Master thesis submitted to achieve the
master’s degree in CS, option Multimedia
at Hasselt University

An optical tracking system for the integration of input devices into head-mounted virtual reality environments using commodity hardware

Author:
Glenn Bergmans

Promotor:
Prof. Dr. Johannes Schöning

2015-2016

Abstract

The aim of this thesis is to provide a tracking method for the user to track their input device while using a head-mounted virtual reality display. Most currently available tracking solutions use expensive hardware unavailable to the average consumer, are specific to a single input device or are incapable of keeping up with the higher refresh rate required of virtual reality applications. The LeapTrack tracking system developed in this thesis seeks to remedy this with an inside-looking-out infra-red stereo tracking solution using commodity hardware, allowing the users to find their input device without breaking immersion.

This thesis first presents a brief introduction on virtual reality and its benefits, and then examines previous work that is related to the problem caused by the lack of vision while using a head-mounted virtual reality display. Following this, an overview of the techniques and hardware used in the LeapTrack tracking system is provided before presenting an implementation of said techniques in the Unity game engine. Finally a user study is conducted to evaluate the impact of LeapTrack on the ability to locate a peripheral device and the level of immersion of the user.

The results of this user study show that users respond positively to the presence of LeapTrack, increasing their accuracy when attempting to pick up the tracked peripheral while minimizing the negative impact on immersion.

Dutch summary

Introductie

Virtual Reality (VR) wordt hoofdzakelijk op twee technische manieren geïmplementeerd, namelijk als een head-mounted display (HMD) en als een Cave Automatic Virtual Environment (CAVE).
Een HMD is een beeldapparaat dat op het hoofd van de gebruiker gedragen wordt en laat een aangepast beeld zien aan ieder oog, met een stereoscopisch 3D effect tot gevolg. Door de positie van het hoofd van de gebruiker te volgen, kan bepaald worden welk deel van de virtuele omgeving (VO) getoond moet worden.
Een CAVE projecteert de VO op de ruimte rond de gebruiker. Een typische CAVE is een kamer van 3 m bij 3 m bij 3 m waarvan de muren, de vloer en soms het plafond gemaakt zijn van computer geprojecteerde schermen. Voor het 3D effect wordt gebruik gemaakt van een shutter-bril gesynchroniseerd met de projectors of schermen. Elk frame wordt dan gerenderd voor het oog dat op dat moment niet bedekt is door de bril.

De eerste HMD werd in 1968 gemaakt aan de universiteit van Harvard door Ivan Sutherland. Deze maakte gebruik van twee miniatuur CRTs om 3D wire-frame graphics te renderen in stereografisch 3D. Door gebruik te maken van halfdoorlatende spiegels was het mogelijk om een augmented reality display te maken. Head tracking was mogelijk met zowel een mechanische als een ultrasonische methode.
De eerste arcade toepassing van VR HMDs was de Virtuality in 1991. Deze machines gebruikten een HMD met twee LCD schermen met een resolutie van 276 x 372. Buiten de HMD werd de gebruiker ook uitgerust met een 3D joystick en een riem. Al deze componenten werden getrackt via elektromagnetische methoden.
Rond 1995 werden de eerste pogingen gedaan om de consumentenmarkt binnen te dringen. Het meest spraakmakende toestel van deze tijd was de Nintendo Virtual Boy (VB). Deze draagbare HMD had geen head tracking en diende op een metalen houder op een tafel geplaatst te worden. Met behulp van twee lineaire arrays van elk 224 rode LED's creëerde de VB monochroom rode graphics met een resolutie van 384 x 224 door elke array over het bijhorend oog te scannen met een platte oscilerende spiegel. De VB eindigde als een commerciële flop.
Pas in 2012 werd er opnieuw interesse getoond voor VR met de kickstarter van de Oculus Rift. Over de spanne van de maandlange campagne legden 9 522 backers US$2 437 429 bij. Het oorspronkelijke doel van US$250 000 werd binnen de 24 uur bereikt. 2016 zal het jaar zijn waarin commerciële VR een realiteit wordt, met de lancering van de Oculus Rift en de HTC Vive voor de pc, PSVR voor de Playstation 4 en de Samsung Gear VR al reeds beschikbaar voor de mobiele markt.

Er kan immers veel voordeel gehaald worden door VR toe te passen in verschillende domeinen. Zo kan het toepassen van VR op vlak van opleiding en training leiden tot het makkelijker en goedkoper kunnen inoefenen van motorische vaardigheden, betere toegang tot leerkrachten en studenten, meer gedetailleerde en interactieve feedback en meer aandacht van de studenten. VR training wordt al toegepast in onder meer de opleiding van chirurgen, soldaten, piloten en astronauten.
Therapie heeft ook baat bij VR. Virtual Reality Exposure Therapy (VRET) waarbij de patient blootgesteld wordt aan een virtuele simulatie van hun fobie is geëvalueerd als zijnde lichtjes beter dan exposure in vivo, wat een van de meest effectieve behandelingen is. VRET heeft als bijkomende voordelen dat het veiliger, sneller, goedkoper en beter beheersbaar is door de therapeut.
VR kan ook toegepast worden voor het visualiseren van big data. Door de patroonherkenning van mensen te gebruiken kunnen immers nieuwe verbanden ontdekt worden in de data. Zo wordt VR al gebruikt in het visualiseren van onder andere landschappen, steden, geografische data en het visualiseren en analyseren van hersenactiviteit.
Engineering design is ook een domein dat potentieel heeft voor VR toepassingen. Het bekijken van een ontwerp vanuit een VR standpunt geeft een betere indruk van grootte en schaal, en laat de designer toe om vanuit het perspectief van de eindgebruiker te kijken.
De grootste driver van VR is de game industrie. VR in gaming staat nog in zijn kinderschoenen, maar voorlopige studies laten zien dat gaming in VR de gebruiker meer immersie en plezier laat voelen dan bij traditionele games. Pretparken maken ook gebruik van VR, en met draadloze VR apparaten zoals de Gear VR wordt het mogelijk om VR in een bewegende attractie te gebruiken, zoals bij de VR rollercoaster in SixFlags.

Als de gebruiker een HMD op zet, blokkeert dit het zicht naar de buitenwereld. Dit wil zeggen dat het voor de gebruiker moeilijk is om hun invoerapparaat terug te vinden eenmaal zij zich in de virtueele realiteit bevinden. De Oculus Rift, HTC Vive en PSVR voorzien hun eigen controller die getrackt wordt door hun eigen tracking technologie, maar deze is niet compatibel met andere invoerapparaten. Om deze terug te vinden is het vaak noodzakelijk om de HMD af te zetten, wat nefast is voor de immersie. De HTC Vive is momenteel de enige HMD die een mogelijke oplossing biedt in de vorm van een camera aan de voorzijde. Dit voorkomt dat de gebruiker de HMD moet afzetten, maar vermindert nog steeds de immersie.

De ideale oplossing zou zijn om een gelijkaardige tracking beschikbaar te maken als degene die gebruikt wordt voor de eigen controllers van de HMDs. De oplossing gepresenteerd in deze thesis zet een stap in deze richting door gebruik te maken van een infrarood tracking systeem. Door een Leap Motion aan de voorzijde van de HMD te bevestigen en een aantal infrarood markers aan te brengen op het apparaat dat getrackt dient te worden, wordt het mogelijk om de rotatie en translatie van het object op te volgen. De Leap Motion maakt gebruik van twee infrarood cameras en drie infrarood LED’s voor hand tracking.

Literatuuronderzoek

Verschillende werken omtrent het tracken en integreren van invoerapparaten in VR werden bestudeerd. Deze werken kunnen in drie categorieën van tracking methodes worden onderverdeeld.
De eerste categorie gebruikt elektromagnetische tracking. Deze techniek is nauwkeurig en responsief, maar kan verstoord worden door elektrische velden van andere objecten. Visuele tracking methodes zoals degene gebruikt in onze implementatie hebben hier geen last van en kunnen gemaakt worden met goedkopere hardware. De bestudeerde papers met deze techniek zijn "The Virtual Tricorder" en "3D Palette: A Virtual Reality Content Creation Tool".
De tweede categorie gebruikt infraroodtracking. Onze tracking implementatie valt ook onder deze categorie. Wij verbeteren de besproken methodes op twee vlakken. Ten eerste gebruiken wij het perspectief van de gebruiker. Hierdoor kan het invoerapparaat getrackt worden in onverwachte posities, wat niet altijd mogelijk is met statische opstellingen. Ten tweede kan onze implementatie toegepast worden op elk object terwijl de besproken papers zich beperken tot een vooraf bekende opstelling. Deze besproken papers zijn "A new Optical Tracking System for Virtual and Augmented Reality Applications", "The Wii Remote as an input device for 3D interaction in immersive head-mounted display virtual reality" en "Ring-shaped Haptic Device with Vibrotactile Feedback Patterns to Support Natural Spatial Interaction".
De derde categorie gebruikt de Microsoft Kinect. Dit toestel bevat een RGB camera en een diepte camera. De Leap Motion die onze implementatie gebruikt bestaat uit twee infraroodcamera’s, maar heeft een groter gezichtsveld (70.6 x 60 graden voor de Kinect versus 132 x 115 voor de Leap Motion) en een hogere framerate (30 Hz voor de Kinect versus 150 Hz voor de Leap Motion). Gezien de aanbevolen framerate voor VR applicaties 90 Hz is, zou de data van de Kinect twee van de drie frames voorspeld moeten worden. De besproken papers die gebruik maken van de tracking van de Kinect zijn "A Dose of Reality: Overcoming Usability Challenges in VR Head-Mounted Displays", "Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences" en "Snake Charmer: Physically Enabling Virtual Objects".

