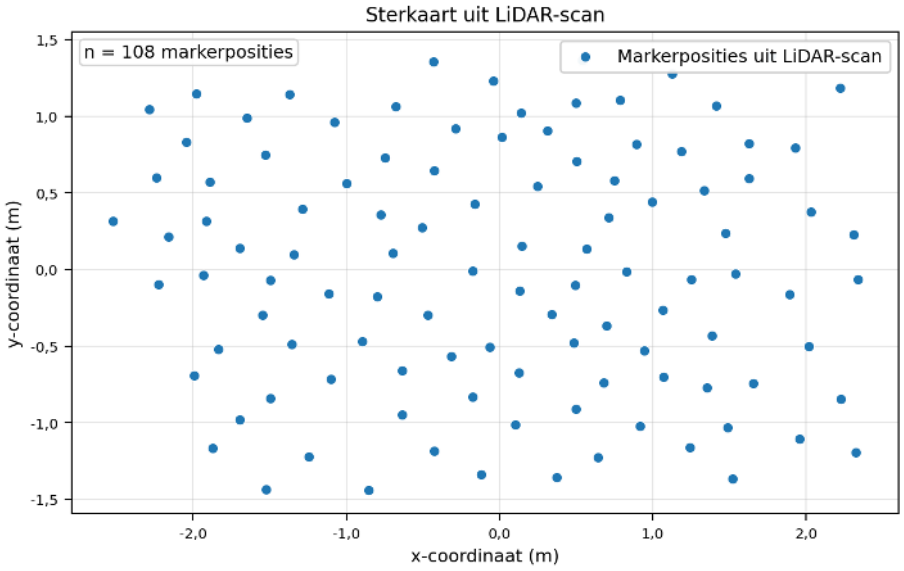
5.2 Sterkaartkwaliteit

De sterkaart vormt de ruimtelijke basis van de poseberekening. Om de kwaliteit daarvan afzonderlijk te beoordelen, worden de ingescande sterposities vergeleken met de ware posities van de fysieke markers. In deze evaluatie worden die posities gemeten met een LiDAR-scan. LiDAR staat voor *light detection and ranging*: een meetsysteem dat met laserpulsen afstanden tot de omgeving bepaalt en daaruit een puntenwolk opbouwt. De LiDAR-meting levert daardoor een onafhankelijke geometrische beschrijving van de markeromgeving, los van de beelden en poses waarmee het prototype zijn eigen kaart opbouwt. De centra van de retroreflectieve markers worden rechtstreeks uit de metrisch correcte puntenwolk geïdentificeerd en opgeslagen als een CSV-bestand dat als referentiekaart voor de vergelijking dient. Figuur 5.4 toont de ruwe LiDAR-scan van de markeromgeving.



Figuur 5.4: LiDAR-scan van de markeromgeving als onafhankelijke referentie voor markerposities. De scan levert een metrische puntenwolk waarmee de geometrische kwaliteit van de opgebouwde sterkaart wordt vergeleken.

Voor de vergelijking worden de LiDAR-markerposities en de opgebouwde sterkaart in hetzelfde coördinatenstelsel gebracht. Nadat overeenkomstige markers geïdentificeerd zijn, wordt per marker de ruimtelijke afwijking tussen de geschatte en gemeten positie bepaald. De evaluatie rapporteert de RMSE, mediaanafwijking en maximale afwijking van de markerfouten. Figuur 5.5 toont de LiDAR-gebaseerde referentiekaart van de markerposities.



Figuur 5.5: LiDAR-gebaseerde referentiekaart van de markerposities, afgeleid uit de metrisch correcte puntenwolk. Deze kaart wordt niet gebruikt door de tracker zelf, maar dient als onafhankelijke referentie voor de evaluatie.

5.3 Evaluatie van de beeldverwerking

Naast de sterkaart wordt de kwaliteit van de beeldverwerking afzonderlijk beoordeeld. Deze evaluaties gebruiken geen externe referentietracker, maar analyseren interne eigenschappen van het prototype: de stabiliteit van de gedetecteerde markercentra, het vermogen om zonder voorgaande pose te localiseren en de posejitter bij een stilstaande opstelling. Daardoor kan het gedrag van de detector, de localisatie en de gefuseerde uitvoer afzonderlijk geïnterpreteerd worden voordat de bewegingsvalidatie met de Vive Tracker wordt besproken.

5.3.1 Detectiestabiliteit

De detectiestabiliteit wordt gemeten op beelden waarbij de fysieke opstelling niet beweegt. Per frame worden de markercentra gedetecteerd en via *nearest neighbour* gekoppeld aan de referentieset uit het eerste frame. Voor elke gevolgde marker wordt de standaardafwijking van de pixelpositie bepaald; de gerapporteerde centroïde-*jitter* is het gemiddelde daarvan over alle gevolgde markers.

De lichte en zware detector worden in dezelfde gecontroleerde trackingomgeving vergeleken. Voor beide detectoren worden de gemiddelde detectietijd, minimale en maximale detectietijd, het gemiddelde aantal detecties en de centroïde-*jitter* per as gerapporteerd. Daarnaast worden beide detectoren ook getest op afzonderlijke beelden met storende omgevingsverlichting. Omdat die beelden geen continue stilstaande sessie vormen, wordt daar geen centroïde-*jitter* berekend; die vergelijking beperkt zich tot detectietijd en aantal gedetecteerde markers.

5.3.2 Localisatiebenchmark

De localisatiebenchmark beoordeelt of het prototype een camerapose kan vinden wanneer er nog geen betrouwbare voorgaande pose beschikbaar is. De test vertrekt vanuit globale identificatie van markerpatronen in het actuele beeld. Dit maakt de benchmark geschikt om de robuustheid van de initiële posebepaling en herlocalisatie te beoordelen.

Voor elk benchmarkbeeld worden de detecties, de gevonden inliers, de rerprojectiefout en de rekentijd van detectie en poseoplossing bijgehouden. De belangrijkste beoordelingsmaten zijn het succespercentage, het gemiddelde aantal detecties, het gemiddelde aantal inliers, de gemiddelde en mediane herprojectiefout en de rekentijd van detector en solver. De benchmark gebruikt de lichte detector met de operationele parameters van het prototype, omdat die detector bedoeld is voor continue pose-uitvoer.

5.3.3 Stabiliteit bij stilstand

De stilstandsevaluatie beoordeelt de posejitter van het prototype wanneer de rig fysiek stilstaat. Hiervoor worden zowel de visuele cameraposes als de uiteindelijke gefilterde uitvoerpose uit dezelfde sessie gebruikt. Die uitvoerpose bestaat uit ESKF-fusie gevolgd door de *One Euro*-afvlakking in de uitvoerlaag. De initiële localisatiepose wordt buiten beschouwing gelaten; voor de visuele analyse worden alleen geldige trackingposes gebruikt en voor de uitvoeranalyse alleen poses nadat de filter geïnitialiseerd is.

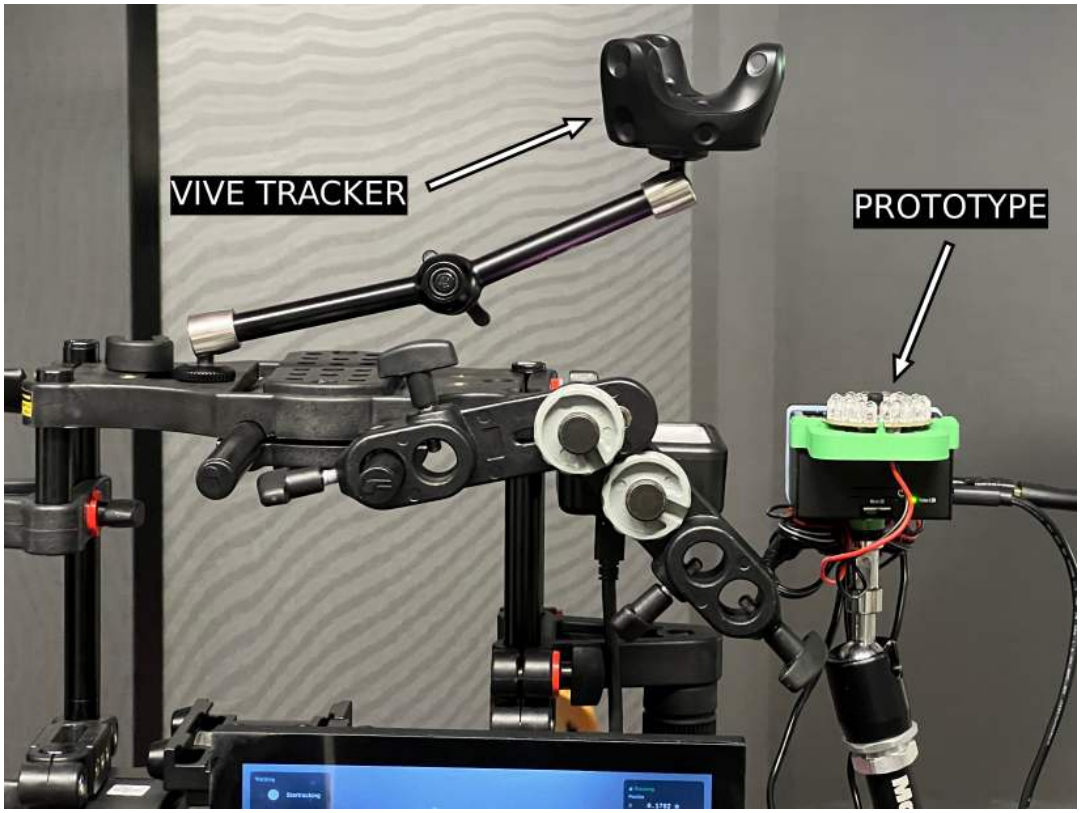
De positie- en oriëntatieafwijkingen worden berekend ten opzichte van de gemiddelde pose over de sessie. De evaluatie rapporteert per as en voor de 3D-positie de standaardafwijking, het 95e

percentiel en de maximale afwijking. Voor de oriëntatie worden roll-, pitch- en yawafwijkingen op dezelfde manier samengevat. Daarnaast worden het aantal inliers en de herprojectiefout van de visuele poses gebruikt als interne kwaliteitsmaten voor de poseberekening tijdens stilstand.

5.4 Nauwkeurigheidsvalidatie met Vive Tracker

De pose-uitvoer in beweging wordt gevalideerd door een Vive Tracker op dezelfde opstelling als het prototype bevestigd. Deze configuratie gebruikt geen LED-wall en geen opnamecamera als referentie. Het doel is uitsluitend om de pose-uitvoer van het ontwikkelde systeem te vergelijken met een onafhankelijk trackingsysteem tijdens dezelfde fysieke beweging. Omdat de Vive Tracker zelf ook meetonnauwkeurigheid heeft, levert deze evaluatie een relatieve overeenkomst tussen beide systemen op en geen absolute prototypefout ten opzichte van de ware positie.

Figuur 5.6 toont de Vive Tracker gemonteerd op de opstelling naast het prototype.



Figuur 5.6: Vive Tracker gemonteerd op de rig naast het prototype voor de nauwkeurigheidsvalidatie. De twee trackers meten niet hetzelfde fysieke punt, waardoor later een leverarm- en coördinatenkalibratie nodig is.

De vergelijking is niet rechtstreeks mogelijk op basis van de ruwe logs. De Vive Tracker en het prototype meten niet exact hetzelfde punt op de opstelling. Daarnaast hebben beide systemen een tijdsoffset en gebruiken ze een ander coördinatenstelsel. De Vive Tracker rapporteert poses in de SteamVR-ruimte, terwijl het prototype poses rapporteert in het wereldkader van de sterkaart en de renderomgeving. Voor een correcte vergelijking moeten deze verschillen eerst worden verwijderd.

5.4.1 Consistentie van de testomgeving

Voor de nauwkeurigheidsvalidatie wordt gebruik gemaakt van twee SteamVR-basisstations en één Vive Tracker. De twee basisstations worden in de evaluatieruimte geplaatst zodat de volledige bewegingszone van de rig zichtbaar is door beide stations. De Vive Tracker wordt naast het prototype op de rig gemonteerd zodat beide systemen tegelijk dezelfde fysieke bewegingen registreren.

De Vive Tracker stuurt zijn posegegevens draadloos via een USB-dongle naar de evaluatiecomputer. SteamVR beheert de tracker en de twee basisstations in een configuratie zonder vereiste VR-bril. De posegegevens worden vervolgens uit de SteamVR/OpenVR-laag gelogd op de eigen updatefrequentie van het systeem.

De omgevingsbelichting wordt laag gehouden, wat voor dit systeem ook de normale virtual-productieworkflow weerspiegelt. De basisstationposities, de markeromgeving, de belichting en de opstelling in de ruimte blijven ongewijzigd tussen kalibratieronden en validatietrajecten, zodat verschillen in resultaten niet veroorzaakt worden door wisselende meetomstandigheden.

5.4.2 Kalibratie van de trackervergelijking

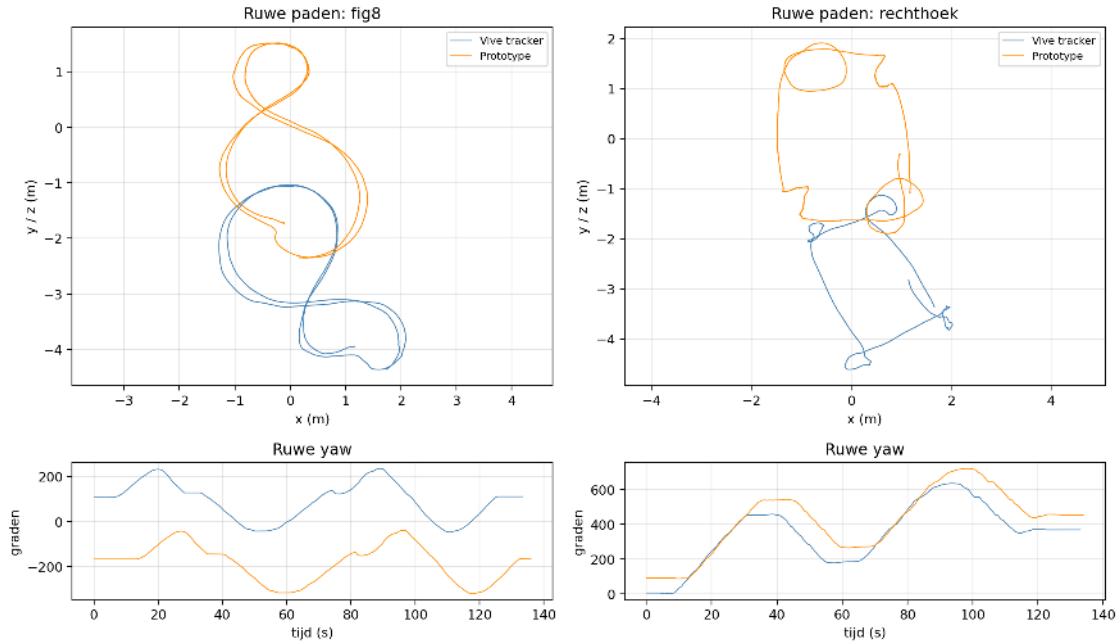
Voor elke installatie van de Vive Tracker op de opstelling wordt een afzonderlijke kalibratie uitgevoerd. Een installatie is een concrete fysieke bevestiging van de Vive Tracker, telkens met een andere afstand en positie ten opzichte van het prototype. De kalibratie bestaat uit drie delen: het bepalen van de vaste geometrische relatie tussen Vive Tracker en prototypesensorpunt, het schatten van de tijdsoffset per registratiepad en het bepalen van de planaire registratie tussen beide coördinatenstelsels. Het yaw-verschil tussen beide sensoren wordt in deze sectie genoteerd als γ. De planaire rotatie die SteamVR omzet naar het evaluatiekader van het prototype wordt genoteerd als Y_R, en de bijbehorende planaire translatie als Y_t.

Vaste installatieparameters

De vaste geometrische relatie tussen Vive Tracker en prototype wordt per installatie bepaald terwijl de opstelling stilstaat. Eerst wordt de Vive Tracker drie keer manueel op het centrum van het prototypesensorframe gelegd. Het gemiddelde van de drie gemeten Vive-posities wordt als referentiepunt van het prototypesensorpunt in het Vive-kader beschouwd. Daarna wordt de Vive Tracker in zijn definitieve houder bevestigd. De positie en yaw van die mount ten opzichte van het prototype blijven voor de volledige kalibratie en validatie van die installatie ongewijzigd. Het verschil tussen de referentiepositie en de gemonteerde Vive-positie levert de verbindingsvector op tussen de Vive Tracker en het prototypesensorpunt.

Voor het bepalen van de yaw-offset γ worden vier koersen van de rig vastgelegd in zowel het prototype- als het Vive-coördinatenstelsel. Het absolute heading-verschil over die vier posities levert γ op.

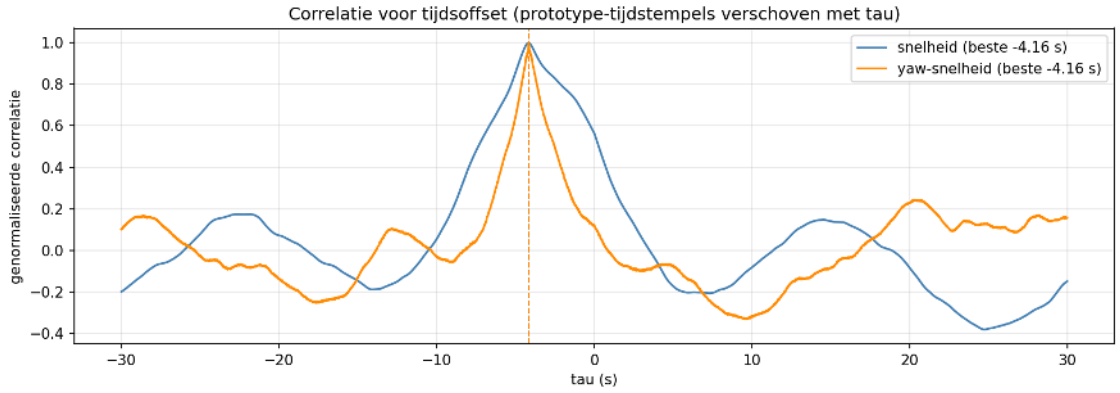
Om Y_R en Y_t te kunnen schatten worden vier registratiepaden afgelegd, elk met een eigen ruimtelijke vorm die per installatie herhaald wordt: een willekeurig pad, een dubbele diagonaal, een rechthoek met hoekrotaties en een dubbele achtvorm. De variatie in padvormen en yaw-verloop zorgt ervoor dat de planaire registratie voldoende bepaalbaar is. Figuur 5.7 toont de ruwe paden en het bijbehorende yaw-verloop van een installatie.



Figuur 5.7: Ruwe paden en yaw-verloop van de achtvorm- en rechthoekregistratiepaden voor configuratie C, weergegeven in de ongeregistreerde SteamVR-coördinaten.

Tijdsynchronisatie

Omdat de Vive Tracker en het prototype elk een eigen logfrequentie en startmoment hebben, wordt per registratiepad een tijdsoffset geschat. De hoofdschatter is de kruiscorrelatie van de yaw-profielen van beide posestromen. De snelheidsprofielen worden enkel als kruiscontrole gebruikt. Het verschil tussen beide offsetschattingen geeft een kwaliteitsmaat voor de synchronisatie. Figuur 5.8 illustreert deze stap voor een kalibratiepad.

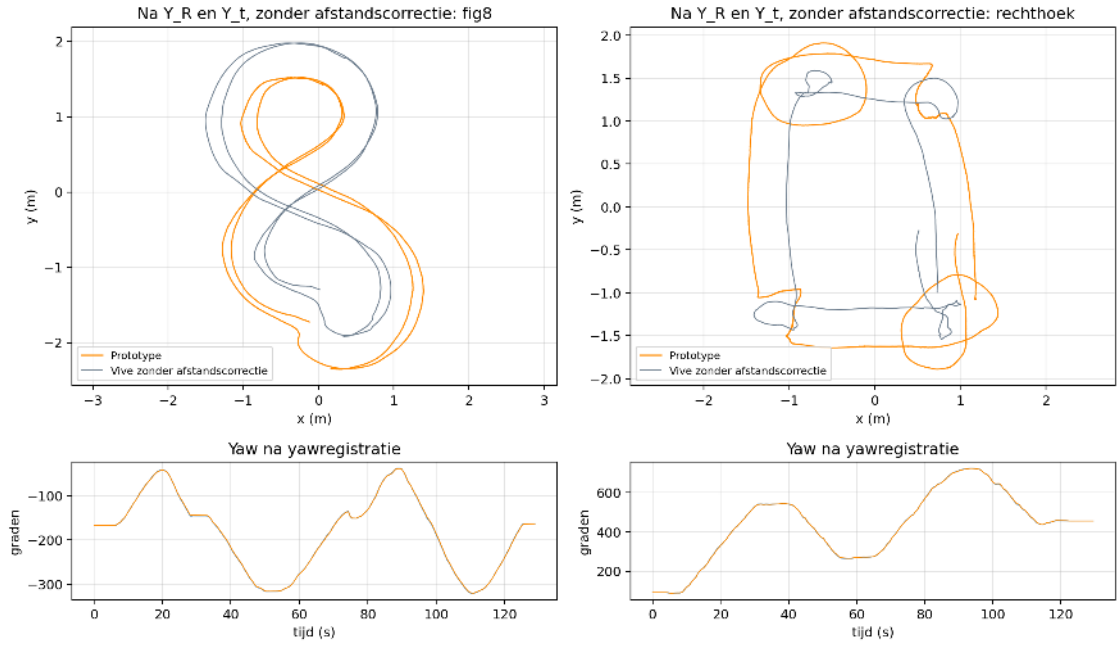


Figuur 5.8: Genormaliseerde correlatie van yaw- en snelheidsprofielen voor het willekeurige kalibratiepad; de yaw-correlatie is de hoofdschatter en de snelheids-correlatie dient als kruiscontrole.

Schatting van Y_R en Y_t

De effectieve verwerkingsvolgorde is als volgt: eerst wordt de tijdsoffset per pad bepaald, daarna wordt de opgemeten verbindingsvector toegepast zodat de Vive-positie naar het prototypesensorpunt wordt omgerekend, en pas daarna worden alle registratiepaden samen gebruikt om Y_R en Y_t

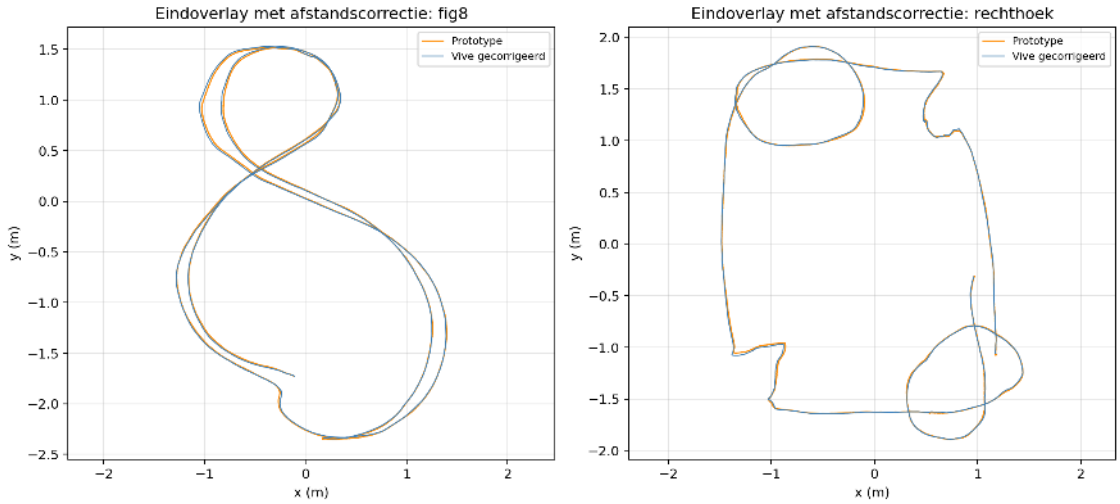
te bepalen. Figuur 5.9 wordt toch behouden omdat die visueel duidelijk maakt welk effect de verbindingsvector afzonderlijk heeft: wanneer alleen de planaire registratie zichtbaar gemaakt wordt en de verbindingsvectorcorrectie nog niet is toegepast, lopen het yaw-verloop en de trajectvorm al gelijk, maar blijft een systematische positieverschuiving zichtbaar.



Figuur 5.9: Trajectoverlays nadat Y_R en Y_t zijn toegepast maar vóór verbindingsvectorcorrectie voor configuratie C.

Na toepassing van alle kalibratieparameters vallen de gecorrigeerde Vive-trajecten en de prototypetrajecten samen in één gemeenschappelijk kader. Figuur 5.10 toont de eindoverlays na volledige kalibratie.

Deze kalibratiemethode is aanvaardbaar ondanks de manuele stap in de bepaling van de verbindingsvector. Per installatie worden drie afzonderlijke referentiemetingen uitgevoerd die gecontrolleerd worden op consistentie, en de volledige procedure wordt herhaald over drie installaties. Bovendien blijft de spreiding van de finaal bepaalde Y_R en Y_t consistent over alle installaties heen, wat aantoont dat de coördinatenregistratie stabiel en reproduceerbaar is: Y_R varieert slechts over ongeveer 0,33°, terwijl de componenten van Y_t binnen ongeveer 1 tot 2 cm blijven. Dat is voldoende klein om de methode als consistent voor deze evaluatie te beschouwen.



Figuur 5.10: Eindoverlays met afstandscorrectie voor de achtvorm- en rechthoekregistratiepaden van configuratie C; prototype en gecorrigeerde Vive-trajecten vallen samen.

5.4.3 Validatie en meetmethode

Een validatietraject verschilt fundamenteel van een kalibratiepad. Tijdens de kalibratie worden specifieke padvormen afgelegd zodat alle onbekende transformatieparameters bepaalbaar zijn. Validatietrajecten zijn vrije bewegingen door de ruimte, afgelegd nadat de kalibratieparameters volledig zijn vastgesteld. De verbindingsvector, γ, Y_R en Y_t worden voor een validatietraject niet opnieuw geschat; alleen de tijdsoffset wordt per traject opnieuw bepaald, omdat elk traject een eigen startmoment en bewegingsinhoud heeft.

De strikte scheiding tussen kalibratie en validatie is essentieel voor een zinvolle foutmeting. Als de transformatieparameters per traject opnieuw zouden worden gefit, zouden prototypeonnauwkeurigheden en Vive-meetruis beide worden geabsorbeerd in de parameteroplossing. De gerapporteerde fout zou dan een artefact zijn van de herhaalde optimalisatie en niet langer de werkelijke nauwkeurigheid van het prototype weerspiegelen. Door de parameters na de kalibratie vast te houden worden de validatieresultaten een onafhankelijke maat voor de prestatie van het systeem.

Het prototype is in staat om in alle zes vrijheidsgraden te tracken, echter wordt er tijdens de validatietrajecten enkel translatie in het horizontale vlak en yaw-rotatie geëvalueerd. Systematische roll- en pitchbewegingen worden niet opgenomen vanwege twee beperkingen: de verrijdbare validatieopstelling laat gecontroleerde kantelingen van de rig niet gemakkelijk toe, en bij scherpe kantelhoeken neemt de detectiestabiliteit sterk af. Door de lage plafondafstand en de beperkte vermogensterkte van de IR-bellichting wordt het teruggekaatste licht onder een scherpe hoek niet efficiënt terug naar de sensor geleid, waardoor de sterdetectie snel degradeert.

Er worden twee categorieën validatietrajecten onderscheiden: normale trajecten en snelle trajecten. Normale trajecten worden uitgevoerd bij gangbare rijsnelheden tot circa 25 cm/s. Die grens volgt uit de combinatie van het zoekvenster van de *nearest-neighbour*-tracker, de plafondafstand en de camera-eigenschappen. Snelle trajecten zijn stresstests die bewust boven deze operationele grens worden uitgevoerd om het gedrag bij hoge snelheden in kaart te brengen.

5.4.4 Beoordelingsmaten

De evaluatiepipeline rapporteert meerdere complementaire foutmaten. De primaire maat is de relatieve positieafwijking: de afstand tussen de geschatte prototypeoorsprong op basis van de gecorrigeerde Vive-meting en de directe pose-uitvoer van het prototype. Naast de gemiddelde positieafwijking wordt het 95e percentiel gerapporteerd om de foutstaart te karakteriseren. De RMSE is beschikbaar als aanvullende samenvatting per traject.

De yaw-afwijking wordt afzonderlijk gerapporteerd als rotatiefout. Daarnaast worden normale en snelle trajecten apart geïnterpreteerd, zodat het gedrag binnen het beoogde werkgebied gescheiden blijft van stressproeven boven de operationele snelheidsgrens. Voor één representatief traject worden de *absolute pose error* (APE), de *relative pose error* (RPE), het tijdverloop van de positie- en yawafwijking en de foutverdeling weergegeven.

De resultaten worden geaggregeerd per traject, per configuratie en per snelheidsklasse, en ook over alle configuraties heen vergeleken. Doordat de afstand tussen Vive Tracker en prototypesensor per installatie verschilt, varieert de gevoeligheid voor yaw-fouten: bij grotere afstand veroorzaakt een kleine yaw-fout een grotere positieverplaatsing aan de prototypepositie. Tegelijk geldt dat bij een kleinere armlengte de fysieke meting van de verbindingsvector relatief onnauwkeuriger wordt ten opzichte van de totale afstand, waardoor de onzekerheid in de yaw-offset toeneemt.

5.4.5 Nauwkeurigheid van de Vive Tracker

Bij de interpretatie van de resultaten moet rekening worden gehouden met de eigen meetonnauwkeurigheid van de Vive Tracker. Deze is in de literatuur onderzocht als referentiesysteem en heeft daarin een posefout die afhangt van de beweegsnelheid en de zichtlijn naar de basisstations. Kuhlmann de Canaviri et al. [49] onderzochten de statische en dynamische nauwkeurigheid van de VIVE Tracker 3.0 via een robotarm bij testsnelheden van 12,5, 25 en 50 mm/s. Met twee basisstations, dezelfde configuratie als in dit onderzoek, bedroeg de RMSE 2,8 mm bij 12,5 mm/s, 8,7 mm bij 25 mm/s en 11,1 mm bij 50 mm/s. Merker et al. [50] bevestigen eenzelfde snelheidsafhankelijke degradatie voor fietsbewegingen met vier basisstations: de gemiddelde positieafwijking neemt toe van 10,4 mm bij 80 omwentelingen per minuut tot 21,0 mm bij 160 omwentelingen per minuut.

De validatietrajecten in dit onderzoek worden niet uitgevoerd bij 50 mm/s maar bij aanzienlijk hogere snelheden. Tijdens normale trajecten beweegt de opstelling met snelheden van typisch 10 tot 25 cm/s, een factor twee tot vijf hoger dan de maximale testsnelheid uit het eerstgenoemde onderzoek. Tijdens snelle trajecten liggen de snelheden nog hoger. De Vive-meetruis is bij die snelheden bijgevolg niet te verwaarlozen. De gerapporteerde afwijkingen zijn daardoor een relatieve inter-systeemfout en geen harde bovengrens op de absolute prototypefout alleen. Vergelijkingen tussen configuraties en snelheidsklassen blijven methodologisch geldig omdat dezelfde Vive-opstelling en evaluatiemethode voor alle metingen worden gebruikt.

5.5 Prestatie- en integratie-evaluatie

Deze evaluatie onderzoekt of het prototype zijn pose snel en consistent genoeg kan aanbieden in een virtual-production-omgeving. De nadruk ligt op updatefrequentie, latency, stabiliteit van

de uitvoerketen en de zichtbare werking in Unreal Engine. Dit onderdeel is dus geen primaire meting van absolute pose-nauwkeurigheid.

5.5.1 Opstelling

De pose-uitvoer van het prototype wordt via FreeD over een ethernetkabel naar de rendering computer gestuurd. Unreal Engine gebruikt deze data om een scène te weergeven op de LED-wall. Voor de systematische posevalidatie wordt de rig op verrijdbare basis gebruikt. Voor vrije cameratracking, het opnemen van eindbeelden en de beoordeling van de integratie met Unreal Engine werd naast het statief ook een configuratie gebruikt waarbij het prototype rechtstreeks op de opnamecamera gemonteerd is.

De FreeD-uitvoer van het prototype wordt in Unreal Engine ontvangen via de Live Link FreeD-plug-in, die deel uitmaakt van de Virtual Production-toolset. Op basis van de ontvangen positie- en oriëntatiedata stuurt de plug-in de virtuele camera in de scène aan. Een *dedicated viewport* die vanuit het perspectief van die virtuele camera rendered zorgt ervoor dat het beeld op de LED-wall overeenkomt met de actuele camerahoek. Figuur 5.11 toont de opstelling met prototype, Unreal Engine en LED-wall.

Figuur 5.11: Opstelling voor de prestatie- en integratie-evaluatie met LED-wall en opnamecamera. Links wordt de statiefrig gebruikt; rechts is het prototype rechtstreeks op de opnamecamera gemonteerd voor vrije cameratracking en eindbeelden.

5.5.2 Methode

Tijdens deze evaluatie wordt het prototype in trackingmodus gebruikt terwijl de rig stilstaand en bewegend wordt getest. Per sessie worden twee CSV-bestanden gelogd. Het eerste bestand, `pose_output.csv`, bevat de uiteindelijke gefilterde uitvoerpose op circa 100 Hz: tijdstempel, trackingsstatus, positie en oriëntatie. Deze pose bevat dus zowel de ESKF-fusie als de *One Euro*-afvlakking. Het tweede bestand, `pose_vision.csv`, bevat de camerapose-updates: tijdstempel,

status, positie, oriëntatie, aantal gedetecteerde sterren, aantal inliers en herprojectiefout in pixels.

Naast de trajectgegevens worden per sessie ook prestatiestatistieken opgeslagen in een JSON-metadatabestand. Daarin zitten de gemiddelde en minimale updatefrequentie van de cameraposeupdates, de gemiddelde en p95-detectietijd per frame, de gemiddelde en p95-poseberekeningtijd, de gecombineerde pipelinetijd (detectie + poseberekening), de verwerkingsvertraging van het ESKF ten opzichte van het cameramoment en de softwarematig gelogde latency van cameraopname tot FreeD-uitvoer. Deze logwaarde omvat niet de volledige zichtbare keten: de beeldverwerking van de Raspberry Pi bevat nog een extra frame in de buffer van de beeldsignaalprocessor (ISP), en de vertraging tussen FreeD-uitvoer en zichtbaar bewegend beeld op de LED-wall wordt afzonderlijk beoordeeld.

Naast de kwantitatieve maten wordt de visuele werking van de virtuele camera beoordeeld. Daarbij wordt nagegaan of bewegingen in de verwachte richting en schaal worden weergegeven en of zichtbare problemen optreden, zoals *jitter*, schokken, foutieve parallax of onstabiel herstel na tijdelijk trackingverlies.