Een alternatieve benadering is het gebruik van lichaamsinput. Aangezien de gebruiker altijd bewust is van zijn eigen lichaam, kan de gebruiker deze inputmethode nooit kwijt raken. Vier categorieën van lichaamsinput worden besproken.
Het tracken van het volledige lichaam geeft toegang tot postuursinformatie die enkel geschat kan worden met de tracking informatie van de HMD en laat interactie toe met lichaamsdelen die vaak niet getrackt worden, zoals de benen van de gebruiker. De paper "SpaceWalk:Movement and Interaction in Virtual Space with Commodity Hardware" wordt besproken als voorbeeld hiervan.
Hand tracking voorziet een intuïtieve manier om met VR te interageren, zowel via directe interactie met de VO zoals het oprapen van een virtueel voorwerp als via gesture controls.

Aangezien onze tracking de Leap Motion gebruikt, kan de hand tracking hiervan gebruikt worden samen met de tracking van het invoerapparaat. De papers "TranSection: Hand-Based Interaction for Playing a Game within a Virtual Reality Game", "The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR" en "Towards Effective Interaction with Omnidirectional Videos Using Immersive Virtual Reality Headsets" worden besproken als voorbeeld van input via hand tracking en het omgaan met de limitaties van deze inputvorm.
De blik van de gebruiker als invoer benutten is beperkt tot de hoofdoriëntatie, maar het is niet onwaarschijnlijk dat toekomstige HMD's oog tracking gaan integreren. "Towards Effective Interaction with Omnidirectional Videos Using Immersive Virtual Reality Headsets" benut de hoofdoriëntatie van de gebruiker, terwijl "Orbits: Gaze Interaction for Smart Watches using Smooth Pursuit Eye Movements" en "Augmenting the Field-of-View of Head-Mounted Displays" de oogpositie zelf gebruiken.
De laatst besproken vorm van lichaamsinput is steminput. Deze is nuttig voor simpele commando's die niet tijdsgevoelig zijn. Steminput wordt besproken aan de hand van de paper "Application of Speech Recognition Technology to Virtual Reality System".

Helaas ontbreekt er vaak fysieke feedback bij lichaamsinput. Draagbare invoerapparaten die bedienbaar zijn zonder zicht kunnen een oplossing bieden waarbij de gebruiker nog steeds bewust is van het invoerapparaat maar ook fysieke feedback heeft bij het gebruik ervan. De papers "Nenya: Subtle and Eyes-Free Mobile Input with a Magnetically-Tracked Finger Ring", "Belt: An Unobtrusive Touch Input Device for Head-worn Displays" en "iSkin: Flexible, Stretchable and Visually Customizable On-Body Touch Sensors for Mobile Computing" werden besproken in dit kader.

Concept

De hardware die vereist is voor de tracking bestaat uit een Leap Motion die bevestigd is aan de voorkant van een HMD. Een Leap Motion is in staat om in stereo infrarood te filmen met een resolutie van 640 x 240 pixels en een framerate van maximum 150 frames per seconde. Ook voorziet deze zijn eigen licht m.b.v. 3 infrarood LED's en wordt deze gebundeld met handtracking software. Op het apparaat dat getrackt dient te worden dienen minstens 3 infrarood markers aangebracht te worden in een niet-isogonaal patroon.

De software is bedoeld om minstens 1 maal per gerenderd frame aangeroepen te worden. Deze zal dan eerst de meest recente videodata ophalen van de Leap Motion. De rauwe sensordata wordt daarna doorzocht om pixels met voldoende helderheid te vinden. Telkens er een dergelijke pixel gevonden wordt, wordt de figuur waartoe deze behoord gezocht d.m.v een contourtracing algoritme. Het Moore-Neighborhood algoritme wordt hier gebruikt, welke de naburige pixels kloksgewijs controleert op zoek naar een pixel met een voldoende hoge helderheid, beginnend bij de pixel na de voorgaande pixel. Eenmaal een dergelijke heldere pixel gevonden is, wordt deze toegevoegd aan de omtrek van de figuur en de naburige pixels van deze nieuwe pixel worden vervolgens gecontroleerd. Wanneer het algoritme de originele pixel terugvindt weet het algoritme dat de omtrek van de figuur compleet is. De pixels binnen deze figuur kunnen genegeerd worden bij het zoeken naar andere figuren. Het centrum van deze figuur wordt berekend als het gemiddelde van elk punt op zijn omtrek en vervolgens opgeslagen.

De figuren gevonden in het beeld van de linker camera moeten vervolgens gepaard worden met de correcte figuren uit het beeld van de rechter camera om m.b.v. triangulatie het punt in de 3D ruimte te vinden waarop de eigenlijke marker zich bevindt. Gezien het hier om parallelle camera's gaat, zal een figuur van de linker camera zich op ongeveer dezelfde y coördinaat bevinden als zijn bijhorende figuur uit de rechter camera. Figuren op dezelfde y coördinaten worden gematcht van links naar rechts. Indien het aantal figuren van de linker camera gevonden op een bepaalde y-as niet gelijk is aan het aantal figuren van de rechter camera, wordt er gebruik gemaakt van de grootte van de figuur om te bepalen welke figuren geen match hebben.

Eenmaal de figuren gepaard zijn, kan het coördinaatpaar herleid worden naar een punt in 3D ruimte. Hiervoor wordt gebruik gemaakt van de volgende formules.

- $y_m = -40/(\tan(\alpha_r) - \tan(\alpha_l))$
- $x_m = (-y_m) * \tan(\alpha_r) - 20$
- $z_m = (-y_m) * \tan(\beta_r)$

(x_m, y_m, z_m) is hier de 3D coördinaat van de marker. $(\alpha_r, \beta_r)\ en\ (\alpha_l, \beta_l)$ zijn de horizontale (α) en verticale (β) hellingsgraad naar de middelpuntspixel van de figuur na correctie voor lensvervorming van respectievelijk het rechtse en linkse camerabeeld. De afstand tussen de twee camera's is 40 mm.

Deze verzameling van 3D coördinaten vormt het patroon dat gebruikt wordt om het invoerapparaat te herkennen. Een calibratie is echter vereist om de positie van dit patroon t.o.v. het invoerapparaat aan te leren. Tijdens deze calibratie dient de gebruiker het fysieke invoerapparaat op dezelfde plaats te houden als de virtuele representatie van dit invoerapparaat. Deze lijst van 3D punten kan vervolgens opgeslagen worden, samen met afstanden en hoeken tussen de punten.

De punten die gevonden worden na de calibratie kunnen met dit referentiepatroon vergeleken worden. De eerste stap in dit proces is het identificeren welke van de nieuwe punten overeenkomen met welke van de referentiepunten. Hiervoor wordt eerst gekeken naar de afstanden tussen de punten. Indien een afstand tussen twee nieuwe punten overeenkomt met een afstand tussen twee punten in het referentiepatroon, krijgt elk van de nieuwe punten een stem voor elk van de twee referentiepunten. Na het overlopen van alle puntcombinaties wordt elk referentiepunt toegekend aan het nieuwe punt dat de meeste stemmen heeft voor dat referentiepunt. Indien er een gelijke score is tussen twee of meer punten, wordt er een nieuwe stemronde gehouden op basis van de hoeken tussen deze punten en reeds geïdentificeerde punten.

Als minstens 3 punten geïdentificeerd zijn, kan m.b.v. het Kabsch algoritme de rotatiematrix berekend worden tussen de twee patronen. Bij het Kabsch algoritme worden eerst de twee puntensets getransleerd naar de oorsprong. Vervolgens wordt de covariantiematrix van deze twee puntensets berekend en ten slotte wordt de singulierewaardenontbinding van de covariantiematrix berekend. De rotatiematrix kan vervolgens verkregen worden door de resulterende rechter singuliere vectoren te vermenigvuldigen met de transpose van de linker singuliere vectoren.

De translatie tussen de twee puntensets wordt bepaald door het verschil tussen de twee zwaartepunten van de puntensets. De rotatie kan uit de rotatiematrix gehaald worden d.m.v. de volgende formules.

- $\theta_x = arctan(r_{32} \,/\, r_{33})$
- $\theta_y = -arcsin(r_{31})$
- $\theta_z = arctan(r_{21} \,/\, r_{11})$

$(\theta_x, \theta_y, \theta_z)$ zijn hier de 3 Eulerhoeken en r elementen van de 3x3 rotatiematrix.

Deze rotatie en translatie kan vervolgens gebruikt worden om de virtuele representatie van het invoerapparaat te transformeren van zijn calibratiepositie naar de huidige positie van het fysieke invoerapparaat.

Implementatie

De tracker werd geïmplementeerd in de Unity game engine en bestaat uit 4 classes en 1 Unity script, geschreven in C# gebruik makend van versie 2 van de Leap Motion SDK en de Math.NET Numerics library. Het Unity script wordt gekoppeld aan een gameobject dat zich onder de Leap Motion VR camera bevindt. Dit gameobject bevat een leeg Anchor object waaronder het virtueel inputapparaat geplaatst wordt. Het script wordt elke frame aangeroepen. Eerst zal het de recentste beelden van de Leap Motion camera's ophalen. Deze worden vervolgens in een instantie van de LeapImage klasse gebruikt om de markers van het fysieke inputapparaat te detecteren en om te zetten in 3D coördinaten.