5.6 Overzicht van de evaluaties

Tabel 5.1 geeft een overzicht van de evaluatielijnen met hun opstelling, doel en de gebruikte beoordelingsmaten. Het volgende hoofdstuk rapporteert de resultaten volgens dezelfde scheiding tussen sterkaartkwaliteit, beeldverwerking, nauwkeurigheidsvalidatie en prestatie- en integratieevaluatie.

Tabel 5.1: Overzicht van de evaluatielijnen en hun beoordelingsmaten

Evaluatie	**Opstelling**	**Doel**	**Maten**
Sterkaartkwaliteit	LiDAR-scan en finale sterkaart	Geometrische fout van de markerkaart	Markerafwijking, gemiddelde fout, maximale fout
Detectiestabiliteit	Stilstaande en belichte testbeelden	Stabiliteit en snelheid van markerdetectie	Detectietijd, aantal detecties, centroïde-jitter
Localisatie-benchmark	Benchmarkbeelden zonder voorgaande pose	Succes van globale posebepaling	Succespercentage, inliers, reprojectiefout, rekentijd
Stabiliteit bij stilstand	Stilstaande rig, vision- en uitvoer-logbestanden	Posejitter zonder externe referentie	Positie- en oriëntatieafwijking, p95, maximum
Posevalidatie	Vive Tracker, installatiekalibratie en validatietrajecten	Relatieve fout tijdens beweging	Positiefout, yawfout, p95, RMSE, APE en RPE
Prestatie en integratie	Unreal Engine, FreeD en LED-wall	Realtime werking van de uitvoerketen	FPS, Hz, latency, visuele stabiliteit

Hoofdstuk 6

Resultaten en kostenanalyse

Dit hoofdstuk brengt de resultaten samen van de evaluaties die in het vorige hoofdstuk zijn beschreven. De kalibraties vormen daarbij het vertrekpunt, omdat de camera-, IMU- en schaalparameters de verdere metingen rechtstreeks beïnvloeden. Daarna wordt beoordeeld hoe nauwkeurig de sterkaart is, hoe stabiel de visuele pose-uitvoer blijft en hoe goed het prototype overeenkomt met de Vive Tracker tijdens beweging. De prestatie in de LED-wallopstelling wordt afzonderlijk besproken en tot slot volgt een kostenanalyse, waarin de materiaalkost van het prototype naast commerciële cameratrackingsystemen wordt geplaatst.

6.1 Kalibratieresultaten

6.1.1 Intrinsieke camerakalibratie

De intrinsieke camerakalibratie bepaalt de cameramatrix en de lensvervorming die in de verdere detectie-, localisatie- en trackingstappen worden gebruikt. De operationele kalibratie in het prototype is opgeslagen in het vaste intrinsicsbestand van de software. Tabel 6.1 vat de belangrijkste waarden samen.

Tabel 6.1: Belangrijkste resultaten van de intrinsieke camerakalibratie. De parameters beschrijven de projectie van het gecorrigeerde camerabeeld en vormen de basis voor detectie, localisatie en PnP-poseberekening.

Grootheid	**Waarde**	**Betekenis**
Beeldresolutie	2304×1296 px	Resolutie van de trackingbeelden
Brandpuntsafstand f_x, f_y	997,451 px; 998,276 px	Schaling van cameracoördinaten naar pixels
Hoofdpunt c_x, c_y	1163,121 px; 650,630 px	Optisch centrum in het beeldvlak
Operationele RMS-fout	0,590 px	Gemiddelde reproj. restfout van de gebruikte intrinsics
Kalibr-residu per as	1,555 px; 1,422 px	Spreiding van de reproj. fout in de Kalibr-controlemeting

Deze kalibratie vormt een ondergrens voor de posekwaliteit: een detector kan een marker niet beter terugprojecteren dan de combinatie van camerakalibratie, centroïdebepaling en kaartnauwkeurigheid toelaat. De latere reprojectiefouten in de localisatiebenchmark moeten daarom niet

als zuivere detectorfouten gelezen worden, maar als een combinatie van alle fouten in de visuele geometrie.

6.1.2 Camera-IMU-kalibratie en IMU-ruis

Naast de camerakalibratie werd de geometrische en temporele relatie tussen camera en IMU bepaald. De camera-IMU-kalibratie levert de vaste transformatie tussen beide sensoren en de tijdsverschuiving waarmee camerametingen en IMU-metingen op elkaar worden afgestemd. De gebruikte IMU-ruisparameters zijn afkomstig uit een afzonderlijk Allan-variantieonderzoek. Tabel 6.2 geeft de waarden die relevant zijn voor de ESKF-integratie.

Tabel 6.2: Resultaten van de camera-IMU-kalibratie en het IMU-ruisonderzoek. De extrinsieke kalibratie legt de vaste transformatie tussen camera en IMU vast; de ruisparameters bepalen hoe zwaar de IMU-metingen in de ESKF-propagatie doorwegen.

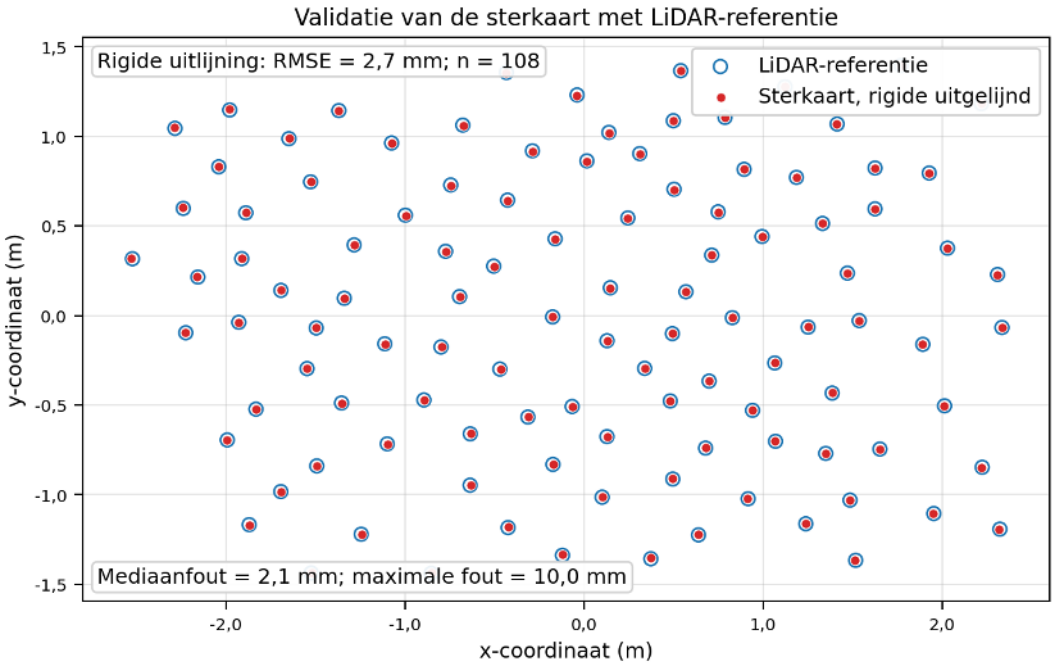
Grootheid	**Waarde**	**Betekenis**
Camera-IMU-tijdsoffset	15,905 ms	$t_{\text{imu}} = t_{\text{cam}} + 0{,}015905\,\text{s}$
Translatie IMU naar camera	$x = -0{,}10$ mm $y = -0{,}70$ mm $z = -14{,}50$ mm	Hefboomarm tussen beide sensoren
Gemiddelde camera-IMU-reprojectiefout	2,276 px	Restfout van de gezamenlijke kalibratie
Gyroscoopruisdichtheid	$7{,}2 \cdot 10^{-5}$	Parameter uit Allan-variantieanalyse
Gyroscoop-*random walk*	$1{,}0 \cdot 10^{-6}$	Parameter uit Allan-variantieanalyse
Versnellingsruisdichtheid	$1{,}63 \cdot 10^{-3}$	Parameter uit Allan-variantieanalyse
Versnellings-random walk	$2{,}35 \cdot 10^{-5}$	Parameter uit Allan-variantieanalyse

De camera-IMU-kalibratie maakt duidelijk dat de IMU niet enkel als losse stabilisator wordt gebruikt. De ESKF-laag is afhankelijk van een vaste ruimtelijke offset, een gemeten tijdsoffset en realistische ruisparameters. Restfouten in deze kalibratie kunnen zich rechtstreeks vertalen naar kleine bias of vertraging in de gefuseerde pose-uitvoer.

6.2 Kwaliteit van de sterkaart

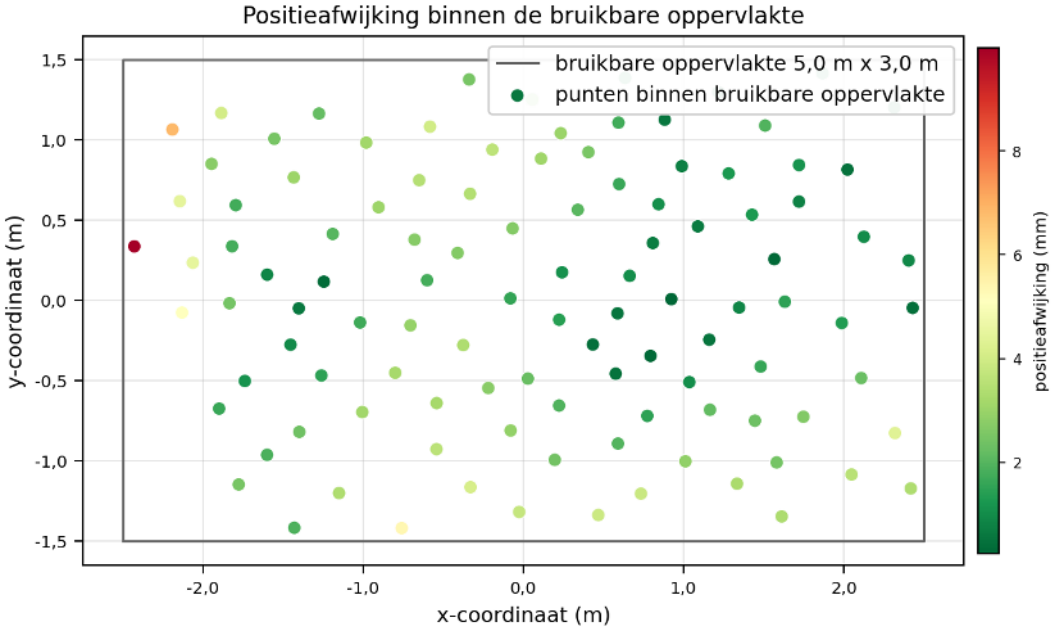
De sterkaart werd geëvalueerd door de geschatte markerposities te vergelijken met de onafhankelijke LiDAR-referentiemeting van dezelfde markeromgeving, zoals beschreven in het vorige hoofdstuk. In totaal werden 108 markers geïdentificeerd door beide technieken en vergeleken over een meetgebied van circa 5 m bij 3,0 m. Markers bij de grens van dit gebied werden uitsluitend vanuit één kijkrichting waargenomen, wat een verwachte nauwkeurigheidsdegradatie naar de randen toe veroorzaakt.

De overlay in figuur 6.1 vergelijkt de opgebouwde sterkaart na rigide uitlijning met de LiDAR-referentie. Over het volledige meetgebied bedraagt de RMSE 2,7 mm met een mediaanfout van 2,1 mm. De maximale afwijking van 10,0 mm treedt op bij de randmarkers, wat verklaarbaar is door de beperktere observatiedekking van de randmarkers: tijdens de kaartopbouw worden deze uitsluitend vanuit één rijrichting waargenomen. De centrale markers vertonen een nauwkeurige uitlijning met de LiDAR-referentie, met afwijkingen in de orde van 1 à 2 mm. Tabel 6.3 vat de kwantitatieve resultaten samen.



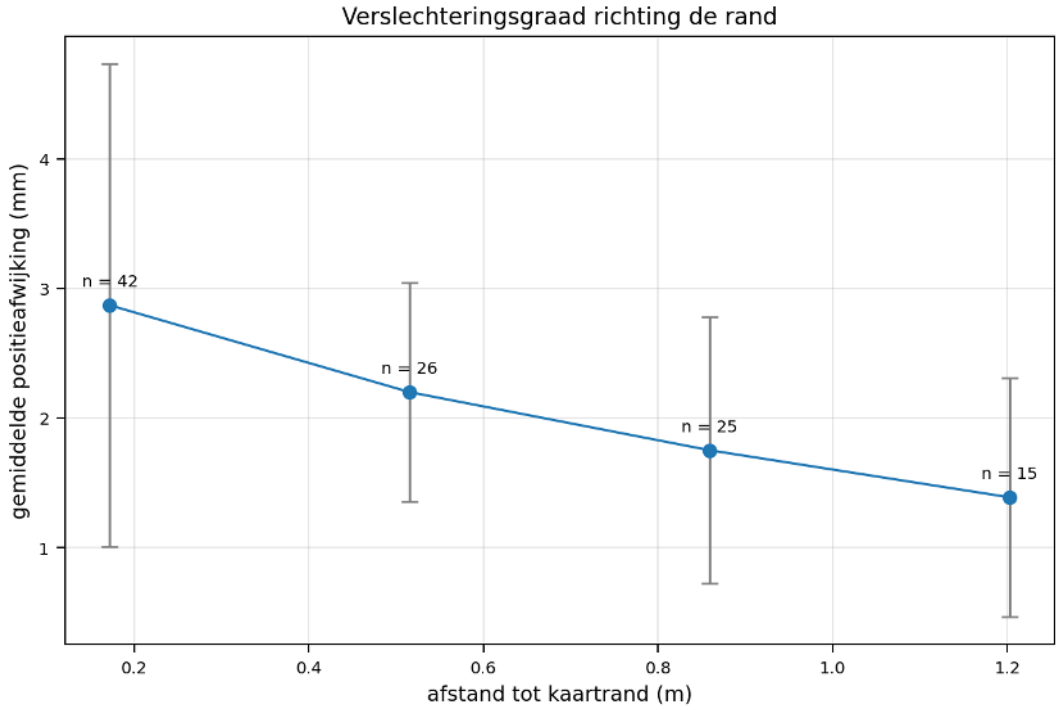
Figuur 6.1: Vergelijking van de opgebouwde sterkaart (rood) met de LiDAR-referentie (blauw) na rigide uitlijning; RMSE = 2,7 mm over 108 markers.

De ruimtelijke verdeling van de afwijkingen is weergegeven in figuur 6.2. De randmarkers vertonen een duidelijk hogere fout dan de centrale markers, consistent met de verwachte observatiedekking-afhankelijkheid. De degradatie is gradueel en lokaal verklaarbaar, niet willekeurig verspreid over de kaart.



Figuur 6.2: Positieafwijking per marker binnen de bruikbare oppervlakte van de sterkaart. De kleur geeft de afwijking ten opzichte van de LiDAR-referentie weer; de hoogste fouten bevinden zich vooral aan de rand van het meetgebied.

Figuur 6.3 vat hetzelfde effect samen als functie van de afstand tot de kaartrand. De gemiddelde positieafwijking neemt af naarmate markers verder van de rand liggen, wat bevestigt dat de kaartkwaliteit vooral aan de grens van het gemeten gebied afneemt.



Figuur 6.3: Gemiddelde positieafwijking van de sterkaart in functie van de afstand tot de kaartrand. De foutbalken tonen de spreiding binnen elke afstandsklasse; markers dicht bij de rand vertonen gemiddeld de grootste afwijking.

Tabel 6.3: Resultaten van de vergelijking tussen sterkaart en LiDAR-referentie. De RMSE en maximale afwijking geven weer hoe goed de door het prototype opgebouwde sterkaart overeenkomt met de onafhankelijke LiDAR-gebaseerde referentie.

Maat	**Resultaat**
Aantal vergeleken markers	108
RMSE	2,7 mm
Mediaanafwijking	2,1 mm
Maximale afwijking	10,0 mm

Een RMSE van 2,7 mm over een meetgebied van 4,5 m bij 3,0 m is een sterk resultaat voor een prototype dat uitsluitend gebruik maakt van consumentgerichte hardware: een brede IMX708-camera, eenvoudige retroreflectieve stickers en een Raspberry Pi 5 als rekeneenheid. De beperkte degradatie aan de randen is een verwacht en verklaarbaar gevolg van de meetopstelling en niet van een fundamentele tekortkoming van het systeem. Voor een virtual-productietoepassing, waarbij het centrale werkgebied het meest wordt gebruikt, is deze geometrische nauwkeurigheid meer dan voldoende.

6.3 Detectiestabiliteit

De detectiestabiliteit werd afzonderlijk onderzocht omdat de nauwkeurigheid van de poseberekening begint bij de pixelpositie van elke gedetecteerde marker. Deze meting beoordeelt dus nog geen camerapose, maar de spreiding van de gedetecteerde centroïdes in het beeld. Per frame worden de ruwe centroïdes gedetecteerd en via nearest neighbour aan de referentieset uit het eerste frame gekoppeld. Per marker wordt daarna de standaardafwijking van de pixelpositie berekend; de gerapporteerde waarden zijn het gemiddelde van die standaardafwijkingen over alle gevolgde markers.

Tabel 6.4 toont dat beide detectoren in een donkere, stilstaande en gecontroleerde trackingomgeving vrijwel dezelfde centroïdestabiliteit halen. De spreiding blijft voor beide configuraties rond 0,1 pixels per as. Het grootste verschil is de rekentijd: de lichte detector verwerkt een frame gemiddeld in 10,0 ms, terwijl de zware detector gemiddeld 28,7 ms nodig heeft.

Tabel 6.4: Detectietijd en centroïde-jitter op 1000 stilstaande beelden. De tabel vergelijkt de lichte detector en de volledige detector onder stationaire omstandigheden in de normale donkere trackingomgeving.