Deze omzetting gebeurt door de beelden te doorzoeken naar pixels boven een bepaalde helderheidswaarde. Deze pixel wordt dan doorgegeven aan een ImageFeature object, welk m.b.v. het Moore-Neighborhood contourtracing algoritme de omtrek van de marker zoekt waar deze pixel aan toebehoort. Pixels die reeds aan een ImageFeature zijn toegewezen kunnen overgeslagen worden bij het doorzoeken van het beeld.

Met een lijst van ImageFeature objecten van het linker- en rechterbeeld kunnen deze vervolgens met elkaar gepaard worden. Hiervoor worden eerst beide lijsten op de y coördinaat van hun zwaartepunt gesorteerd. Per ImageFeature worden alle ImageFeatures op ongeveer dezelfde y coördinaat vervolgens op de x coördinaat van hun zwaartepunt gesorteerd. De ImageFeature objecten van het linkerbeeld kunnen dan op volgorde van x coördinaat gepaard met de ImageFeature objecten van het rechterbeeld. Indien er meer ImageFeature objecten gevonden zijn in één van de beelden, worden alle mogelijke paren van ImageFeature objecten die de x volgorde respecteren uitgeprobeerd en wordt er een score toegekend op basis van het verschil in oppervlakte van de ImageFeature objecten. De combinatie met de laagste score wordt behouden en de niet gepaarde ImageFeature objecten weggegooid. De twee zwaartepunten van elk paar ImageFeature objecten worden gecorrigeerd voor lensvervorming en vervolgens gebruikt om d.m.v. triangulatie het punt in 3D ruimte te bepalen waar de marker zich bevindt.

Het Unity script gebruikt deze lijst van 3D punten verkregen uit het ImageFeature object om tijdens de calibratiefase samen met een overlay van de Leap Motion de gebruiker te helpen om het fysieke object op de juiste locatie te brengen. Tijdens de calibratie zal deze lijst van punten gebruikt worden om een PatternFinder object te creëren. Ook wordt het verschil tussen het Anchor gameobject en het zwaartepunt van de coördinaten in deze lijst bijgehouden. Bij het aanmaken van een PatternFinder object wordt de lijst van 3D punten die het patroon vormen opgeslagen samen met de afstanden en hoeken tussen de punten. Bij de volgende update zal de nieuwe lijst met 3D punten vergeleken worden met het originele patroon in het PatternFinder object met als doel het verschil in rotatie en translatie te berekenen. Deze vergelijking begint door te achterhalen welke punten uit de nieuwe lijst overeenkomen met welke punten van het originele patroon. Hiervoor wordt eerst gekeken naar de afstanden tussen de punten. Indien het verschil in afstand tussen twee nieuwe punten en een afstand tussen twee punten in het originele patroon kleiner is dan de foutmarge, krijgt elk van de nieuwe punten een stem voor elk van de twee referentiepunten. Na het overlopen van alle puntcombinaties wordt elk referentiepunt toegekend aan het nieuwe punt dat de meeste stemmen heeft voor dat referentiepunt. Indien er een gelijke score is tussen twee of meer punten, wordt er een nieuwe stemronde gehouden op basis van de hoeken tussen deze punten en reeds geïdentificeerde punten.

Vervolgens kan de rotatiematrix berekend worden m.b.v. het Kabsch algoritme. Hiervoor worden twee lijsten van 3 x 1 matrices opgesteld. De eerste lijst bevat matrices met de coördinaten van de geïdentificeerde punten uit de nieuwe lijst, de tweede bevat matrices met de coördinaten van de overeenkomstige punten uit de originele lijst. Per lijst wordt er een zwaartepunt berekend. Elk element in de lijst wordt naar de oorsprong verplaatst door het bijhorende zwaartepunt hiervan af te trekken, en vervolgens opgeteld bij de covariantiematrix na vermenigvuldigen met de transpose van het overeenkomstige punt verplaatst naar de oorsprong uit de tweede lijst. Hierna wordt de singulierewaardenontbinding van de covariantiematrix berekend. De Math.NET library geeft deze terug in de vorm $U * S * V^T$. De rotatiematrix kan vervolgens verkregen worden door de transpose van V^T te vermenigvuldigen met de transpose van U.

Een Orientation object wordt gebruikt om de waarden nodig voor de transformatie terug te geven. De translatiewaarden worden verkregen door het verschil tussen de zwaartepunten van de twee lijsten te nemen. De rotatiewaarden worden uit de rotatiematrix genomen d.m.v. de volgende code.

```
orient.Yaw = Mathf.Rad2Deg * Mathf.Atan2(r[2, 1], r[2, 2]);
orient.Pitch = Mathf.Rad2Deg * Mathf.Atan2(-r[2, 0],
               Mathf.Sqrt(Mathf.Pow(r[2, 1], 2) + Mathf.Pow(r[2, 2], 2)));
orient.Roll = Mathf.Rad2Deg * Mathf.Atan2(r[1, 0], r[0, 0]);
```

De informatie in het Orientation object wordt in het Unity script gebruikt om de lokale positie van het Anchor gameobject te wijzigen. Deze wordt eerst verplaatst naar de som van het zwaartepunt van het originele patroon en de translatie uit het Orientation object. Vervolgens wordt deze geroteerd volgens de waarden uit het Orientation object. Ten slotte gebeurt er een translatie naar het eerder opgeslagen verschil tussen de originele positie van het Anchor gameobject en het zwaartepunt van het originele patroon.

Evaluatie

Om de invloed van LeapTrack te bepalen op het vermogen van de gebruiker om een invoerapparaat terug te vinden en de impact op de immersie vast te stellen, werd een gebruikerstest uitgevoerd. In dit experiment werd het vermogen van de gebruiker om een invoerapparaat zoals een game controller te bereiken getest terwijl de gebruiker in een VR applicatie zat. Twee condities werden getest, namelijk zonder enige tracking en met LeapTrack. LeapTrack stond de gebruikers toe om een virtuele representatie van het invoerapparaat te zien in de VO. 12 studenten, waarvan 7 mannelijk en 5 vrouwelijk, in de leeftijdscategorie van 20 tot 25 jaar namen vrijwillig deel aan het experiment.

Deelnemers werden eerst verzocht om plaats te nemen aan een leeg bureau. Er werd hen gevraagd een Oculus Rift DK2 HMD op te zetten en er werd hun een Xbox 360 controller aangegeven waaraan m.b.v. K'NEX aan een “antenne” bolvormige infraroodreflectoren bevestigd waren. De deelnemer kreeg dan de instructie om voorwaarts te bewegen met de linker analoge stick. Na een korte afstand kwam de gebruiker de eerste “wegversperring” tegen. Hier werd hun gevraagd om naar rechts te kijken en beide handen te plaatsen in een aangegeven gebied, welk daarna geleidelijk aan een groene kleur kreeg. Eenmaal het gebied van elke hand volledig groen gekleurd was, verdwenen deze gebieden en kreeg de deelnemer de instructie om hun controller op te nemen en verder voorwaarts te bewegen. Gedurende de loop van een uitvoering van het experiment kwamen de deelnemers 7 wegversperringen tegen. Tijdens de eerste wegversperring werd mondeling uitgelegd wat de deelnemer hierbij moest doen. Bij de latere wegversperringen kregen de deelnemers visuele hints in de VO, tenzij er om uitleg gevraagd werd. De tijd die de deelnemer nodig had om de controller terug te vinden werd bijgehouden, samen met het aantal pogingen dat de deelnemer deed om de controller vast te nemen. Tijdens de eerste 4 wegversperringen bleef de controller liggen waar de deelnemer deze had neergelegd. Gezien de deelnemer hier bewust was van de locatie van de controller, wordt er bij wegversperring 2, 3 en 4 gesproken over "bewuste" prestaties. Bij de laatste 3 wegversperringen werd de controller verplaatst terwijl de deelnemer zijn handen in de aangegeven gebieden hield. De deelnemer is zich dan niet langer bewust van de locatie van de controller en er wordt hier dan ook gesproken over "onbewuste" prestaties. Eenmaal de eerste uitvoering van het experiment voltooid was door alle 7 wegversperringen te voltooien en het einde van het parcours te bereiken, werd er aan de deelnemer gevraagd om hun immersie op een schaal van 1 tot 10 te beoordelen. 10 stelt hierbij het gevoel voor dat ze werkelijk aanwezig waren in de VO, terwijl 1 het gevoel voorstelt dat de deelnemer in een kamer zat en naar een HMD scherm keek.

Een tweede uitvoering werd hierna gestart met LeapTrack actief. Voordat de deelnemer de instructie kreeg om vooruit te bewegen, werd eerst gevraagd om de calibratie uit te voeren door de fysieke controller op dezelfde plaats te houden als een virtuele controller en te bevestigen door op een knop te duwen. Deze stap mocht herhaald worden tot de gebruiker tevreden was met de tracking. Het aantal pogingen werd bijgehouden. Er werd hun dan gevraagd om een SUS enquête mondeling in te vullen over de gebruiksvriendelijkheid van de calibratieprocedure. De rest van de uitvoering verliep identiek aan de uitvoering zonder LeapTrack actief. Na de voltooiing van de tweede uitvoering werd de deelnemer weer gevraagd om hun immersie te beoordelen op een schaal van 1 tot 10, en om een SUS enquête in te vullen over de gebruiksvriendelijkheid van de tracking.

Uit de resultaten blijkt dat met LeapTrack het gemiddeld aantal pogingen om de controller op te rapen daalt van 1.17 naar 1.03 tijdens bewuste prestaties en van 3.64 naar 1.39 tijdens onbewuste prestaties. Een T-test laat zien dat de metingen van bewuste prestaties statistisch niet significant zijn ($p = 0.1765$), maar dat de metingen van onbewuste prestaties dit wel zijn ($p < 0.01$). De benodigde tijd om de controller op te rapen verhoogde met LeapTrack van 3.37 seconden naar 4.12 seconden tijdens bewuste prestaties en daalde van 6.11 seconden naar 5.83 seconden tijdens onbewuste prestaties. Uit een T-test bleek dat de verhoging tijdens bewuste prestaties statistisch significant is ($p < 0.05$), maar de verlaging tijdens onbewuste prestaties niet ($p = 0.2810$). De deelnemers ervaarden gemiddeld minder immersie met LeapTrack, met een rating van 7.17 zonder LeapTrack en een rating van 6.75 met LeapTrack. Een T-test toonde aan dat dit niet statistisch significant was ($p = 0.3383$). De calibratie scoorde positief op de SUS enquête, met een mediaan van 83.75. 7 van de 12 deelnemers waren tevreden met de calibratie na hun eerste poging, 4 waren tevreden met hun tweede poging. De tracking scoorde lager op de SUS enquête, met een mediaan van 73.75.

Conclusie

In deze thesis werd een infrarood tracking systeem voorgesteld dat gebruik maakt van de Leap Motion en bedoeld is voor VR HMD’s. LeapTrack vereist één enkele calibratie waarbij het te tracken apparaat op dezelfde locatie wordt gehouden als een virtuele representatie, waarna het in staat is om het voorwerp met zes graden vrijheid te tracken. Elk type infrarood markering kan gebruikt worden, aangenomen dat deze voldoende infraroodlicht uitzenden of reflecteren. Minstens 3 markeringen zijn vereist, geplaatst in een niet-isogonaal patroon.

Het tracking systeem werd geïmplementeerd in de Unity game engine, waarbij slechts maximum twee aanroepingen naar het tracking systeem per update nodig waren, namelijk één om een lijst van herkende markeringen op te vragen en vervolgens één om het te tracken patroon te bevestigen of om de rotatie en translatie van een nieuw patroon t.o.v. het referentiepatroon te bemachtigen. Er werd ook getoond hoe de bekomen rotatie en translatie toegepast moet worden op de virtuele representatie van het getrackte apparaat.

LeapTrack heeft echter wel zijn limitaties. Zo is de tracking enkel beschikbaar wanneer het getrackte apparaat zich binnen het gezichtsveld van de gebruiker bevindt. Ook is het nodig om markeringen toe te voegen aan het te tracken apparaat, wat moeilijk of ongewenst kan zijn. Als infrarood tracking systeem is het ook gevoelig voor de belichting van de ruimte waarin deze gebruikt wordt. De tracking kan onnauwkeurig zijn indien deze gebruikt wordt in een omgeving met veel infrarood licht of reflecterende oppervlaktes.

Tot slot werd er ook een gebruikerstest uitgevoerd waarin de deelnemers een zoekactie moesten uitvoeren, met en zonder LeapTrack. Gebruikers reageerden positief op zowel de calibratie als de tracking en drukten een voorkeur uit naar de aanwezigheid van de virtuele voorstelling van het getrackte apparaat in de VO. Deelnemers deden er langer over om het apparaat terug te vinden met tracking, maar waren in staat om het apparaat op te nemen met grotere nauwkeurigheid.

Acknowledgements

First and foremost, I would like to express my great appreciation to my promotor, Prof. dr. Johannes Schöning, for his guidance, advice and enthusiasm.

I would also like to thank Donald Degraen, for setting up the VR-capable PC used during development and testing.

Last, but not least, I would like to thank my father for helping me gain new insights, my family for their love and support and my friends for safeguarding my sanity.

Table of contents

List of figures

List of tables

Chapter 1

Introduction and Motivation

This chapter provides the reader with an introduction to the thesis titled "An optical tracking system for the integration of input devices into head-mounted virtual reality environments using commodity hardware". The goal of this thesis is to provide a tracking method for the user to track their input device while using a head-mounted virtual reality display. First a brief introduction on virtual reality (VR), its core concepts and its history will be presented followed by the benefits it provides to several application domains. The thesis problem will then be stated and the thesis structure will be outlined.

The Merriam-Webster dictionary defines VR as an artificial environment which is experienced through sensory stimuli (as sights and sounds) provided by a computer and in which one's actions partially determine what happens in the environment [1].

There are two main technical ways to implement a virtual environment (VE), namely the head-mounted display (HMD) and the Cave Automatic Virtual Environment (CAVE).
A HMD is a display device worn on the head of the user that provides a viewing space for each eye, creating a stereoscopic 3D effect. Head tracking is employed to determine what portion of the virtual space the user is currently observing. HMDs are the oldest of the two VR implementation, being conceived in 1968 [2].
The CAVE is named in reference to the allegory of the cave from Republic by Plato [3]. As the cave from The Republic projected shadows on the walls, the CAVE projects the VE on the physical space around the user. The typical CAVE is a cubic room of 3 m by 3 m by 3 m, whose walls, floor and sometimes ceiling are made of computer-projected screens [4].
CAVEs need of course not be limited to cubic spaces. Any space capable of surrounding the user would suffice, including spherical ones [5]. In order to create a 3D effect, CAVEs employ shutter glasses synchronized with the projectors or screens. Each frame is then rendered for the eye that is not covered by the shutter glasses at that time. The first CAVE was created in 1991 [6].

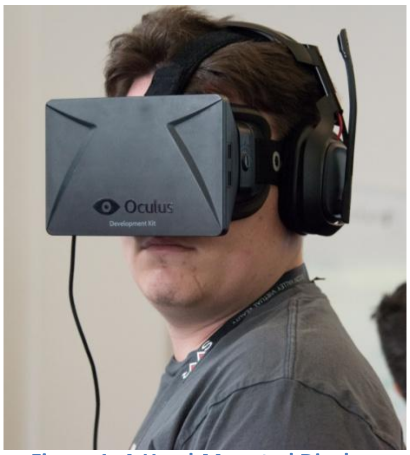
Figure 1: A Head-Mounted Display (The Oculus Rift Development Kit 1) [Image by Dcoetzee]

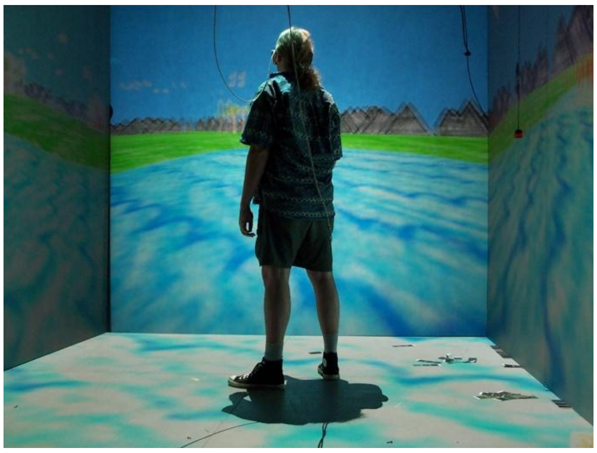
Figure 2: The CAVE at EVL, University of Illinois at Chicago [Image by Davepape]

This thesis will be focusing on VR HMDs. As such, it is useful to see how they have evolved over time, what difficulties they have faced and how these problems were overcome.

1.1 A brief history of head-mounted virtual reality

This subchapter lists the most notable VR HMDs of their respective time periods, starting with its academic inception, followed by the first foray into the arcades, the subsequent attempts at establishing a foothold in the consumer market and finally the current lineup of consumer HMDs.

The first head-mounted virtual reality display was created in 1968 at Harvard University by Ivan Sutherland. The so-called "Sword of Damocles" utilized two miniature cathode ray tubes to render wire-frame graphics in stereographic 3D with a field of view of 40 degrees. Using half-silvered mirrors, the display was partially see-through and thus able to create an augmented reality display from the changing perspective of the user. In order to determine this perspective, both an ultrasonic and a mechanical tracking method were created [2].

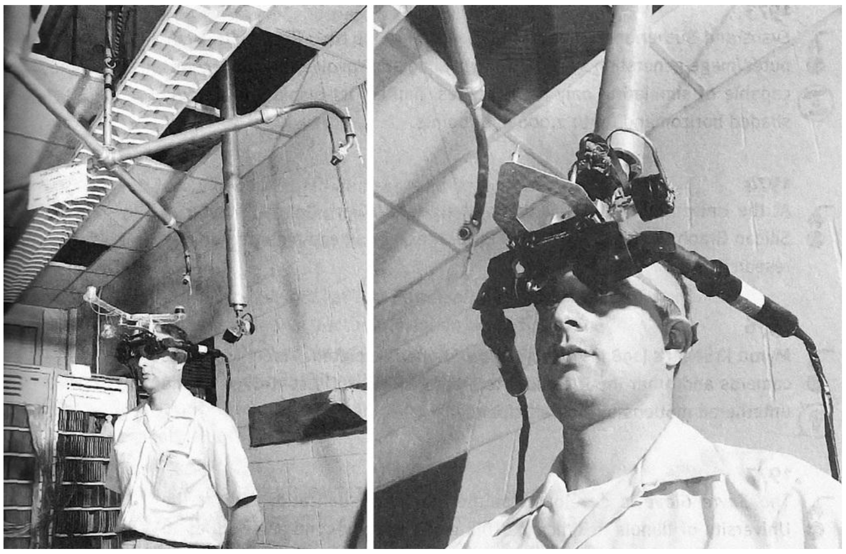

Figure 3: The Sword of Damocles with ultrasonic tracking (left) and mechanical tracking (right)
[Image by Ivan Sutherland]

VR stayed within the realms of research until 1991 where it made its arcade debut in the form of the Virtuality. These gaming machines utilized a HMD with two LCD screens, each having a resolution of 276 x 372. In addition to the HMD, the user was also equipped with a single 3D joystick and a waistband. All of these systems used electromagnetic tracking to determine their position. Up to four Virtuality systems could be networked together for multiplayer games [7]. When using the system, the player would stand in a circular elevated arena, or be seated in a cockpit depending on the model of the system. The main complaint about the Virtuality systems at the time was their lack of resolution and software support [8], with four games being available for the standing model and four for the seated model [9].