Configuratie	**Gem. tijd (ms)**	**Min. (ms)**	**Max. (ms)**	**Detecties gem.**	σ_x / σ_y **(px)**
Lichte detector	10,0	9,8	28,6	33,0	0,108 / 0,137
Zware detector	28,7	28,4	38,9	33,6	0,107 / 0,132

Dezelfde detectoren werden ook op 23 afzonderlijke beelden met storende omgevingsverlichting getest. Die beelden vormen geen continue stilstaande sessie, waardoor centroïdestabiliteit daar geen geldige maat is. De vergelijking in tabel 6.5 rapporteert daarom alleen detectietijd en aantal gedetecteerde markers.

Tabel 6.5: Detectorvergelijking op belichte beelden met variabele camerastand. Deze bewegende dataset bevat meer omgevingslicht en wordt gebruikt om het gedrag van de detectoren buiten de normale donkere trackingcondities te vergelijken.

Configuratie	**Gem. tijd (ms)**	**Min. (ms)**	**Max. (ms)**	**Gem. detecties**
Lichte detector	11,1	9,7	16,2	11,3
Zware detector	32,5	29,7	39,5	23,1

De belichte beelden tonen waarom beide detectoren een andere rol hebben. De lichte detector is geschikt voor real-time tracking omdat hij snel is, maar hij mist dimmere markers wanneer er veel omgevingslicht aanwezig is. De zware detector vindt in dezelfde beelden gemiddeld meer dan dubbel zoveel markers, maar is ongeveer drie keer trager. Dat maakt de zware detector vooral nuttig voor kaartopbouw en controlebeelden, terwijl de lichte detector beter aansluit bij continue pose-uitvoer.

6.4 Localisatiebenchmark

De localisatiebenchmark evalueert de poseberekening wanneer er nog geen voorgaande pose beschikbaar is. De benchmark gebruikt de lichte detector met de operationele parameters van het prototype en lost de pose op met SQPnP en Levenberg-Marquardtverfijning op de inlierset.

Tabel 6.6: Localisatiebenchmark met de lichte detector. De tabel rapporteert de slaagkans, reprojectiefout en verwerkingstijd van globale localisatie zonder vooraf beschikbare pose.

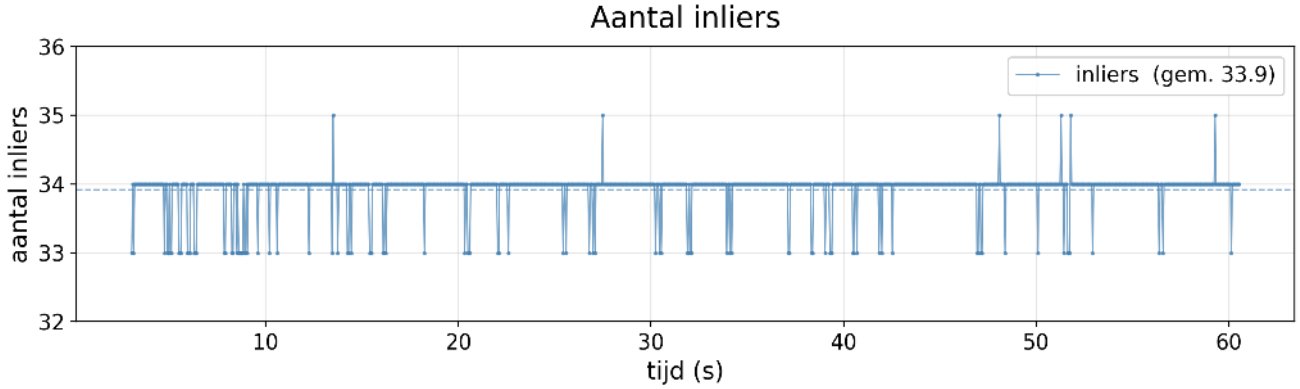
Dataset	**Detecties gem.**	**Inliers gem.**	**Reproj. gem./med. (px)**	**Tijd det./solv. (ms)**
Belicht	11,3	8,4	1,220 / 1,224	8,0 / 1,9
Optimaal	33,0	26,7	1,761 / 1,785	10,0 / 0,9

Alle 1023 benchmarkbeelden worden succesvol gelokaliseerd. De gecombineerde gemiddelde herprojectiefout bedraagt 1,749 px en de mediaan bedraagt 1,785 px. De gemiddelde detectietijd ligt rond 10 ms en de solver zelf blijft rond 0,9 ms. Daarmee ligt de rekentijd voor localisatie hoofdzakelijk in de detectiestap, niet in het oplossen van de pose. De zware detector wordt hier niet als localisatieconfiguratie beoordeeld; zijn rol is eerder kaartopbouw en detectie onder ongunstige belichting, zoals in de vorige sectie besproken.

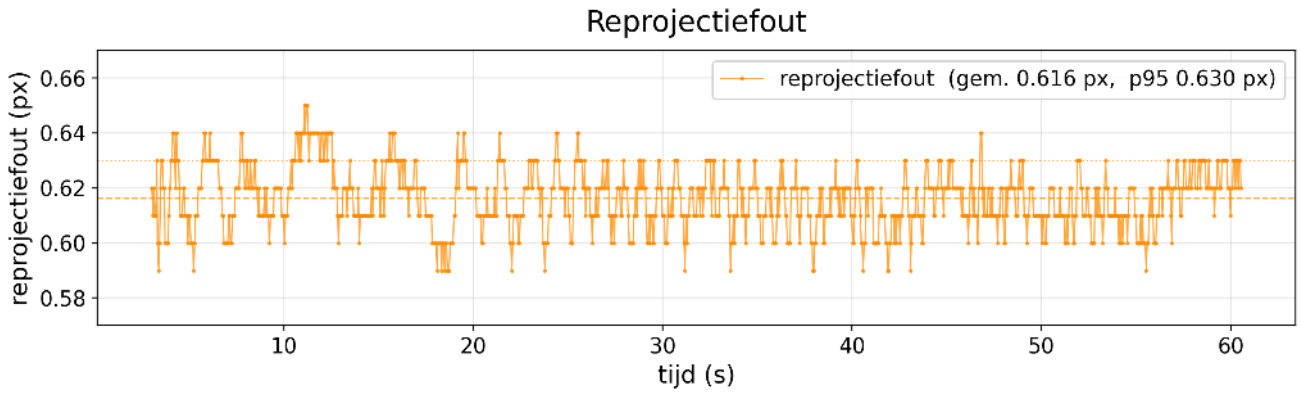
6.5 Stabiliteit bij stilstand

Stilstandmetingen worden gebruikt om de jitter en eventuele drift van de positie en oriëntatie te beoordelen. Hiervoor worden zowel de geldige visuele trackingposes als de uiteindelijke gefilterde uitvoerpose uit dezelfde sessie gebruikt. De afwijkingen worden berekend ten opzichte van de gemiddelde pose over de sessie.

Naast de posejitter worden ook het aantal inliers en de reprojectiefout van de visuele trackingposes gerapporteerd. Deze maten beschrijven de interne kwaliteit van de visuele poseberekening. Het aantal inliers geeft aan hoeveel markerdetecties door de poseoplossing worden ondersteund; figuur 6.4 toont dit aantal voor dezelfde prestatie-evaluatie. De reprojectiefout toont hoe goed de berekende pose de detecties opnieuw in het beeld projecteert en wordt afzonderlijk weergegeven in figuur 6.5.

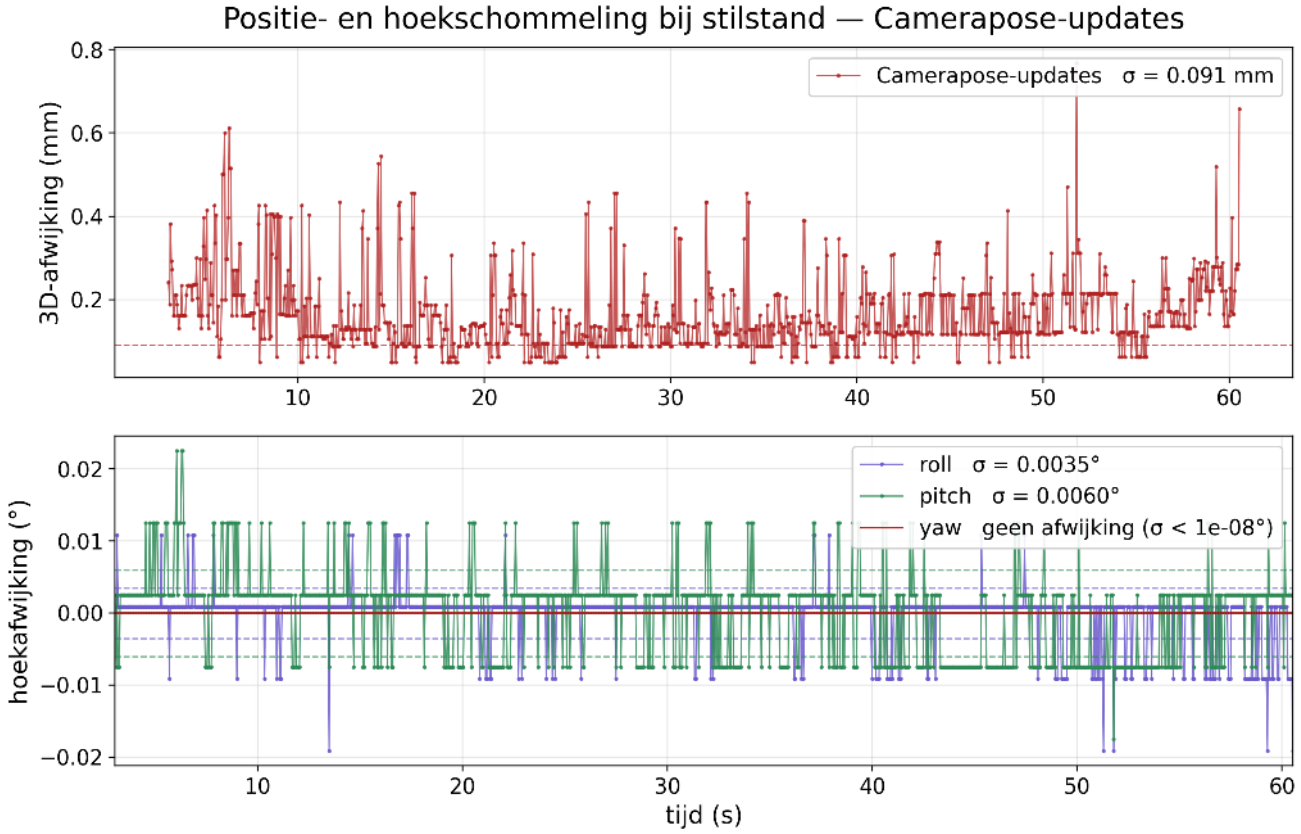


Figuur 6.4: Aantal inliers van de visuele poseberekening tijdens de prestatie-evaluatie. De inliers geven aan hoeveel markerkoppelingen de pose ondersteunen.

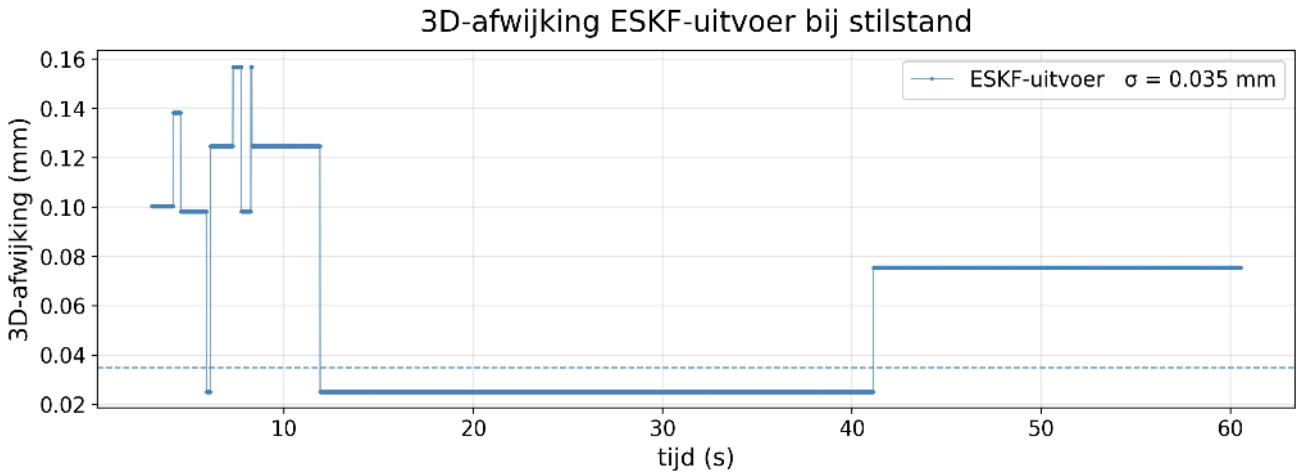


Figuur 6.5: Reprojectiefout van de visuele poseberekening tijdens de prestatie-evaluatie. De reprojectiefout geeft weer hoe goed de geschatte pose de gedetecteerde markercentra verklaart.

De positieschommeling bij stilstand wordt vervolgens afzonderlijk weergegeven voor de ruwe visuele poses en voor de gefilterde uitvoerpose. Figuur 6.6 toont de afwijking van de vision-poses ten opzichte van hun gemiddelde positie. Figuur 6.7 toont dezelfde maat voor de uiteindelijke uitvoerpose, zodat zichtbaar wordt hoeveel hoogfrequente schommeling door de filter- en afvlakkingsketen wordt onderdrukt.



Figuur 6.6: Positieschommeling van de visuele cameraposes bij stilstand, berekend ten opzichte van de gemiddelde visuele pose. Deze grafiek toont de ruwe optische posevariatie zonder ESKF-afvlakking.



Figuur 6.7: Positieschommeling van de gefilterde uitvoerpose bij stilstand, berekend ten opzichte van de gemiddelde uitvoerpose. De vergelijking met de visuele pose toont het stabiliserende effect van ESKF-fusie en *One Euro*-afvlakking.

De figuren tonen het tijdsverloop, maar vatten de spreiding niet numeriek samen. Daarom geeft tabel 6.7 de standaardafwijking of gemiddelde afwijking, het 95-percentiel en de maximale afwijking voor positie, roll, pitch en yaw. Die tabel maakt het verschil tussen de ruwe vision-poses en de gefilterde uitvoerpose kwantitatief vergelijkbaar.

Tabel 6.7: Pose-stabiliteit bij stilstaande opstelling. De tabel onderscheidt de ruwe visuele pose van de gefilterde uitvoerpose, zodat zichtbaar wordt hoeveel positieschommeling door ESKF-fusie en *One Euro*-afvlakking wordt gereduceerd.

Maat	**Bron**	**Std./gem.**	**p95**	**Max.**
3D-positieafwijking (mm)	Vision	0,167	0,368	0,774
3D-positieafwijking (mm)	**Gefilterde output**	**0,057**	**0,123**	**0,155**
roll-afwijking (°)	Vision	<0,01	0,01	0,02
roll-afwijking (°)	Gefilterde output	<0,01	<0,01	<0,01
pitch-afwijking (°)	Vision	0,01	0,01	0,02
pitch-afwijking (°)	Gefilterde output	<0,01	<0,01	0,01
yaw-afwijking (°)	Vision	<0,01	<0,01	<0,01
yaw-afwijking (°)	Gefilterde output	<0,01	<0,01	<0,01

Over 1194 geldige visuele cameraposes en een meetduur van ongeveer 60 seconden blijft de 3D-p95-positieafwijking van de vision-poses beperkt tot 0,368 mm. De uiteindelijke gefilterde uitvoerpose verlaagt deze p95-positieafwijking tot 0,123 mm en reduceert dus de zichtbare positieschommeling. De oriëntatie blijft eveneens stabiel: roll en pitch blijven ruim onder 0,03°, terwijl de yawwaarde binnen de resolutie van de gelogde data constant blijft. De hoeken wijken in de gefilterde uitvoerpose niet zichtbaar af van de gemiddelde pose.

6.6 Tracking tijdens beweging met Vive Tracker

De posevergelijking tussen het prototype en de Vive Tracker rapporteert de relatieve positieafwijking nadat de kalibratieparameters zijn vastgelegd en enkel de tijdsoffset per validatietraject opnieuw wordt bepaald. Omdat de Vive Tracker zelf ook meetonnauwkeurigheid heeft, zijn de gerapporteerde waarden relatieve inter-systeemafwijkingen en geen absolute prototypefout ten opzichte van de ware positie. Per configuratie zijn vijf trajecten met normale snelheid en drie snelle stressproeftrajecten uitgevoerd.

De primaire vergelijking wordt gebaseerd op de trajecten met normale snelheid. De snelle trajecten worden afzonderlijk besproken als stressproeven die het gedrag buiten het beoogde werkgebied zichtbaar maken. De gedetailleerde resultaten worden eerst per configuratie vergeleken, daarna opgesplitst volgens snelheidsklasse en vervolgens ingezoomd op één representatief traject.

6.6.1 Vergelijking tussen configuraties

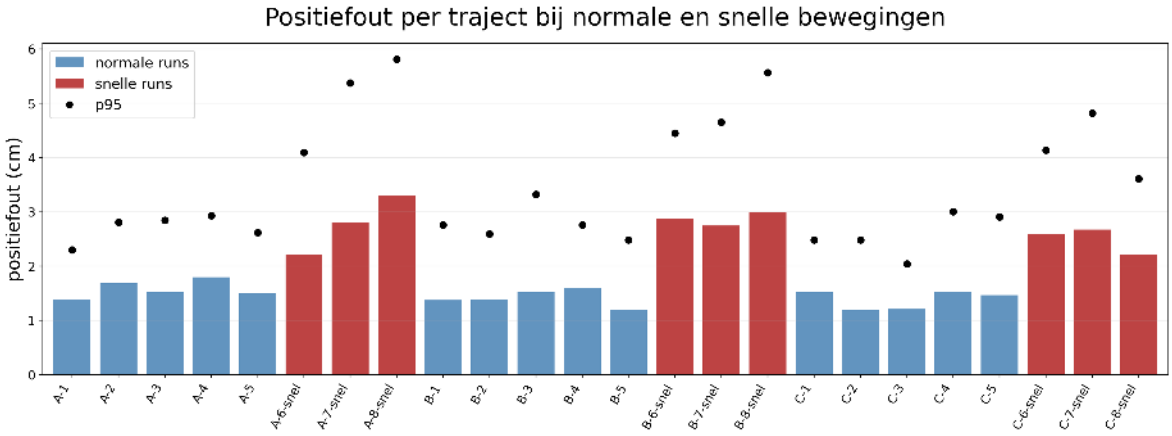
Tabel 6.8: Samenvatting van de relatieve inter-systeemafwijking voor normale validatietrajecten per configuratie.

Configuratie	**Gem. pos. (cm)**	**p95 (cm)**	**RMSE (cm)**	**Gem. yaw (°)**
A	1,577	2,699	1,700	0,732
B	1,412	2,781	1,587	0,685
C	1,382	2,583	1,523	0,808
Totaal normale runs (15)	**1,457**	**2,688**	**1,603**	**0,742**

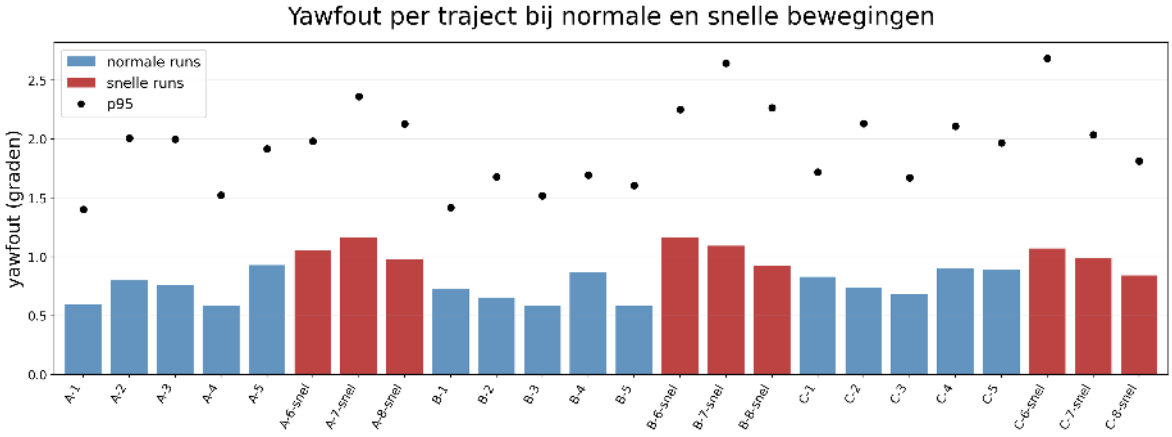
Over alle vijftien normale validatietrajecten samen bedraagt de gemiddelde positieafwijking 1,457 cm, met een gemiddelde p95 van 2,688 cm en een gemiddelde yawafwijking van 0,742°. De drie configuraties verschillen in de afstand tussen de gemonteerde Vive Tracker en het prototype: 7,96 cm (A), 20,93 cm (B) en 45,12 cm (C). Binnen deze vergelijking geeft configuratie C de laagste gemiddelde positieafwijking, terwijl configuratie B de laagste yawafwijking heeft. Dat verschil is plausibel in het licht van de configuratiegeometrie. Bij grotere afstand veroorzaakt een kleine yaw-fout een grotere positieverschuiving aan het prototypesensorpunt, waardoor een lange armlengte gevoeliger wordt voor oriëntatiefouten. Bij een zeer kleine armlengte wordt de fysieke meting van de verbindingsvector relatief onnauwkeuriger ten opzichte van de totale afstand, waardoor de onzekerheid in de yaw-offset toeneemt. Configuratie B ligt tussen beide extremen en vormt daardoor de meest gebalanceerde configuratie.

6.6.2 Normale versus snelle trajecten

Figuren 6.8 en 6.9 tonen respectievelijk de gemiddelde positie- en yawfout per traject, met snelle trajecten rood gemarkeerd. Bij normale snelheid liggen de positiefouten overwegend tussen 1 cm en 2 cm. Bij de snelle trajecten loopt de fout op tot boven de 4 cm. Daarbij geldt dat bij hogere snelheden niet alleen de prototypefout toeneemt, maar ook de meetruis van de Vive Tracker zelf, zoals onderbouwd in de evaluatiemethode op basis van de literatuur. De gerapporteerde afwijkingen bij snelle trajecten zijn daardoor een bovenbegrenzing die beide systemen omvat.

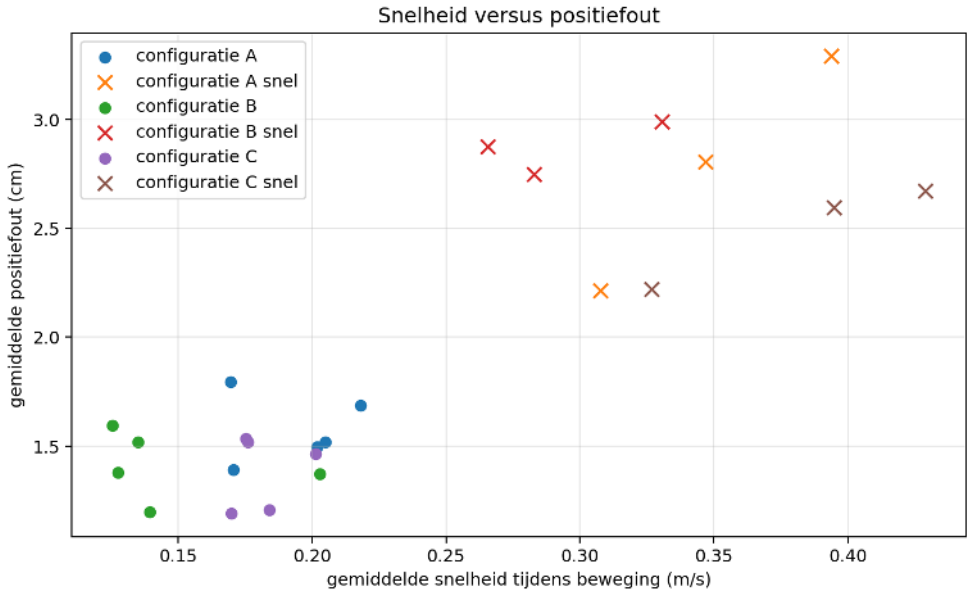


Figuur 6.8: Gemiddelde positiefout en p95 per validatietraject over alle drie configuraties; snelle stressproeftrajecten zijn rood gemarkeerd.



Figuur 6.9: Gemiddelde absolute yawfout en p95 per validatietraject; snelle trajecten vertonen een duidelijk hogere yawfout.

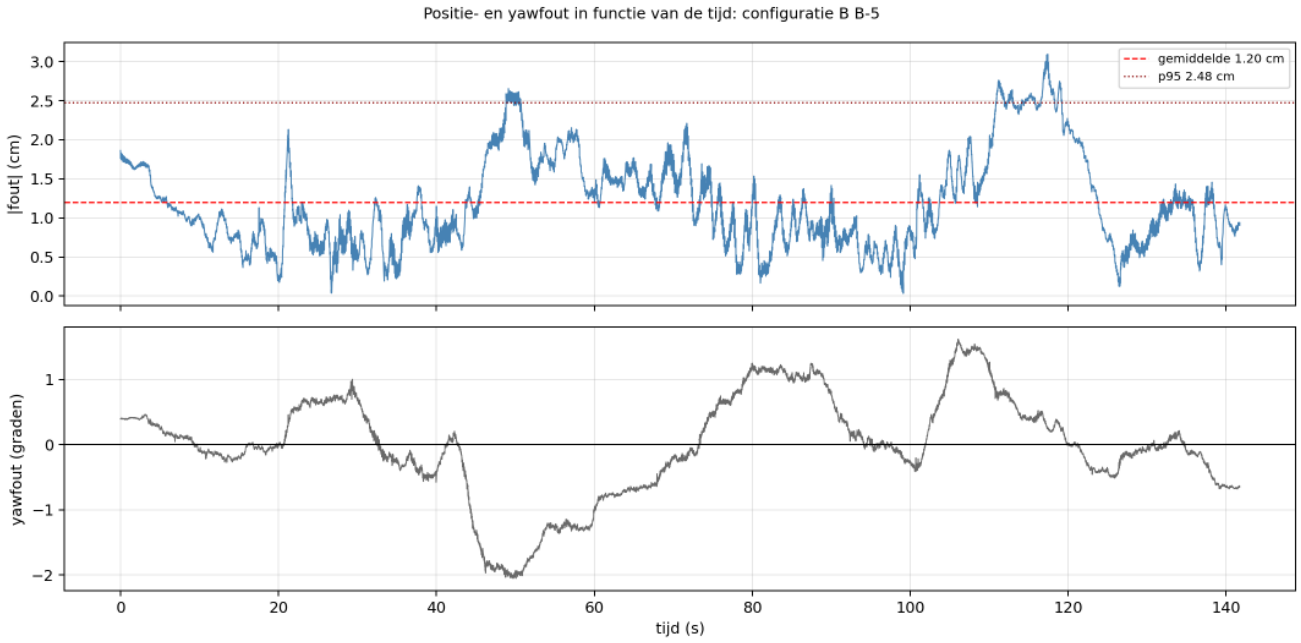
Het verband tussen gemiddelde snelheid en positiefout is weergegeven in figuur 6.10. De normale trajecten clusteren links van de 25 cm/s-grens en vertonen geen fouttoename met snelheid; de snelle trajecten vormen een duidelijk afgescheiden groep met een sterkere stijging. Dit bevestigt dat de fouttoename bij hoge snelheid primair samenhangt met de algoritmische beperkingen van de nearest-neighbour-tracker, de lage plafondafstand en de updatefrequentie van de visuele poseberekening.



Figuur 6.10: Gemiddelde positiefout als functie van de gemiddelde bewegingssnelheid per traject; normale en snelle trajecten zijn per configuratie onderscheiden.

6.6.3 Ingezoomde analyse van traject B-5

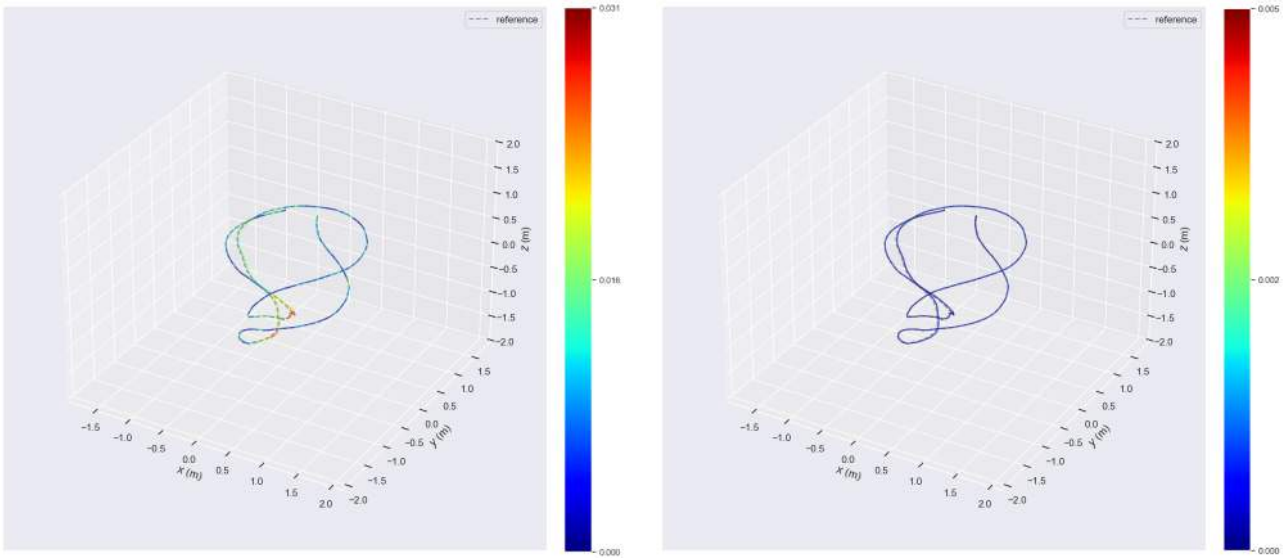
Traject B-5 wordt als ingezoomd voorbeeld gebruikt omdat configuratie B de meest gebalanceerde configuratie vorm en B-5 een representatief traject is. Figuur 6.11 toont het tijdverloop van de positie- en yawafwijking.



Figuur 6.11: Tijdverloop van de positieafwijking (boven) en de yawafwijking (onder) voor traject B-5.

Figuur 6.12 vergelijkt vervolgens de *absolute pose error* (APE) en de *relative pose error* (RPE) langs hetzelfde traject. APE geeft de absolute positieafwijking ten opzichte van het gecorrigeerde

Vive-referentietraject per punt langs het volledige pad. De fout piekt vooral in de snellere bochten, waar de visuele tracker meer belast wordt en dus de IMU-voorspellingen zwaarder doorwegen. RPE vergelijkt daarentegen niet de absolute positie ten opzichte van het volledige wereldkader, maar de lokale relatieve trajectconsistentie over een vaste afstand. Daardoor is deze metriek minder gevoelig voor een globale offset en blijft het trajectbeeld stabieler, met overwegend lage foutniveaus buiten de dynamischste bochtsegmenten.



(a) *Absolute pose error* (APE)

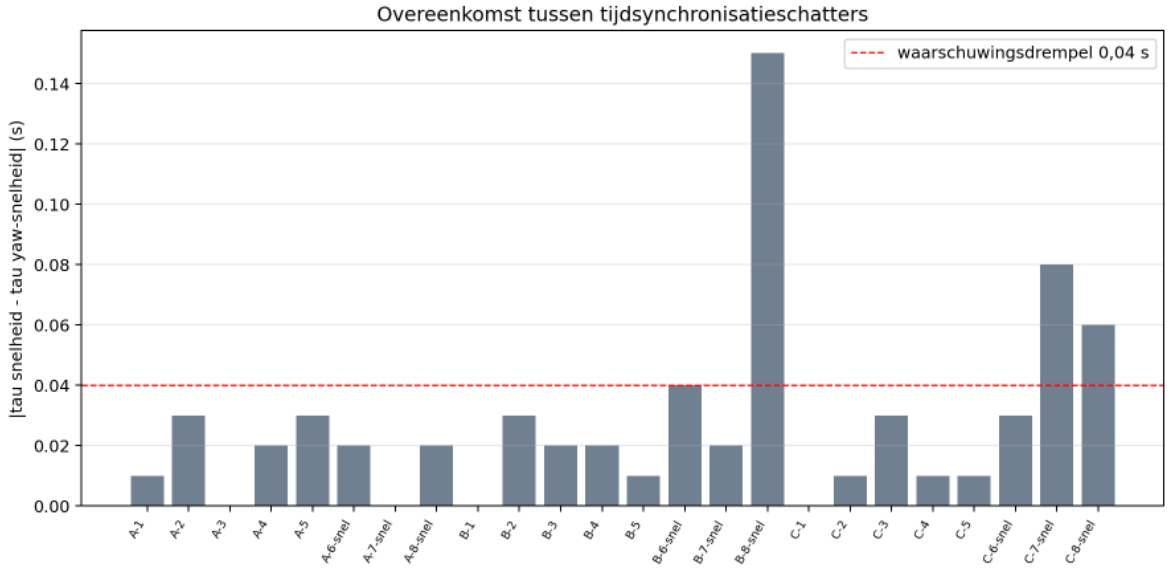
(b) *Relative pose error* (RPE)

Figuur 6.12: Ruimtelijke vergelijking van de APE en RPE voor traject B-5. De APE-kaart toont de absolute fout ten opzichte van het gecorrigeerde Vive-referentietraject; de RPE-kaart toont de lokale trajectconsistentie over een vaste afgelegde afstand.

6.6.4 Tijdsynchronisatie

De tijdsynchronisatie is cruciaal voor de nauwkeurigheid van de vergelijking: een resterende tijdsoffset van 40 ms bij een snelheid van 20 cm/s veroorzaakt al een schijnbare positieverschuiving van circa 8 mm. Per traject wordt de tijdsoffset bepaald via kruiscorrelatie van de snelheidsprofielen van beide systemen. Als onafhankelijke kruiscontrole wordt dezelfde correlatie ook op de yaw-snelheidsprofielen uitgevoerd. Wanneer beide schatters naar dezelfde waarde wijzen, is de tijdsoffset betrouwbaar bepaald; een groot verschil tussen beide wijst op een laag signaal-ruisverhouding in het traject of op een bewegingspatroon dat weinig structuur heeft voor één van de twee korrelatiekanalen.

Figuur 6.13 toont het absolute verschil tussen de twee tijdsoffsetschatters per traject. Voor nagenoeg alle trajecten bedraagt dit verschil minder dan 0,04 s, wat wijst op een robuuste synchronisatie. Enkele snelle trajecten overschrijden deze grens maar dit heeft geen impact op de vergelijking aangezien die trajecten niet opgenomen worden in de resultaten.