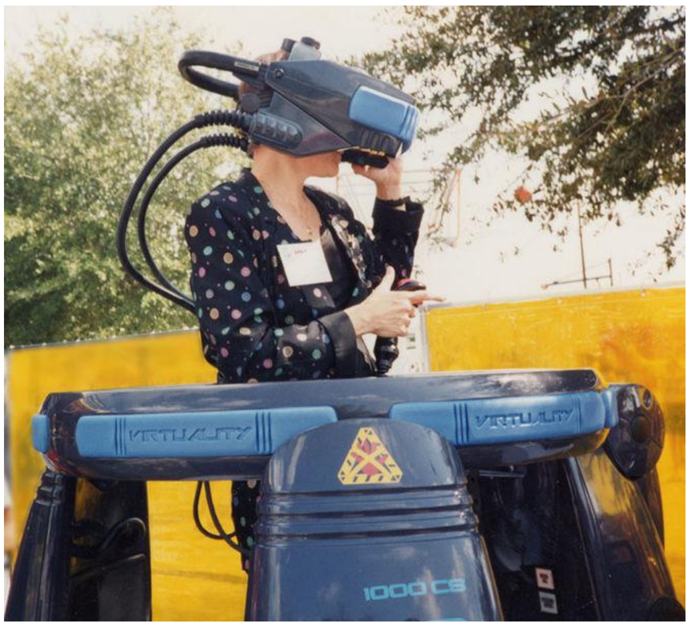

Figure 4: The standing model of the Virtuality
[Image by Skydeas]

Around 1995 the first attempts were made to enter the consumer market. The most high-profile VR device at that time was the Nintendo's Virtual Boy (VB). The VB was a red and black plastic visor with a neoprene eyepiece mounted on a metal stand. It came bundled with a black M-shaped controller with 2 D-pads and 6 buttons, 2 of which are trigger buttons on the back of the controller. Whilst the VB was designed as a portable console, it required to be placed on a table for use. The system was powered via the controller, either by 6 AA batteries or with an AC adapter. The VB created monochrome red 3D graphics with a resolution of 384 x 224 using a pair of linear LED arrays of 224 LEDs each. Each array was rapidly scanned across their designated eye by a flat oscillating mirror.
Unfortunately, the VB ended up a commercial flop. Matt Zachara and José P. Zagal propose 6 reasons for this, namely the undefined identity of the product, the comparatively poor display, it being uncomfortable to use, the difficulty to explain and demonstrate the product and the lack of a killer app [10].

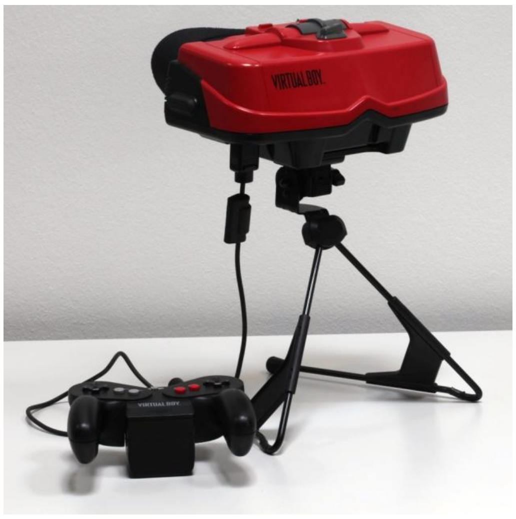

Figure 5: The Virtual Boy
[Image by Rama & Musée Bolo]

It was not until the 2012 kickstarter campaign for the Oculus Rift that public interest in virtual reality was re-established. The campaign reached its original US$250 000 goal in less than 24 hours and went on to raise US$2 437 429 from 9 522 backers over its month-long duration [11]. 2016 is shaping up to be the year of commercial virtual reality with the Oculus Rift [12] and HTC Vive [13] launching on the pc market, Sony preparing PSVR [14] for their Playstation 4 and the Samsung Gear VR [15] already available for the mobile market, not to mention the many others attempting to enter the VR market.
This notion is further supported by looking at how the Oculus Rift is avoiding the pitfalls of the VB. The Rift avoids the identity problem by marketing itself as a high-end VR headset and makes it clear a powerful PC is required for an optimal experience. The OLED display has a resolution of 2160 x 1200, resulting in a resolution of 1080x1200 per eye. This 1080p display is similar to the capabilities of the average desktop monitor, other than it having a refresh rate of 90 Hz, whereas the average desktop monitor refreshes at 60 Hz. The Rift weighs a comfortable 470g, evenly divided over the user's head. The isolating gameplay experience is

remedied by mirroring the Rift video output to a monitor and by allowing applications to include multiplayer capabilities. The difficulty in explaining the product still remains, but a possible solution might exist in green screen technology, allowing commercials to use a mixed reality approach by placing camera footage of the player inside a render of the virtual world. The Oculus Rift comes bundled with the platforming adventure game Lucky's Tale, but whether or not this will be a killer app remains to be seen.

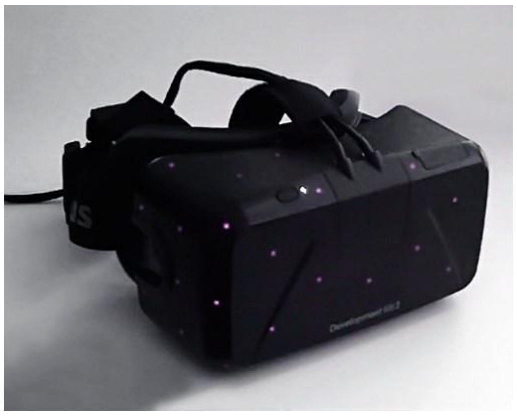
Figure 6: The Oculus Rift Developer Kit 2
[Image by ATs Kurvet]

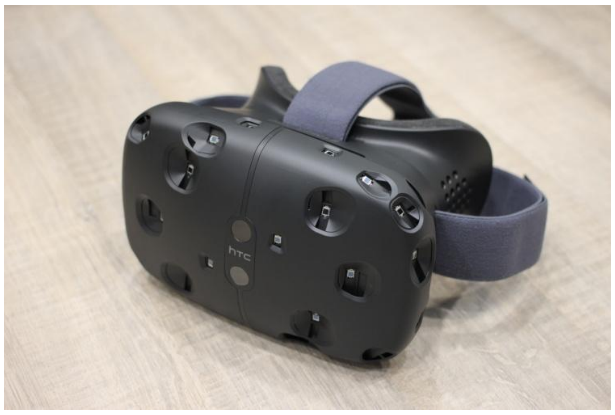
Figure 7: The HTC Vive prototype
[Image by Maurizio Pesce]

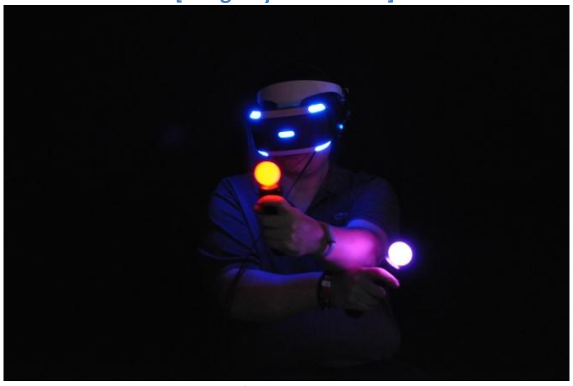
Figure 8: The PSVR prototype
[Image by Wuestenigel]

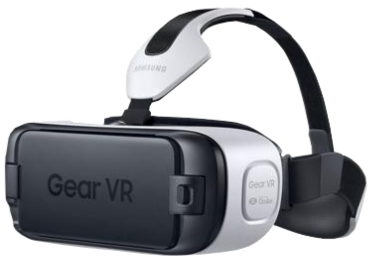
Figure 9: The Samsung GearVR
[Image by CS104group72015]

VR is finally ready to enter the home of the consumer and there could not be a more exciting time to help unleash the hidden potential that it holds and help shape the virtual future. Other application domains where VR has since established a foothold are already reaping the many benefits it brings, many of which can be applied to consumer products.

1.2 The benefits of virtual reality

This subchapter will cover the benefits that can be derived by applying VR to several application domains. The discussed domains include education and training, therapy, big data, engineering and gaming.