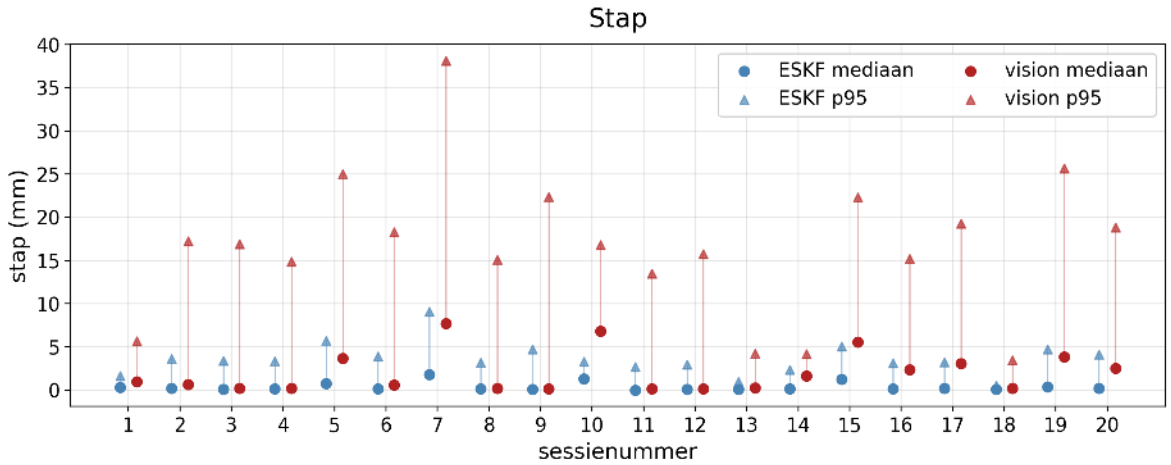


Figuur 6.13: Absoluut verschil tussen de snelheidsgestuurde en yaw-snelheidsgestuurde tijdsoffsetschatters per traject; de rode stippellijn markeert de waarschuwingsdrempel van 0,10 s.

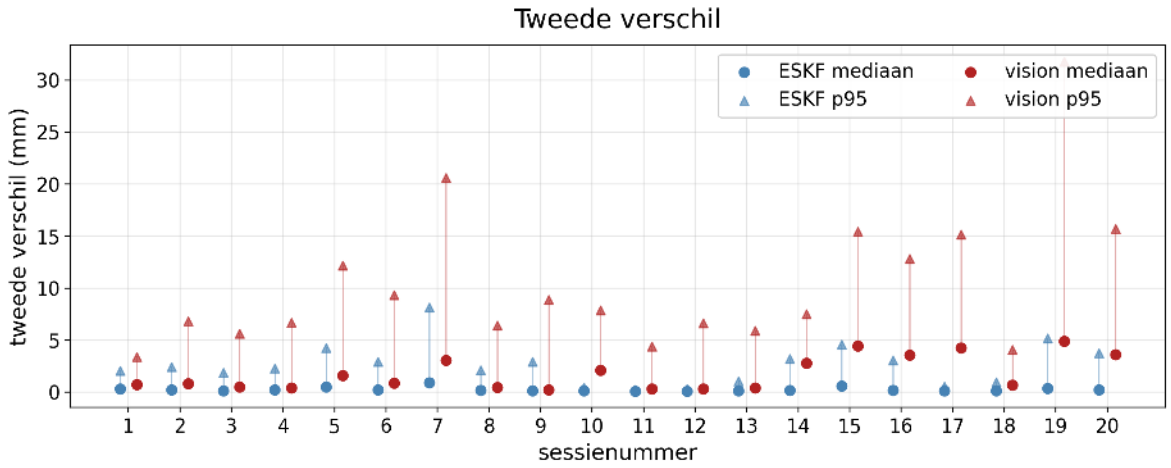
6.6.5 Interne positieschommeling tijdens beweging

Naast de vergelijking met de Vive Tracker werden de bewegingslogs ook intern geanalyseerd. Deze analyse gebruikt alleen de pose-uitvoer van het prototype en is dus geen Vive-pose. De stap is de Euclidische afstand tussen twee opeenvolgende posities, $d_k = \|p_k - p_{k-1}\|$. Het tweede verschil is de eindige tweede positiedifferentie, $s_k = \|p_k - 2p_{k-1} + p_{k-2}\|$. Deze tweede maat beschrijft vooral abrupte positieveranderingen of schokkerigheid in de uitvoer.

Figuur 6.14 toont de stapgrootte per sessie voor de ruwe vision-poses en de gefilterde uitvoerpose. Omdat de opstelling tijdens deze logs beweegt, bevat deze maat zowel echte verplaatsing als meetruis. Figuur 6.15 toont daarom ook het tweede verschil, dat abrupte positieveranderingen en hoogfrequente schommelingen sterker zichtbaar maakt. Tabel 6.9 vat dezelfde analyse numeriek samen.



Figuur 6.14: Stapgrootte van de vision-poses en de gefilterde uitvoerpose in de opgeschoonde bewegingslogs. Deze metriek is intern berekend en gebruikt geen Vive Tracker als referentie.



Figuur 6.15: Tweede positiedifferentie van de vision-poses en de gefilterde uitvoerpose in de opgeschoonde bewegingslogs. Deze metriek legt abrupte positieveranderingen sterker bloot dan de stapgrootte.

Tabel 6.9: Interne positieschommeling in de opgeschoonde bewegingslogs. Deze waarden zijn berekend uit opeenvolgende prototypeposes en gebruiken geen Vive Tracker als externe referentie.

Maat	**vision (mm)**	**Gefilterde uitvoer (mm)**
Mediaan van de stapmediaan per sessie	0,82	0,17
Mediaan van de stap-p95 per sessie	16,88	3,32
Mediaan van het tweede verschil per sessie	0,86	0,19
Mediaan van de tweede-verschil-p95 per sessie	7,70	2,36

Omdat de opstelling tijdens deze logs werkelijk beweegt, bevat de stapgrootte zowel echte verplaatsing als meetruis. Het tweede verschil is daardoor de nuttigere indicator voor hoogfrequente schommelingen. De gefilterde uitvoerpose ligt voor beide maten duidelijk lager dan de ruwe vision-poses, wat overeenkomt met de stilstandanalyse: de uitvoerketen vermindert de positieschommeling, terwijl de hoeken geen vergelijkbare afwijking vertonen.

6.6.6 Bespreking van de validatieresultaten

Prototype-stabiliteit

De stabiliteitsanalyse beoordeelt de uitvoer van het prototype zelf. Daarbij is de gefilterde uitvoerpose de relevante eindpose die naar Unreal Engine wordt doorgestuurd. Deze pose bestaat uit ESKF-fusie en de daaropvolgende *One Euro*-afvlakking. Bij stilstand blijft de gefilterde uitvoerpose beperkt tot een 3D-p95-positieafwijking van 0,123 mm ten opzichte van de gemiddelde uitvoerpose. De ruwe vision-pose heeft in dezelfde sessie een p95 van 0,368 mm, maar die waarde beschrijft niet het uiteindelijke trackingresultaat.

Ook tijdens beweging vermindert de gefilterde uitvoerpose de interne positieschommeling. De stap-p95 daalt van 16,88 mm voor de vision-poses naar 3,32 mm voor de uitvoerpose, terwijl de p95 van het tweede verschil daalt van 7,70 mm naar 2,36 mm. Deze resultaten tonen vooral dat de combinatie van ESKF en *One Euro*-afvlakking de pose-uitvoer gladder en constanter maakt

dan de afzonderlijke visuele updates. Ze zeggen nog niets over de absolute positiejuistheid in de ruimte; daarvoor is de vergelijking met een externe tracker nodig.

Foutbronnen in de Vive Tracker vergelijking

De gerapporteerde positieafwijkingen zijn relatieve inter-systeemfouten, geen absolute prototypefout ten opzichte van de ware positie. Drie foutbronnen dragen bij aan de gemeten afwijking.

De eerste foutbron is de tijdsynchronisatie. De tijdsoffset tussen de Vive Tracker en het prototype wordt per traject bepaald via kruiscorrelatie. Voor nagenoeg alle trajecten stemmen de twee schatters overeen binnen 0,04 s; bij 20 cm/s leidt een offset van 40 ms tot een positieverschuiving van circa 8 mm.

De tweede foutbron is de eigen meetonnauwkeurigheid van de Vive Tracker. De Vive Tracker heeft een restsysteemruis die afhankelijk is van de kwaliteit van de basisstationopstelling, de zichtlijn en de bewegingssnelheid. Bij de snelheden die in dit onderzoek worden gereden, typisch 10 tot 25 cm/s voor normale trajecten en hoger voor snelle stressproeftrajecten, is de Vive-meetruis niet te verwaarlozen. De gerapporteerde afwijkingen bevatten daardoor zowel de werkelijke prototypeonnauwkeurigheid als de Vive-meetruis; de prototypefout is bijgevolg gelijk aan of lager dan de gerapporteerde waarden. De twee foutbronnen kunnen met deze evaluatiemethode niet worden gescheiden.

De derde foutbron is de kalibratiefout van de transformatieparameters. De verbindingsvector, γ, Y_R en Y_t worden eenmalig per installatie bepaald en daarna vastgehouden. Eventuele restfouten in deze parameters vertalen zich in een systematische bias die in de trajectresultaten zichtbaar is.

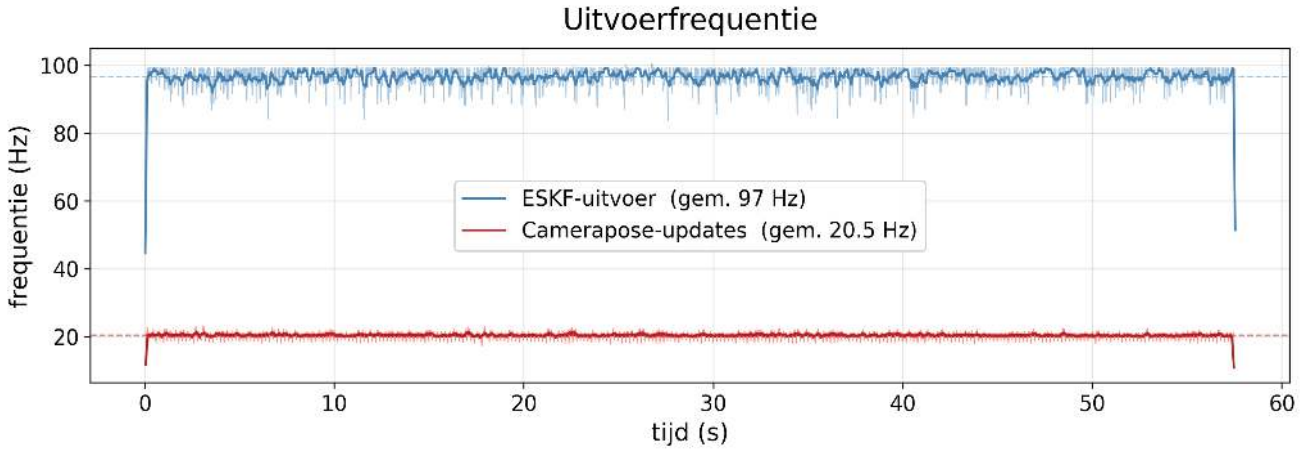
Vergelijking met de Vive Tracker

De Vive-analyse beoordeelt niet de interne stabiliteit, maar de relatieve overeenkomst tussen het prototype en een onafhankelijk trackingsysteem. Over alle normale validatieruns bedraagt de gemiddelde relatieve positieafwijking 1,457 cm, met een p95 van 2,688 cm. De positieafwijking vertoont daarbij geen structurele drift doorheen een traject: de afwijking oscilleert rond een beperkt gemiddelde zonder cumulatief op te lopen. De yawafwijking blijft voor normale trajecten gemiddeld onder 1° en is daarmee stabiel genoeg voor correcte cameraoriëntatie in een renderomgeving.

Bij hogere snelheden stijgt de positieafwijking significant. De primaire oorzaak is een algoritmische beperking: de nearest-neighbourtracker verliest sterren bij snelle verplaatsingen, waardoor het prototype vaker terugvalt op globale herlocalisatie. De grens van circa 25 cm/s is echter geen louter algoritmische grens: ook de plafondafstand en de updatefrequentie van de visuele poseberekening zijn parameters in de snelheidslimiet. Met een grotere plafondafstand of een hogere *framerate* zou dezelfde tracker al bij hogere snelheden goed functioneren. De grens is daardoor een systeemeigenschap van dit prototype, niet een fundamentele beperking van de inside-out aanpak.

6.7 Prestatie en *latency* in de LED-wallopstelling

De prestatie-evaluatie beschrijft hoe snel het prototype posegegevens beschikbaar maakt voor de renderomgeving. Daarbij zijn twee zaken belangrijk: hoe vaak Unreal Engine een nieuwe pose ontvangt en hoeveel vertraging er zit tussen het camerabeeld en de verstuurde FreeD-pose. Figuur 6.16 toont de updatefrequenties van de visuele pose en de gefilterde FreeD-uitvoer. Tabel 6.10 vat daarna de belangrijkste verwerkingstijden samen voor stilstand en beweging.



Figuur 6.16: Updatefrequenties van de vision-pose en de gefilterde FreeD-uitvoer tijdens de prestatie-evaluatie.

Tabel 6.10: Realtime prestaties van het prototype bij stilstand en beweging. De tabel groepeert camerafrequentie, visuele pose-updatefrequentie, uitvoerfrequentie en gemeten verwerkingstijden uit de beschikbare logs.

	Stilstand		**Beweging**	
Maat	**gem.**	**p95/p5**	**gem.**	**p95/p5**
Camerapose-updatefrequentie (FPS)	19,70	16,90	19,37	15,91
Detectietijd per frame (ms)	40,51	44,46	95,93	102,52
Poseberekeningtijd per frame (ms)	1,36	4,37	1,98	5,73
Gecombineerde pipelinetijd (ms)	41,87	48,83	97,90	108,25
ESKF-verwerkingsvertraging (ms)	7,12	11,97	7,30	13,63
Gelogde tracking-latency (ms)	49,29	62,84	105,25	121,97
Gemiddelde FreeD-uitvoerfrequentie (Hz)	96,00	n.v.t.	97,82	n.v.t.

Bij stilstand halen de visuele pose-updates gemiddeld 19,70 FPS; tijdens beweging bedraagt dit 19,37 Hz. De gefilterde FreeD-uitvoer draait gemiddeld op 97,82 Hz, waardoor Unreal Engine vaker een pose-update ontvangt dan de camera nieuwe visuele correcties levert.

De hogere latency tijdens beweging komt hoofdzakelijk door de detectiestap: de detectietijd stijgt van gemiddeld 40,51 ms bij stilstand naar 95,93 ms tijdens de bewegingslogs. Dat wijst niet op een trage FreeD-uitvoer of op een structureel falende IPPE-/PnP-stap, want de poseberekening zelf blijft beperkt tot gemiddeld 1,98 ms. De bewegingslogs bevatten vaker onscherpe beelden, gedeeltelijk zichtbare markers en momenten waarop de tracker na tijdelijk verlies opnieuw moet herlocaliseren. In die situaties is vooral de beeldverwerking en detectie zwaarder dan bij een stilstaande camera met stabiele markercentra.

De waarden in tabel 6.10 beschrijven bovendien alleen de softwarematig gelogde keten tot aan de FreeD-uitvoer. Ze zijn dus geen volledige vertraging van fysieke camerabeweging tot zichtbaar beeld op de LED-wall. In de Raspberry Pi-cameraketen zit nog één extra beeld in de buffer van de beeldsignaalprocessor (ISP), wat bij ongeveer 20 FPS overeenkomt met circa 50 ms bijkomende vertraging. Daarnaast werd voor de render- en weergaveketen een bijkomende vertraging van ongeveer 150 ms tussen FreeD-uitvoer en zichtbaar bewegend beeld op de LED-wall vastgesteld. De volledige zichtbare vertraging ligt daardoor hoger dan de gelogde tracking-*latency* alleen.

6.7.1 Integratie met Unreal Engine en LED-wall

De visuele evaluatie beschrijft hoe de door FreeD aangestuurde virtuele camera zich gedraagt in Unreal Engine en op de LED-wall. De FreeD-pakketten worden in Unreal Engine ontvangen via de Live Link FreeD-plugin, waarna de virtuele camera in de scène wordt aangestuurd. Voor de LED-wallweergave werd gebruikgemaakt van *nDisplay*; de bijhorende *nDisplay*-sessie werd via Switchboard gestart en beheerd. nDisplay rendert daarbij een *inner frustum*: het beeldgebied dat de opnamecamera op dat moment effectief ziet. Figuur 6.17 toont de overeenkomst tussen dat *inner frustum* en het perspectief van de opnamecamera. Figuur 6.18 toont daarna hoe de weergave op de LED-wall door het perspectief aangepast wordt tijdens camerabeweging.

(a) *Inner frustum* op de LED-wall, gegenereerd door *nDisplay*. Alleen het zichtbare gebied van de opnamecamera wordt in hoge kwaliteit weergegeven.

(b) Perspectief van de opnamecamera zelf. Dit beeld correspondeert met het zichtbare gebied dat op de LED-wall als *inner frustum* wordt weergegeven.

Figuur 6.17: Bovenaan de *nDisplay*-weergave van het *inner frustum* en het overeenkomstige perspectief van de opnamecamera. Samen tonen deze beelden welk deel van de virtuele scène effectief zichtbaar is voor de opnamecamera en dus op de LED-wall in hoge kwaliteit gerenderd wordt.