An analysis of VR as a teaching tool for surgical operations conducted by Diana Peterson and Cynthia Robertson [16] concluded that virtual simulations have the potential to benefit

learning, design, analysis, and communication. They have identified 4 areas where VR can have a positive influence on the learning process.
The first area is **motor proficiency**. Since organic subjects to practice motor skills on such as human cadavers and animals are costly and have limited availability, an alternative is needed. Currently this alternative has taken the form of inanimate models. A VR alternative would allow for unlimited access to motor skill training and drastically lower the cost of training materials.
The second area is **group learning**. One of the best ways of learning a subject is to teach it. The networking capabilities of VR systems can allow teacher-student interactions across the globe, connecting both experts and students with one another.
The third area is **interactive feedback**. A virtual environment allows for the collection of a wide variety of data. This allows for a more accurate and objective review of the performance of the trainee, thus providing more useful feedback. Replaying the performed actions is also possible, granting the trainee or reviewer the opportunity to review any mistakes made.
The fourth and final area is **student attention**. Students were found to be more interested in a virtual environment, either due to their link with video games, technological nature, multimodal engagement or plain novelty.
While this study was limited to surgical simulations, there is no reason to assume these benefits are limited to this field. For instance, VR is already being employed to train military [17], pilots and astronauts [18]. The 4 discussed areas are universal in nature and can be considered applicable in nearly any training scenario.

Another field VR enhances is therapy. In a review of the influence of VR in treating anxiety disorders Alessandra Gorini and Giuseppe Riva [19] conclude that virtual reality exposure therapy (VRET) is slightly more effective than in vivo exposure, which is among the most effective treatments. VRET provides many other benefits compared to in vivo exposure. Many patients are reluctant to expose themselves to their real phobic stimulus. Having them face a virtual representation of it might alleviate their concerns. This also grants the therapist exact control over their session, allowing for the intensity to be increased or decreased and for the session to be stopped at the push of a button. It also comes with a cost benefit in both time and money. For example, if the anxiety occurs during the landing of an aircraft, this moment can be replicated in VR without a need to experience the whole flight before the landing initiates. Since the VRET session takes place in a controlled environment, it allows for the therapist to monitor their patient with different instruments, letting them follow the symptom reduction with greater accuracy. The therapist can follow along with what the patient sees and identify the exact stimulus that causes the anxiety.

Of course, the benefits of VR need not be limited to the medical sector. Having a VR representation of big data can give valuable insights, since it allows the user to quite literally look at the data from a different angle. In a study on big data visualization in immersive virtual reality environments, Marko Teras and Shriram Raghunathan have found that in addition to the plain visualization of big data with the purpose of leveraging human pattern recognition, VR is already being used in more specific areas, such as the visualization and manipulation of landscapes, cities and geographical data and the visualization and analysis of brain activity [20].

VR also benefits engineering design. By being able to view a model of a future construction as if one was actually there, new insights can be acquired as a VR viewpoint does a better job of conveying size and scale. In a study concerning worker safety during construction, VR was used to place users in virtual sites that contained workplace hazards. Results found that designers could gain a fuller understanding of the hazards in VR compared to drawings, and contractors were able to better leverage their experiential knowledge [21]. VR design reviews are used not only in construction, but in other sectors such as the automotive industry as well. By letting the reviewers experience a virtual representation of the proposed implementation, they are able to see the change from the perspective of the end user [18].

The video game industry has long been a driver of VR. As the first lineup of commercial HMDs is about to enter the market this year, game developers are still experimenting with what the medium is capable of. While the full extent of the benefits VR will bring to gaming are uncertain at this point, preliminary findings are showing an increase in the level of immersion and enjoyment experienced by the user compared to non-VR games [22] [23]. This is very promising since high levels of immersion can result in a feeling of spatial presence, a state in which the user has the sensation of being physically situated within the VE and will only perceive action possibilities that are relevant to the VE, unaware of the actions that are linked to the real environment [24].
Theme parks are interested in the gaming potential of VR as well [25], and have reported similar findings [26]. With the advent of wireless VR devices such as the Gear VR it becomes possible to experience VR while mobile, which SixFlags is attempting to leverage with a VR rollercoaster ride [27]. VR in theme parks has the added benefit of being able to keep the ride fresh by changing the displayed VR scene while keeping the same infrastructure.

While VR as is already has much to contribute to numerous application domains, there are many aspects of it that are not quite on point yet. While some are simply a matter of current technical limitations, such as the screen door effect caused by a sub-optimal screen resolution [28], others are caused by design choices of the technical implementation of the VR viewing device. The goal of this thesis is to address one of these problems specific to HMDs and offer a potential solution.

1.3 The problem of obscured vision

In this subchapter the thesis problem is stated and a solution is proposed.

Due to the nature of a HMD, the user is blind to the outside world once it is equipped. This means that once the user enters VR, they will no longer be able to find their input device. In order to pre-empt this problem, the Rift, Vive and PSVR each have their own wand-style controller for player interaction, piggybacking on the tracking technology used for their headset itself. These controllers can be visualized inside the application if the developer so desires. This feature allows the user to locate these devices as long as they are within the tracking range of the setup.

However, the problem still persists for any peripheral devices developed without VR in mind, such as controllers, mouse and keyboard, steering wheels, etc. As such, it can be a frustrating experience finding buttons or even the peripheral itself without being able to rely on eyesight, and restoring said eyesight would require removing the headset, shattering any established immersion. The HTC Vive is currently the only HMD that attempts to alleviate this handicap by implementing a front facing camera. Whilst effective in the sense that the user no longer has to remove their headset, it still suffers from the main drawback of diminishing the immersion of the user.

A possible solution can be found in the augmented virtuality substratum of the reality-virtuality continuum, which encompasses all possible variations and compositions of real and virtual objects [29]. By augmenting the VE with location data acquired by tracking the peripheral, it would be possible to integrate these devices into VR in a similar fashion to that of the proprietary wand-style controllers. This way developers can create a virtual representation of the peripheral matching the style of the application. This would allow the user to locate their device with minimal impact on immersion.

Tracking these peripherals does pose an additional challenge, in the sense that the shape of the peripheral is unknown. As such, a versatile tracking method is needed that can function without relying on this information being available. As it is unknown at this point when the next generation of consumer HMDs will arrive and whether or not they will include these kinds of tracking capabilities, LeapTrack will be limited to commodity hardware. This is to ensure a consumer solution is available without having to rely on these unknown factors.

LeapTrack, the tracking solution proposed in this thesis uses an infrared inside-looking-out approach. First, a Leap Motion [30] is attached to the front of the HMD. This device uses two calibrated monochromatic infrared cameras and is intended as a low-cost hand tracking apparatus. It was initially designed as an outside-looking-in device, but support was added for inside-looking-out hand tracking for VR HMDs via a special mount. By using an inside-looking-out approach with the camera orientation linked to the position of the head of the user, tracking is still possible in unexpected positions such as when the peripheral is dropped on the floor.

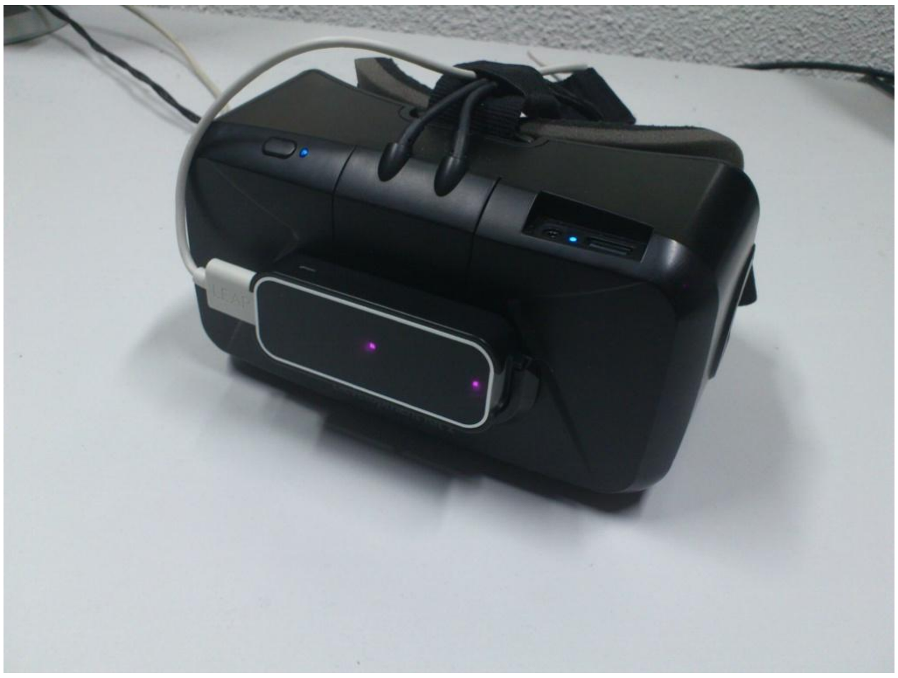

Figure 10: A Leap Motion mounted to an Oculus Rift DK2

As the Leap Motion is able to produce its own infrared lighting, the user can utilize any kind of infrared marker to identify the peripheral for tracking. While attaching infrared LEDs or spherical retroreflective markers would provide the optimal results, any infrared reflecting material can be used in a pinch. A minimum of three markers are required for tracking with 6 degrees of freedom (DOF), with additional markers generally providing more consistent tracking results.

After a single calibration step, information about the rotation of the tracked object and its position in reference to the mounted Leap Motion can be acquired with reasonable accuracy. The nature of the object to be tracked and how it is represented in the VE is left up to the developer of the VR application.

1.4 Thesis structure

This subchapter outlines the structure of this thesis.

In chapter two the proposed solution is put in the context of related work. It has been informed by research from three areas: virtual reality interactions, body input and eyes-free input.

Chapter three presents the tracking system, which enables locating the input device with reasonable accuracy, in detail.

This is followed by chapter four, in which the implementation of the tracking system is covered along with its integration in the Unity game engine.

In chapter five the impact of the LeapTrack tracking solution is demonstrated on the user experience by means of a user study where the speed of locating the peripheral will be evaluated alongside the usability of the tracking solution and the impact on immersion.

Finally, conclusions are drawn in chapter 6.

Chapter 2

Related work

This chapter examines previous work that is related to the problem caused by the lack of vision while using a head-mounted virtual reality display. It is informed by research from four categories. In the first subchapter titled "Virtual Reality Interactions", focus is placed on works that employ the same strategy as this thesis, namely tracking the input device. The second subchapter, "Body Input", features works that tackle the problem from a different direction, eschewing peripherals altogether and focusing on input provided by the body of the user instead. In the third and final subchapter, "Eyes-free input", works that take the middle ground by utilizing input devices that require no tracking are examined.

2.1 Virtual Reality Interactions

In this subchapter related works are examined that track an input device for use in a VE. For each work, the tracking solution employed is stated along with the implementation of the virtual representation of the input device. At the end of the subchapter, the advantages and disadvantages of the discussed tracking methods are compared to those of LeapTrack.

The Virtual Tricorder [31] proposes a versatile way of interacting in a virtual environment. The intent is to create a device capable of interacting with VR in multiple ways in the same manner as the Tricorder from the Star Trek series achieves multiple functionalities in a single device. To achieve this, a setup was created using a FakeSpace BOOM (Binocular Omni-Orientation Monitor), which is a VR binocular mounted on a multi-link arm, and a Logitech Flymouse, an ultrasound-tracked 3D mouse. The virtual Tricorder was mapped on to the physical location of the Flymouse.

The Tricorder was able to represent several different tools for use in VR with the advantage of physical feedback from the Flymouse. The user would be able to select which tool to use while in VR without the need to pick up another physical input device. The paper offers four examples of such virtual tools, namely a 2D menu anchored to the Tricorder, a virtual grappling iron that lets the user move around the virtual scene by dragging it around themselves, a semi-transparent cone projected from the front of the Tricorder which is used to select and then position objects and a 3D magic lens that displays an alternate version of the VE viewed through it.

3D Palette: A Virtual Reality Content Creation Tool [32] maps a physical pen and tablet onto a virtual representation to create an enhanced input device with the intent of facilitating 3D scene creation. Both the Wacom pen and the tablet were tracked by Polhemus Fastraks, wired electromagnetic sensors providing 6 degrees of freedom (DOF) tracking relative to a transmitter with a refresh rate of up to 120 Hz, depending on the number of sensors used. The electromagnetic field of the Wacom tablet does interfere with the Fastrak tracking, but this is compensated by using the input of the pen on the tablet. CrystalEyes stereoscopic shutter glasses were used to display the virtual reality and a head-mounted microphone was used for voice commands.

The virtual tablet displays a range of options the user can select using the pen or using vocal shortcuts. These options include loading and editing 3D models, drawing textures and selecting color, as well as non-modeling related options such as enabling or disabling the microphone and allows for movement by drawing movement vectors onto the tablet. The virtual nature of the tablet is exploited to full extent by displaying options as 3D graphics popping out of the tablet. The tracking of the pen is not only for visualization purposes, as it can be used in the scene to select and reposition objects.

A new Optical Tracking System for Virtual and Augmented Reality Applications [33] also uses pen and tablet as an input method to demo the tracking system, but here the input solely originates from the alternative tracking method used. The hardware setup consists of retroreflective spheres attached to a HMD, pen and tablet, which are illuminated by infrared LED arrays and filmed by a calibrated stereo pair of cameras with an infrared pass filter operating at a 30 Hz frame rate. This rig of cameras and LED arrays is mounted to a fixed frame with an adjustable baseline of up to 2m. The setup uses an outside-looking-in approach, where the optical sensor is placed in a fixed location and landmarks are placed on the objects to be tracked.

On the software side, three steps are taken in order to convert the images captured by the pair of cameras into positional information for the tracked devices. The first step is blob detection, where the bright spots caused by the reflected infrared light from the markers are identified and false positives are filtered out. The second step is matching the blobs from one camera with the corresponding blobs from the other camera using epipolar constraints. The third and final step is 3D reconstruction, which uses the matched blob locations to determine the 3D position of the marker. These 3D points are then gathered into constellations, which in turn can then be identified as their corresponding tracked device.

The Wii Remote as an input device for 3D interaction in immersive head-mounted display virtual reality [34] researches the possibility of leveraging commodity hardware, specifically the Wii Remote, to create a cheap motion tracked input device for VR. The Wii Remote is a wand-style controller intended for use with the Nintendo Wii console. The "Wiimote", as it is commonly known, utilizes Bluetooth for wireless connectivity and is equipped with 3 linear accelerometers that are used in conjunction with an infrared sensor for motion detection. This infrared sensor is intended to be used with a sensor bar, which has 2 clusters of 5 infrared LEDs at each end of the bar.

In order to avoid having to surround the user with infrared light sources to accomplish 360 degree tracking, two alternative setups are suggested.
The first setup uses an outside-looking-in approach, where an optical sensor in the form of a second Wiimote is mounted above the head of the user and infrared LEDs are attached to the primary Wiimote to serve as landmarks. The pitch and the roll of the primary remote can be determined using the accelerometer readings of said remote, while the yaw and position is calculated using the infrared sensor of the second controller.
The second setup opts for an inside-looking-out approach, with the sensor bar being attached to the ceiling instead of in front of the user. While this new position has the advantage of being better suited for 360 degree use, this unfortunately means that the Wiimote has to point with the sensor towards the ceiling at all times, making it unintuitive to use. This setup does lend itself better to multi-user configurations, as the infrared lights can be used by any Wiimote in the setup space.

It is worth noting that while the tracking was found to be accurate enough, it was far from perfect due to the jitter on the readings from the Wiimote. In an experimental setup using the outside-looking-in approach, it was found that higher pitch values resulted in decreased accuracy. The use of a smoothing factor was found to increase the accuracy in these scenarios, though it contributed little to low pitch scenarios. This does highlight that additional processing of tracking input can improve the usability of the tracking data when using non-perfect tracking methods.

An often forgotten aspect of VR interaction is haptic feedback. While 6 DOF input devices are commonly used in VR, there is rarely any feedback when their virtual representations interact with objects in the virtual scene. **Ring-shaped Haptic Device with Vibrotactile Feedback Patterns to Support Natural Spatial Interaction** [35] attempts to remedy this by introducing a wireless infrared tracked ring capable of providing haptic feedback depending on its location in the virtual scene.

The setup uses an outside-looking-in approach utilizing a WorldViz Precision Position Tracking system to track a single infrared LED attached to the ring with 3 DOF. An actuator placed on the inside of the ring is used for vibrotactile feedback and a joystick is attached outside of the ring for additional input. The ring is intended to be worn on the first segment of the index finger in order to facilitate pointing gestures, touch feedback and comfortable use of the joystick. An Oculus Rift DK2 HMD is used to visualize the virtual scene.

Several vibrotactile feedback patterns were considered in an experiment involving the user touching and penetrating a virtual sphere. This experiment led to the conclusion that vibrotactile peaks at the outline of the virtual object are preferred for touching and penetrating gestures, a lack of feedback is less noticeable than a vibrotactile peak and that increasing the feedback as the user nears the object can aid in the finding of said objects.

A Dose of Reality: Overcoming Usability Challenges in VR Head-Mounted Displays [36] tackles the same problem as this thesis does, but attempts to solve it with a solution leaning closer to reality on the reality-virtuality continuum. Three studies are performed. In the first two studies, a setup is used with an RGB webcam attached to the DK1 HMD. The objects that are to be integrated into VR are placed in a green screen environment and the user is wearing a marker on each of their hands for hand tracking purposes. The third study takes place in a normal environment and uses a Microsoft Kinect instead of a regular webcam.

The first study integrates a keyboard into the VE when the user reaches out to interact with it. Having the keyboard visible in VR was found to greatly reduce error rates. No difference in performance was found between providing a full view of reality compared to only visualizing the keyboard and the hands of the user. The second study is an extension of the first one, but incorporates additional objects. The results of this study indicate that users prefer to view a selective subset of reality as opposed to the full view. There was no significant difference between partial reality, which displayed all interactable objects, and minimal reality, which only displayed the area around the hands of the user. The third study displayed the faded silhouettes of any person detected by the Kinect, turning them opaque when the user wished to interact with them. While it did improve awareness of proximate persons, users indicated they would prefer textual notifications or abstract representations as the silhouettes were too distracting, even in faded form.

While most tracking solutions strive for a one-on-one mapping of the tracked object and the virtual representation, this may sometimes be undesirable, for instance when the physical location of the peripheral would be behind a virtual wall. In **Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences** [37] a framework was created that warps the virtual space to match the location of a physical object. The setup utilized a Microsoft Kinect suspended overhead which tracked the hand of the user and a cube, both marked with retroreflective markers. A wand-style controller marked with retroreflective markers was used for a comparative study. Visualization was done with an Oculus DK2 HMD.