Binnen het bruikbare trackinggebied voelt rondlopen met de camera vloeiend aan. De ESKF en de *One Euro*-afvlakking doen daarbij duidelijk hun werk: kleine schommelingen in de visuele pose worden gedempt voordat de pose via FreeD naar Unreal Engine gaat. Zichtbare storingen treden vooral op wanneer de tracker te hoog, te laag of te schuin naar het plafond kijkt. Op die momenten verdwijnen te veel markers uit het bruikbare beeldgebied of worden ze minder betrouwbaar gedetecteerd, waardoor het systeem zich moet herlocaliseren en tijdelijk geen correcte poses kan doorsturen.

(a) Linker perspectiefweergave in schermvullende modus. De vervorming op de LED-wall past zich aan de actuele camerapose aan.

(b) Rechter perspectiefweergave in schermvullende modus. Samen met de linkerweergave toont dit beeld hoe perspectief verandert tijdens camerabeweging.

Figuur 6.18: Schermvullende perspectiefweergaven van de LED-wall. Deze beelden tonen hoe de render zich bij camerabeweging aanpast zodat parallax en kijkrichting blijven overeenkomen met de actuele pose-uitvoer.

6.8 Kostenanalyse

Deze sectie vergelijkt de hardware-aankoopkost van het prototype met die van commerciële cameratrackingsystemen voor virtual production. De vergelijking beperkt zich tot de directe hardware-aankoopkost: installatiewerk, kalibratiebegeleiding, onderhoudscontracten en softwarelicenties vallen buiten beschouwing, maar zijn bij commerciële systemen doorgaans ook een significant deel van de totale eigendomskost.

6.8.1 Kostenopbouw van het prototype

De kostenopbouw is opgesplitst in drie delen. Tabel 6.11 geeft de componenten van de trackingmodule zelf. Tabel 6.12 bevat alleen de markerinfrastructuur die nodig is om het prototype functioneel te gebruiken. Tabel 6.13 geeft onderdelen die het gebruiksgemak verhogen, maar niet nodig zijn voor een functionele cameratracker. De prijzen zijn afkomstig van de productpagina's van de respectieve leveranciers.

Tabel 6.11: Indicatieve hardware-aankoopkost van de trackingmodule. De tabel bevat enkel componenten die nodig zijn voor het basisfunctionerende prototype, exclusief optionele bediening en visualisatie.

Component	**Prijs**	**Bron**
Raspberry Pi 5 (16 GB)	€ 215	[51]
Camera Module 3 NoIR	€ 28	[43]
Adafruit BNO085 IMU	€ 20	[44]
IR-ledring	€ 1	[52]
KKSB behuizing	€ 25	[45]
3D-printmateriaal (PLA)	$<$ € 1	Bambu Studio
Subtotaal trackingmodule	**€ 290**	

Tabel 6.12: Indicatieve hardware-aankoopkost van de markerinfrastructuur. Deze kost omvat de passieve retroreflectieve markers die nodig zijn om een bruikbare plafondkaart op te bouwen.

Component	**Prijs**	**Bron**
Retroreflectieve markers	€ 23	[53]
Basis functionerend pakket	**€ 313**	

Tabel 6.13: Optionele uitbreidingen die niet meegerekend worden in het basispakket. Deze componenten verhogen gebruiksgemak of opslagcapaciteit, maar zijn niet noodzakelijk voor tracking via de basisopstelling.

Component	**Prijs**	**Bron**
8”-touchscreen	€ 50	[46]
NVMe SSD Kit 256 GB	€ 60	[54]

De trackingmodule zelf kost € 290. Samen met de retroreflectieve markers kost een functionele opstelling € 313. Dit is het minimale functionerende pakket: de markeromgeving is noodzakelijk voor tracking, maar het touchscreen en de NVMe SSD niet. Het touchscreen verbetert de bediening en visualisatie op de rig, maar het prototype kan ook via SSH vanuit een externe laptop bediend worden. De SSD is eveneens een comfort- en opslaguitbreiding zonder impact op de trackingprestaties en wordt daarom niet in de minimumprijs meegerekend.

6.8.2 Vergelijking met commerciële systemen

Tabel 6.14 plaatst de totale indicatieve hardware-aankoopkost van het prototype naast enkele commerciële inside-out cameratrackingsystemen. De prijzen voor het prototype zijn zelf opgeteld uit de componentprijzen in tabellen 6.11 en 6.12. De commerciële prijzen zijn afkomstig van publieke productpagina’s of officiële verdelers. De nauwkeurigheidskolom combineert de gemeten stationare afwijking van het prototype met gepubliceerde specificaties van commerciële systemen; die waarden zijn daarom indicatief en niet rechtstreeks methodologisch gelijkwaardig.

Tabel 6.14: Vergelijking van de indicatieve hardware-aankoopkost voor inside-out cameratracking in virtual production. De tabel plaatst het prototype naast commerciële systemen op basis van publiek beschikbare prijs- en specificatie-informatie [10, 20, 19, 11, 12].

Systeem	**Kost**	**Nauwk.**	**Opmerking**
Dit prototype	**€ 313**	< 0,2 mm*; < 0,01°*	Trackingmodule, passieve markers
Antilatency Basic Kit	>€ 2600	<1 mm; <0,1°	Trackingmodule, actieve markers
Stype RedSpy Air	>€ 7.000	<0,1 mm; <0,003°	Beperkt oppervlak
Stype RedSpy Air Plus	>€ 18.000	<0,1 mm; <0,003°	Software-ontgrendeling
Mo-Sys StarTracker Mini	>€ 14.000	<1 mm; <0,1°	Trackingmodule, passieve markers
Mo-Sys StarTracker Max	>€ 28.000	0,01 mm; 0,001°	Basis tracking-kit
Stype RedSpy 5.0 Pro	>€ 38.000	<0,1 mm; <0,003°	Volledige broadcast-kit

* Maximale poseafwijking bij stilstand, er kan geen juiste analyse gemaakt worden over de absolute fout van het prototype.

Van de commerciële systemen in tabel 6.14 ligt alleen de Antilatency Basic Kit in een prijsklasse die nog enigszins in de buurt van het prototype komt. Die prijs omvat de actieve markerinfrastructuur, maar het blijft een *entry-level* systeem zonder *genlock sync*, *lens encoder* ondersteuning en *ethernet* verbinding. Wanneer zulke uitbreidingen nodig zijn, loopt de systeemprijs op tot meer dan €5000. De Stype RedSpy Air-prijs is bovendien een gereduceerde indie-studioprijs en is daarom slechts beperkt vergelijkbaar met een academisch prototype. De Air Plus-, Mo-Sys- en professionele Stype-systemen bevinden zich in een andere marktklasse en dienen vooral als context voor de bovengrens van commerciële *startrackers*.

6.8.3 Bespreking van de kostenanalyse

De kostenanalyse toont vooral dat een werkende inside-out markertracker met gemakkelijk verkrijgbare hardware kan worden opgebouwd voor een veel lager bedrag dan commerciële systemen. Dat resultaat moet nuchter geïnterpreteerd worden. Het prototype is geen commercieel product en mist verschillende voorzieningen die in professionele virtual-productionomgevingen belangrijk zijn.

De sterke punten zijn de lage stationaire jitter, de lage materiaalkost, de reproduceerbare componentkeuze en het gebruik van goedkope passieve markers. De nauwkeurigheidsvalidatie blijft relatief ten opzichte van de Vive Tracker en is dus geen absolute foutmeting. Toch blijft de interpretatie relevant: de gemeten relatieve positieovereenkomst, met een gemiddelde afwijking rond 1,5 cm en een p95 onder 3 cm voor normale trajecten, ligt in een bereik dat voor veel virtual-productiontoepassingen voldoende is voor geloofwaardige parallax en camerabeweging.

De belangrijkste beperkingen zijn de afhankelijkheid van cameraupdates, de vertraging tussen camerabeeld en pose-uitvoer, en het ontbreken van *genlock*-achtige oplossingen die zulke vertragingen in professionele ketens kunnen beheersen. Daarnaast is er geen ondersteuning voor *lens encoders*, waardoor lensafhankelijke parameters zoals zoom en focus niet automatisch in de virtuele camera worden meegenomen. De kostenwinst is daardoor reëel, maar mag niet gelezen worden als gelijkwaardigheid met commerciële systemen. Het prototype toont vooral aan welke kernfunctionaliteit haalbaar is met goedkope hardware, en welke technische onderdelen nog ontbreken voor een professioneel inzetbaar cameratrackingsysteem.

Hoofdstuk 7

Conclusie en toekomstig werk

7.1 Conclusie

Dit onderzoek onderzocht in welke mate een kostenefficiënt inside-out cameratrackingsysteem met passieve retroreflectieve markers bruikbaar is voor virtual production. Daartoe werd een volledig werkend prototype ontworpen, gebouwd en geëvalueerd, gebaseerd op consumentgerichte hardware voor een totaalprijs van € 312,70, inclusief trackingmodule en passieve markerinfrastructuur.

De geometrische kwaliteit van de opgebouwde sterkaart werd gevalideerd via vergelijking met een onafhankelijke LiDAR-referentie. Over een meetgebied van 4,5 m bij 3,0 m werd een RMSE van 2,7 mm bereikt. Deze geometrische nauwkeurigheid is voldoende als metrische referentie voor de poseberekening, en daarom ook voldoende voor een virtual-production-opstelling.

De pose-uitvoer werd gevalideerd via vergelijking met een Vive Tracker die op dezelfde opstelling was gemonteerd. Over drie onafhankelijke installaties bleef de p95 van de positieafwijking die voor alle normale validatieruns onder 3 cm. De yawafwijking bleef consistent onder 1°. Er werd geen structurele drift waargenomen: de afwijking oscilleert rond een beperkt gemiddelde zonder cumulatief op te lopen, wat wijst op stabiele tracking.

Dit prototype bevat de kernfunctionaliteit die nodig is voor een bruikbaar resultaat in een kleinschalige of educatieve virtual-production-opstelling: inside-out tracking met passieve markers, gefilterde realtime pose-uitvoer, en een visueel stabiele werking op een LED-wall. Door gangbare conventies zoals FreeD en het verwachte coördinatenstelsel te ondersteunen, kan het prototype rechtstreeks in een bestaande renderomgeving worden gebruikt. De stabiele Vive-overeenkomst en de lage drift tonen aan dat de nauwkeurigheid volstaat voor gecontroleerde camerabewegingen waarbij geloofwaardige parallax belangrijker is dan sub-millimeternauwkeurigheid.

7.2 Toekomstig werk

Op basis van de bevindingen en de bekende beperkingen van het prototype worden de volgende richtingen voor verder onderzoek en ontwikkeling geïdentificeerd.

Validatie met een referentiesysteem van hogere nauwkeurigheid. De huidige validatie tegenover een Vive Tracker volstaat voor een eerste evaluatie, maar levert geen absolute referentie op studioniveau. Een volgende stap is daarom een validatie met een nauwkeuriger optisch meetsysteem, bijvoorbeeld Qualisys, zodat de absolute positie- en oriëntatiefout van het prototype eenduidiger kan worden bepaald.

Lagere beeldverwerkingslatentie via optische IR-filtering. De huidige verwerkingsvertraging in de beeldpipeline ligt nog hoger dan bij professionele systemen. Een optisch IR-filter dat hoofdzakelijk infraroodlicht doorlaat, kan de beeldvoorverwerking vereenvoudigen doordat minder niet-relevante informatie verwerkt moet worden. Dat kan de verwerkingslatentie verlagen zonder dat daar meteen duurdere rekenhardware voor nodig is.

ESKF-gebaseerde begeleiding van het NN-zoekvenster. De huidige nearest-neighbour-tracker gebruikt een vast zoekvenster rond de gereprojecteerde positie, ongeacht de verwachte bewegingsrichting. Door de ESKF-positiepredictie te gebruiken om per ster een richtingsgewijs zoekgebied op te bouwen vanuit de laatste gekende pose, kan de effectieve snelheidslimiet omhoog.

Fellere ingebouwde IR-belichting. De huidige IR-ledring biedt voldoende licht voor het centrale werkgebied maar presteert minder goed bij randmarkers. Het gedrag bij grotere plafondafstanden is niet getest en daarmee niet gekend. Een krachtigere, meer geconcentreerde IR-lichtbron ingebouwd in de behuizing zou de detectiebetrouwbaarheid aan de grenzen van het meetgebied verbeteren en het systeem geschikt maken voor een ruimer bereik van plafondhoogten.

Integratie van lens encoders. Het prototype bepaalt momenteel enkel de camerapose. Voor inzet in een volwaardige virtual-production-workflow is het zinvol om ook lensparameters zoals focus, zoom en iris mee uit te lezen, zodat de renderpipeline niet alleen de positie en oriëntatie van de camera, maar ook de optische toestand van de lens kan volgen.

Uitgebreide parametertuning. Zowel de detectieparameters (drempelwaarden, morfologische instellingen, circulariteitsgrens), de ESKF-ruismatrices als de *One Euro*-parameters zijn in dit prototype conservatief ingesteld. Systematische tuning via geoptimaliseerde validatietrajecten zou de nauwkeurigheid, de robuustheid en de balans tussen jitter en vertraging verder kunnen verbeteren zonder hardwarewijzigingen.

Gecombineerde voeding. Het prototype wordt volledig door de Raspberry Pi gevoed. Enkel de IR-ledring vereist 12 V en heeft daarvoor een aparte voedingskabel nodig, wat resulteert in twee stroomkabels. Door een 12 V-voeding te voorzien en de Raspberry Pi daar ook uit te voeden via een ingebouwde spanningsomvormer, kan het systeem worden teruggebracht tot één stroomkabel, wat de praktische bruikbaarheid in een studioomgeving verbetert.

Literatuurlijst

[1] "Virtual Production," 2026. Geraadpleegd op 30 mei 2026.

[2] "XR Glossary," 2026. Geraadpleegd op 6 juni 2026.

[3] "OptiTrack Motion Capture Systems," 2026. Geraadpleegd op 30 mei 2026.

[4] OptiTrack, *OptiTrack for Virtual Production*, 2026.

[5] Vicon, *Virtual Production Camera Tracking*, 2026.

[6] "Body suit tracking demonstration," 2026. Geraadpleegd op 6 juni 2026.

[7] Mo-Sys Engineering, *StarTracker: In-studio Optical Camera Tracking System*, 2022.

[8] Mo-Sys Engineering, *StarTracker Max: Optical Camera Tracking System*, 2024.

[9] Stype, *The Stype Book: Tracking and rendering systems for leading-edge XR, AR and VR production*, 2023.

[10] "Antilatency for Virtual Production," 2026. Geraadpleegd op 30 mei 2026.

[11] "Mo-Sys StarTracker Max," 2026. Geraadpleegd op 1 juni 2026.

[12] "Stype RedSpy 5.0 Pro Broadcast Package," 2026. Geraadpleegd op 1 juni 2026.

[13] "StarTracker – In-Studio Optical Camera Tracking," 2026. Geraadpleegd op 30 mei 2026.

[14] "Mo-Sys Unveils StarTracker Max Features," 2026. Geraadpleegd op 1 juni 2026.

[15] "RedSpy 5.0 Brochure," 2026. Geraadpleegd op 1 juni 2026.

[16] "Quick Start," 2026. Geraadpleegd op 1 juni 2026.

[17] M. Wuttke, *StarTracker Max: Optical Camera Tracking System Manual*. Mo-Sys Engineering, 2023.

[18] "Alt Tracking Hardware Specifications," 2026. Geraadpleegd op 1 juni 2026.

[19] "Mo-Sys StarTracker Mini," 2026. Geraadpleegd op 1 juni 2026.

[20] "The World's Most Popular Camera Tracking System Gets Even Better," 2026. Geraadpleegd op 1 juni 2026.

[21] "RedSpy," 2026. Geraadpleegd op 1 juni 2026.

[22] B. Kirollos and T. Povey, "High-accuracy infra-red thermography method using reflective marker arrays," *Measurement Science and Technology*, vol. 28, no. 9, p. 095405, 2017.

[23] H. Wang, Q. He, G. Guan, B. Leng, and D. Zeng, "The Fast Method for Correction of Distortion on Infrared Marker-Based Tracking System," *International Journal on Smart Sensing and Intelligent Systems*, vol. 6, no. 1, pp. 259–277, 2013.

[24] D. Berga Garreta, "IRMA: Real-time 3D tracking based on Infrared Reflective Markers for large spaces," bachelor's thesis, Universitat Pompeu Fabra, 2014.

[25] Q. Lin, R. Yang, Z. Zhang, K. Cai, Z. Wang, M. Huang, J. Huang, Y. Zhan, and J. Zhuang, "Robust Stereo-Match Algorithm for Infrared Markers in Image-Guided Optical Tracking System," *IEEE Access*, vol. 6, pp. 52421–52433, 2018.

[26] Q. N. Xing, D. Y. Yan, X. M. Hu, J. Q. Lin, and B. Yang, "Real-Time Identification and Tracking of Infrared Markers Based on Kalman Filter," *Applied Mechanics and Materials*, vol. 249–250, pp. 1147–1153, 2012.

[27] N. Poulopoulos and E. Z. Psarakis, "A blobs detection algorithm based on a simplified form of the fast radial symmetry transform," in *2017 22nd International Conference on Digital Signal Processing (DSP)*, pp. 1–5, IEEE, 2017.

[28] Y. Ou, H. Deng, Y. Liu, Z. Zhang, X. Ruan, Q. Xu, and C. Peng, "A Fast Circle Detection Algorithm Based on Information Compression," *Sensors*, vol. 22, no. 19, p. 7267, 2022.

[29] H.-B. Lee, J.-W. Kim, and Y.-H. Seo, "High Accurate Coordinates Estimation of IR Markers Captured by Multiple Sensors," *IEEE Sensors Journal*, vol. 25, no. 16, pp. 31053–31064, 2025.

[30] P. McIlroy, S. Izadi, and A. Fitzgibbon, "Kinectrack: 3D Pose Estimation Using a Projected Dense Dot Pattern," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 6, pp. 839–851, 2014.

[31] H. Uchiyama and Y. Oyamada, "Transparent Random Dot Markers," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 254–259, IEEE, 2018.

[32] R. S. Zeynalov, A. A. Yakubenko, and A. S. Konushin, "The matching of infrared markers for tracking objects using stereo pairs," *Pattern Recognition and Image Analysis*, vol. 23, no. 4, pp. 468–473, 2013.

[33] H. Wu, Q. Lin, R. Yang, Y. Zhou, L. Zheng, Y. Huang, Z. Wang, Y. Lao, and J. Huang, "An Accurate Recognition of Infrared Retro-Reflective Markers in Surgical Navigation," *Journal of Medical Systems*, vol. 43, no. 6, 2019.

[34] T. Collins and A. Bartoli, "Infinitesimal Plane-Based Pose Estimation," *International Journal of Computer Vision*, vol. 109, no. 3, pp. 252–286, 2014.

[35] G. Terzakis and M. Lourakis, "A Consistently Fast and Globally Optimal Solution to the Perspective-n-Point Problem," in *Computer Vision – ECCV 2020*, pp. 478–494, Springer International Publishing, 2020.

[36] S. Li, C. Xu, and M. Xie, "A Robust O(n) Solution to the Perspective-n-Point Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1444–1450, 2012.

[37] S. Zhuang, Z. Zhao, L. Cao, D. Wang, C. Fu, and K. Du, "A Robust and Fast Method to the Perspective-n-Point Problem for Camera Pose Estimation," *IEEE Sensors Journal*, vol. 23, no. 11, pp. 11892–11906, 2023.

[38] X. Xie and D. Zou, "Depth-Based Efficient PnP: A Rapid and Accurate Method for Camera Pose Estimation," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 9287–9294, 2024.

[39] N. Kadner, "Camera Tracking for Virtual Production," *American Cinematographer*, pp. 64–72, 2025.

[40] M. Agistri, "Star tracker measurement filtering via UF/EKF with recursive prefiltering of IMU reading," master's thesis, Politecnico di Milano, 2021.

[41] "Raspberry Pi Active Cooler," 2026. Geraadpleegd op 30 mei 2026.

[42] "Raspberry Pi M.2 HAT+," 2026. Geraadpleegd op 30 mei 2026.

[43] "Raspberry Pi Camera 3 Wide NoIR," 2026. Geraadpleegd op 30 mei 2026.

[44] "Adafruit 9-DOF Orientation IMU Fusion Breakout – BNO085," 2026. Geraadpleegd op 30 mei 2026.

[45] "KKSB Case for Raspberry Pi 5 and M.2 NVMe HAT," 2026. Geraadpleegd op 30 mei 2026.

[46] "8 Inch Touchscreen IPS Display 1280x800 Small Portable Monitor," 2026. Geraadpleegd op 31 mei 2026.

[47] P. Furgale, J. Rehder, and R. Siegwart, "Unified Temporal and Spatial Calibration for Multi-Sensor Systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1280–1286, IEEE, 2013.

[48] G. Casiez, N. Roussel, and D. Vogel, "1 Euro Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2527–2530, ACM, 2012.

[49] L. Kuhlmann de Canaviri, K. Meiszl, V. Hussein, P. Abbassi, S. D. Mirraziroudsari, L. Hake, T. Potthast, F. Ratert, T. Schulten, M. Silberbach, Y. Warnecke, D. Wiswede, W. Schiprowski, D. Heß, R. Brüngel, and C. M. Friedrich, "Static and Dynamic Accuracy and Occlusion Robustness of SteamVR Tracking 2.0 in Multi-Base Station Setups," *Sensors*, vol. 23, no. 2, p. 725, 2023.

[50] S. Merker, S. Pastel, D. Bürger, A. Schwadtke, and K. Witte, "Measurement Accuracy of the HTC VIVE Tracker 3.0 Compared to Vicon System for Generating Valid Positional Feedback in Virtual Reality," *Sensors*, vol. 23, no. 17, p. 7371, 2023.

[51] "Raspberry Pi 5 16GB RAM – Pricewatch," 2026. Geraadpleegd op 1 juni 2026.

[52] "24 Grain IR LED Board for Surveillance Cameras," 2026. Geraadpleegd op 1 juni 2026.

[53] "3M White/Silver Reflective Circles – 1m Roll, 30mm Circles," 2026. Geraadpleegd op 1 juni 2026.

[54] "Raspberry Pi NVMe SSD Kit 256GB," 2026. Geraadpleegd op 1 juni 2026.

Bijlage A

Configuratieparameters

Tabel A.1 tot en met tabel A.5 geven een overzicht van de belangrijkste parameters van het prototype zoals gebruikt in de huidige prototypeconfiguratie en de evaluaties in dit onderzoek.

Tabel A.1: Cameraparameters die door de beeldverwerking en poseberekening worden gebruikt. De tabel groepeert resolutie, intrinsieke parameters, distortiecoëfficiënten en de ingestelde plafondhoogte.

Parameter	**Waarde**	**Beschrijving**
Resolutie	2304×1296	Opnameresolutie in pixels
Framerate	20 fps	Beeldfrequentie
Sluitertijd	500 μs	Vaste belichting
Analoge gain	0,0	Sensorversterking
Scherpstelmodus	Handmatig	Autofocus uitgeschakeld
Lensstand	10,0	Vaste lensstand (geen eenheid)

Tabel A.2: Parameters van de lichte runtime-detector (`StarDetectorLight`). Deze detector is geoptimaliseerd voor continue tracking en gebruikt een beperkte set bewerkingen om de verwerkingstijd laag te houden.

Parameter	**Waarde**	**Beschrijving**
Downsamplingfactor	1	Detectie op volle resolutie
Gaussische blur (kernel)	3	Ruisonderdrukking
Drempelwaarde	150	Vaste binairisatiedrempel
Minimale bloboppervlakte	200 px^2	Ondergrens voor detectie
Maximale bloboppervlakte	1100 px^2	Bovengrens voor detectie
Minimale circulariteit	0,60	Compactheidseis

Tabel A.3: Parameters van de volledige detector (`StarDetector`). Deze detector wordt gebruikt voor kaartopbouw en debugbeelden en voert extra stappen uit voor robuustere markerselectie.

Parameter	**Waarde**	**Beschrijving**
Maximale saturatie (witheidsmasker)	120	HSV-S-kanaal
Minimale helderheid (witheidsmasker)	140	HSV-V-kanaal
Achtergrondschatting kernel	31	Volle-resolutie-equivalent voor morfologische opening
Minimale Otsu-drempel (*peak_floor*)	8	Ondergrens voor Otsu-drempelwaarde na achtergrondaftrekking
Maximale Otsu-drempel (*peak_ceil*)	60	Bovengrens; voorkomt dat een lichtbron de Otsu-drempel te hoog drijft
Minimale bloboppervlakte	$200\,\mathrm{px}^2$	Ondergrens voor detectie
Maximale bloboppervlakte	$1000\,\mathrm{px}^2$	Bovengrens voor detectie
Minimale circulariteit	0,60	Compactheidseis

Tabel A.4: Parameters voor posekwaliteit en plausibiliteitscontrole. De tabel bevat de grenswaarden waarmee localisatie, tracking, reprojectiefout, inliers en poseplausibiliteit worden beoordeeld.

Parameter	**Waarde**	**Beschrijving**
NN-zoekvenster (tracking)	50 px	Max. pixelafstand bij NN-matching met geprojecteerde kaartsterren
Minimaal aantal trackingsterren	5	Minimumaantal matches voor continue tracking; absolute ondergrens is 3
Maximale rotatiesprong	10°	Maximale rotatieverandering per frame tijdens tracking
Tracking-reprojectiegate	`maxReprojM` · f_x	Pixelgrens voor herlocalisatie tijdens tracking
RANSAC-iteraties (localisatie)	250	Aantal posehypotheses bij initiële localisatie met SQPnP
RANSAC-inliervenstering	5,0 px	Max. reprojectiefout om als inlier te gelden binnen RANSAC
PoseEstimator-validatie	<8 px, ≥3 inliers	Hard gate voor `CameraPoseMeasurement.valid`
Verdict Correct	≥70 %, <3 px	Matchrate en reprojectiefout
Verdict Probable	≥50 %, <5 px	
Verdict Partial	≥30 %, <8 px	
ESKF-initialisatie	≥4 inliers, <5 px	Strengere gate voor eerste visuele ESKF-initialisatie
ESKF-sigmaschaling	$e_{\text{reproj}}/2{,}5$	Meetruis wordt verhoogd bij reprojectiefout boven 2,5 px
Z-saniteitscontrole	$0 < -z \leq h_{\text{ceiling}}$	Anders wordt z teruggezet naar de gekalibreerde plafondafstand
Kaartgrensmarge	0,75 m	Toegelaten overschrijding van kaartgrens

Tabel A.5: FreeD-uitvoerparameters. Deze waarden bepalen hoe de interne trackingpose wordt omgezet naar het FreeD-pakket dat naar de renderomgeving wordt verzonden.

Parameter	**Waarde**	**Beschrijving**
ESKF-updatefrequentie	100 Hz	Dead-reckoning tussen cameraframes
Hoekcodering	×32768	Graden naar signed 24-bit integer
Positiecodering	×64 000	Meter naar fixed-point (mm×64)
Protocol	UDP	FreeD D1-pakket

Bijlage B

Code of Conduct AI

Gedragscode voor gebruik van generatieve AI-tools bij academische opdrachten

IIW, UHasselt & KU Leuven, 2025-2026 (versie 2026-01-30)

Dit formulier ondersteunt studenten en docenten bij **transparant, verantwoord en ethisch** gebruik van Generatieve AI (GenAI) bij academische opdrachten.

ALS STUDENT WORDT VAN JOU VERWACHT DAT JE:

- deze gedragscode **zorgvuldig leest**, zodat je weet welke vormen van GenAI-gebruik zijn toegestaan voor de opdracht en onder welke voorwaarden;
- **aanvinkt** welke vormen van toegelaten GenAI-gebruik werden toegepast bij het uitwerken van het ingediende werk en zo ja, **welke tools** werden gebruikt;
- een volledig **overzicht bijhoudt van interacties met GenAI-tools** en dit kan voorleggen op vraag van de docent, maar enkel wanneer **"Ja, indien gelogd"** is aangevinkt;
- dit formulier **dateert en ondertekent** om formeel te bevestigen dat GenAI-tools op een correcte manier zijn gebruikt bij het uitwerken van het ingediende werk;
- het ondertekende formulier **toevoegt aan het ingediende werk**, waardoor het niet nodig is om gedetailleerde duiding over GenAI-gebruik in het werk zelf op te nemen of de gebruikte tools in de literatuurlijst te vermelden, tenzij de docent anders aangeeft.

OPO:	7833 Masterproef – Informatica 2857 Masterproef – Elektronica-ICT
Promotor(en):	Prof. dr. Nick Michiels
Opdracht:	Masterscriptie
Datum of periode:	Academiejaar 2025-2026

Vormen van gebruik	Toegelaten door docent?	Richtlijnen bij toegestaan gebruik OF Verklaring waarom gebruik niet is toegestaan	Gebruikt door student?	Tools gebruikt door student
Brainstormen over nieuwe ideeën, mogelijk aanpakken …	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De student gaat na of het idee effectief **nieuw** is en of de gekozen aanpak **geschikt** is. De GenAI-suggesties zijn waarschijnlijk gebaseerd op bestaand werk, dat in dat geval moet worden vermeld in de **referenties**.	☐ Nee ☒ Ja	Claude, Codex
Als een **zoektool** om informatie te vinden over een onderwerp	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De student houdt er rekening mee dat de **betrouwbaarheid** van resultaten niet gegarandeerd is en de ecologische impact van GenAI groot is. De student moet resultaten altijd **verifiëren** aan de hand van niet-probabilistische informatiebronnen.	☐ Nee ☒ Ja	Claude, Codex, Copilot
Ondersteunen bij **literatuurstudie**	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	Enkel **zoeken** naar en **verwerken** (samenvatten, analyseren) van literatuur. De student wordt verwacht **zelf inzichten** uit de literatuur te halen en de literatuurstudie **zelf te schrijven**. GenAI kan referenties, data of inzichten verzinnen of verkeerd interpreteren, dus de student **controleert** deze altijd.	☐ Nee ☒ Ja	Claude, Codex
Kleine aanpassingen (spelling, grammatica, interpunctie, format) aan **zelfgecreëerde** inhoud (bv. tekst, slides, data, code)	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De oorspronkelijke inhoud moet **door de student zelf** worden gecreëerd. Het gebruik van GenAI is beperkt tot kleine aanpassingen die de student altijd moet **verifiëren**, aangezien GenAI-tools fouten maken.	☐ Nee ☒ Ja	Claude, Codex
Aanzienlijke aanpassingen of **refactoring** die stijl, structuur of helderheid van **zelfgecreëerde** inhoud (bv. tekst, slides, data, code) beïnvloeden	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De oorspronkelijke inhoud moet **door de student zelf** worden gecreëerd. De student moet suggesties **kritisch beoordelen**, aanpassingen **verifiëren** en ervoor zorgen dat deze **niet verder gaan dan** de oorspronkelijke inhoud en diens **betekenis of aanpak niet veranderen**.	☐ Nee ☒ Ja	Claude, Codex
Genereren van **visuele elementen** (grafieken, diagrammen of andere vormen van visualisatie, mock-ups, tekeningen, video's, animaties)	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De eigenlijke inhoud en beslissingen (bv. geschikte visualisatie, het script, ontwerpideeën) moeten **van de student zelf** komen. De student **controleert** de visuele elementen en ziet erop toe dat ze origineel zijn en geen auteursrechten schenden. Visuele elementen moeten de onderliggende werkelijkheid **waarheidsgetrouw** weergeven, indien van toepassing.	☒ Nee ☐ Ja	

Boilerplate codefragmenten genereren (bv. scripts voor inlezen, verwerken en analyseren van data, of uitvoeren van triviale berekeningen)	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De student moet alle ingediende code **verifiëren, begrijpen en kunnen toelichten**, inclusief de onderliggende keuzes. Hou er rekening mee dat GenAI foutieve of suboptimale oplossingen kan genereren.	☐ Nee ☒ Ja	Claude, Codex
Vertalen van geschreven inhoud of **transcriberen** van gesproken inhoud (bv. interviews)	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De student is verantwoordelijk voor het **controleren van de output** op nauwkeurigheid, toon en context. Voor transcriberen van gesproken inhoud moet de student eerst **toestemming** verkrijgen van de spreker.	☒ Nee ☐ Ja	
Vragen om feedback op jouw werk	☐ Nee ☒ Ja ☐ Ja, indien gelogd ☐ Niet van toepassing	De student moet **kritisch reflecteren** over de feedback alvorens deze te verwerken in overeenstemming met de toegelaten vormen van GenAI gebruik. Vergeet niet dat GenAI feedback kan genereren die **niet in lijn ligt met de richtlijnen van de opdracht**.	☒ Nee ☐ Ja	
Genereren van volledige oplossingen of **stukken inhoud** (bv. volledige alinea's tekst of volledige berekeningen, algoritmes of ontwerpen)	**NIET TOEGESTAAN**	Het examenreglement bepaalt dat de docent bij de beoordeling van het werk van de student in staat moet zijn om **de kennis, het begrip en de vaardigheden van de student zelf correct te beoordelen**.	NVT	NVT
Andere vormen van gebruik (aan te vullen door docent)	☐ Nee ☐ Ja ☐ Ja, indien gelogd ☒ Niet van toepassing	(aan te vullen door docent)	☐ Nee ☐ Ja	NVT

Ik verklaar hierbij dat:

- ik geen GenAI-assistentie heb gebruikt bij gegevens of onderwerpen onder een geheimhoudingsverklaring (NDA), auteursrechtelijk beschermd materiaal, of gevoelige of persoonlijke gegevens omwille van privacy. Ik heb **voorafgaandelijke toestemming** verkregen van de docent, supervisor of externe partij om aangereikte gegevens in een GenAI-tool in te voeren;
- het ingediende werk **mijn eigen kennis, begrip en vaardigheden weerspiegelt**. Ik ben me ervan bewust dat de docent mijn kennis, begrip en vaardigheden kan controleren in een format waarin GenAI-gebruik is uitgesloten;
- ik begrijp dat als ik deze gedragscode over het gebruik van GenAI niet naleef, dit kan worden beschouwd als een **onregelmatigheid** volgens het onderwijs-, examen- en rechtspositiereglement (OER). In dat geval kan een procedure inzake onregelmatigheden worden opgestart (art. 18.1-18.3 OER), wat kan leiden tot **sancties** zoals een aangepast eindresultaat.
- ik een **kritische en wetenschappelijke houding** heb aangenomen bij het gebruik van GenAI. Ik ben me bewust van de neiging van GenAI om **te hallucineren, te misleiden, te behagen, vooroordelen te vertonen, enz.** en dat GenAI resultaten kan genereren die niet in overeenstemming zijn met de richtlijnen van de opdracht;
- ik me ervan bewust ben dat ik altijd de **volledige verantwoordelijkheid** draag voor de inhoud van het werk dat ik indien. Eventuele fouten of vergissingen kunnen niet worden toegeschreven aan de gebruikte GenAI-tools;
- ik weet dat ik meer informatie kan vinden over het **gebruik van GenAI** via de UHasselt en KU Leuven pagina's;
- ik **bij twijfel contact opneem met de docent**, gezien GenAI-tools snel evolueren en nieuwe mogelijkheden kunnen leiden tot nieuwe regelgevende onzekerheden, die het best open en transparant worden aangepakt.

Naam student(en):	Leander Winters
Studentennummer(s):	2262219
Datum:	8/06/2026
Handtekening(en):	