To achieve the desired effect, two mapping techniques were explored, then combined as a hybrid technique. The first technique, body warping, manipulates the virtual representation of the body of the user in such a way that at the point of contact with the real object, the body part also appears to be at the virtual object. The second technique, world warping, manipulates the coordinate system of the VE so that the virtual object appears in the same location as the physical object. The hybrid technique employs world warping during the head movements of the user, then uses body warping to fill in the remaining warp. This hybrid technique was found to be less noticeable by the user, creating the highest sense of presence compared to both the individual techniques and the wand controller.

Rather than having to guide the user to the peripheral, it is also possible to position the peripheral where the user expects it to be. **Snake Charmer: Physically Enabling Virtual**

Objects [38] utilizes a Robai Cyton Gamma 300 robotic arm to move a physical object to where the user expects it to be in the VE. An outside-looking-in approach is used with a Microsoft Kinect tracking the hand of the user within the operating range of the arm. Visualization is done with an Oculus DK2.

The arm is capable of switching between different endpoints. Three kinds of endpoints are considered, namely passive endpoints which simulate textures or local features, active endpoints which simulate physical stimuli such as temperature or airflow and input endpoints, which provide input methods in the form of buttons, dials, touch screen and pressure sensors.

In conclusion, the discussed related works can be categorized in three tracking methods. The first category uses electromagnetic tracking. While popular in papers written in the nineties, they have fallen out of favor in more recent works. While accurate and responsive, any object capable of generating an electric field can interfere with the tracking. A visual tracking method as employed in our implementation does not suffer this disadvantage and can be created with commodity hardware.
The second category uses infrared tracking, as does LeapTrack. The methods used in the discussed papers are improved upon in two major ways. Firstly, LeapTrack opts for an inside-looking-out perspective. While an outside-looking-in perspective would be better at guiding the user to their peripheral, it requires the camera to be set up in a way that the peripheral is visible in any circumstance. The inside-looking-out perspective uses the vision range of the user, allowing them to dictate the direction in which the peripheral is most likely to be. This allows the peripheral to be detected even in unexpected locations. The second improvement is the versatility of the detection algorithm used in LeapTrack. The user is able to teach the system which object to track, whereas other methods track a pre-defined configuration.
The third category uses the Microsoft Kinect. This device includes both an RGB camera with a resolution up to 1920 x 1080 pixels and a field of view of 84.1 x 53.8 degrees, and an infrared depth camera with a resolution up to 512 x 424 pixels and a field of view of 70.6 x 60 degrees, both capturing at 30 Hz. Included in the Kinect SDK are skeletal tracking algorithms. LeapTrack opts for the Leap Motion. This device uses two monochromatic infrared cameras with a resolution of 640 x 240 pixels and a field of view of 132 x 115 degrees capturing at up to 150 Hz. The Leap Motion SDK includes hand tracking algorithms. The frame rate of the Leap Motion is better suited for VR as HMDs aim to render at a frame rate of at least 90 Hz. Data acquired from the Kinect would have to be predicted for two out of every three frames for a smooth experience.

A different perspective on providing immersive input during VR sessions is to eschew peripherals altogether, instead opting to utilize the body of the user as a means of input. With no peripherals to lose track of, the problem described in this thesis would be rendered moot.

2.2 Body Input

In this subchapter related works are examined that discuss input methods that utilize the body of the user. For each work, the input method and the required tracking setup are described. At the end of the subchapter, the advantages and disadvantages of the discussed input methods are compared to those of LeapTrack.

Stefan Greuter and David J. Roberts look into full body movement in VR in **SpaceWalk: Movement and Interaction in Virtual Space with Commodity Hardware** [39]. The SpaceWalk platform consists of two hardware components, a tracking station and a VR backpack. The tracking station is a stationary setup using a computer connected to a wireless network and a Kinect camera to track the user. This computer processes the skeletal and location tracking information supplied by the Kinect and conveys this information in the form of events to the VR backpack. The VR backpack consists of an Oculus Rift Developer Kit version 1 (DK1) VR HMD connected to a tablet computer, along with an external phone charger battery to power the HMD and a mini display port to HDMI adapter to connect the video output of the tablet with the DK1. The tablet uses the data from the tracking station and the orientation of the HMD to update and render the virtual scene, even providing a warning if the user is about to exit the area tracked by the Kinect. It is possible to run a copy of the tablet software on the tracking station computer in order to visualize what the user is experiencing.

In **TranSection: Hand-Based Interaction for Playing a Game within a Virtual Reality Game** [40], hand tracking is used as one of the main methods of interacting with the designed game. This hand tracking is provided by the Leap Motion. In the VR puzzle platformer game, the user is seated behind a desk with on it a monitor and several other virtual objects. On the virtual monitor, a 2D game is visible where the user can control their character with their real keyboard. While exploring the level, several puzzles will be encountered that need to be solved by picking up and using the virtual objects on the desk. For example, a cup of water can be poured over a fire blocking the way on the monitor.

Since the game requires the user to switch between using keyboard and hands as an input mechanism, the user might lose track of the position of their hand on the keyboard. To help alleviate this problem, the virtual keyboard is mapped to match the location of the physical keyboard. In addition, any ineffective key pressed on the physical keyboard will be highlighted on the virtual keyboard, giving the user an idea how far their hand is placed off the mark. Since the Leap Motion is mounted on the HMD, hand tracking is not available when the hands of the user are not within their line of sight. When this is the case, the game assumes the user has placed their hands on the keyboard, and will render the avatar of the hands in a resting position on top of the virtual keyboard.

This game does highlight one of the design limitations with seated experiences. Since the user cannot move, any item that the user can interact with must be placed within arm's reach of the user. **The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR** [41] attempts to solve this by growing the arm of the user as the user attempts to interact with objects outside of their reach. Two Polhemus Fastraks are utilized to track the hand of the user. The Go-Go technique maps the position of the virtual hand on that of the physical hand when it is near the user. Once a threshold of two thirds of the arm length of the user is crossed by the hand, the mapping switches to a non-linear one, extending the reach of the virtual hand further than that of the physical hand. To visualize this, the application displays the position of the real hand as a small cube in addition to the extended virtual hand.

The use of VR HMDs truly means being surrounded by the virtual world and there is no non-interactive medium that conveys this as well as omni-directional video (ODV). **Towards Effective Interaction with Omnidirectional Videos Using Immersive Virtual Reality Headsets** [42] sets out to capitalize on this newfound freedom by proposing a new interaction scheme. Whereas ODV require mouse or keyboard interaction to view a different section when viewing these on traditional setups, HMDs allow the viewpoint to be mapped to the orientation of the head of the user. This is an intuitive way of dealing with the spatial component of ODV, but makes it more challenging to deal with the time component. Since the outside vision of the user is obstructed by the HMD, the user might find it difficult to interact with an input device for video features such as play/pause and forward/rewind. A solution is found in the form of gesture controls. By mounting a Leap Motion on the HMD, hand gestures formed in front of the HMD can be linked to the aforementioned video features. For instance, making a push gesture signals play in this implementation.

An alternative to gestures is gaze input. **Orbits: Gaze Interaction for Smart Watches using Smooth Pursuit Eye Movements** [43] proposes a gaze interaction technique intended for smart watch usage. By using smooth pursuits where the eyes follow a moving object, the jitter that occurs when the eyes focus on a static target can be avoided. Orbits leverages this by having a dot orbit around each of the options. This orbit can vary in phase, angular speed and direction depending on the option, allowing it to be easily mapped to the measured eye movement.

While eye tracking is not available in the current line-up of commercials HMDs, it is probable that future versions will incorporate it due to the advantages it brings, such as foveated rendering [44]. The head orientation of the user can be utilized instead for current day applications. The smart watch can be simulated in VR, or discarded altogether in favor of an alternative virtual display.

The human gaze can also be leveraged for non-input purposes. **Augmenting the Field-of-View of Head-Mounted Displays** [45] surrounds the lenses of a HMD with an array of LEDs to expand the FOV and engage the peripheral vision of the user. Three visualization modes were developed, namely “full environment”, “objects of interest only” and “countervection visualization”. “Full environment” attempts to extend the FOV as faithfully as possible, displaying the scene as-is. “Objects of interest only” limits itself to only displaying select objects. “Countervection visualization” displays a series of vertical bars that move in the same direction the avatar of the user is moving to reduce the feeling of self-motion, and by extension cyber sickness.

In the search task study performed in this paper, an increase in the finding speed was noted with the objects of interest visualization compared to the default view of the HMD. This visualization mode is of particular interest, as a peripheral tracked by an outside-looking-in method can be visualized this way.

All the previously discussed eyes-free methods are reliant on touch. **Application of Speech Recognition Technology to Virtual Reality System** [46] highlights an alternative input method in the form of speech recognition. An implementation using the Microsoft Speech SDK is described in which the user is able to navigate a maze in VR using only voice commands. Supported commands included going forward, stopping, turning 10 degrees left or right and turning 180 degrees. The system was concluded to have high accuracy and good real-time command processing.

In conclusion, the biggest advantage of body input over traditional input peripherals is that the user is always aware of the position of their body.
Full body tracking gives an advantage over simply tracking the HMD and wand controllers in the sense that information about the posture of the user, such as the difference between crouching and bending over, is lost when only tracking the peripherals. Without full body tracking, interactions such as kicking a ball are no longer an option either as none of the wand controllers are intended to be equipped to the feet of the user. While foot tracking is theoretically possible with LeapTrack by marking the shoe of the user as the input device, it is not recommended as the inside-looking-out method employed would require the user to constantly look at their feet, which would not make for a compelling experience.
Hand tracking allows for an intuitive way to interact with VR, and gestures can be employed for more complex interactions. Distance is a limiting factor, but this can be worked around, assuming no interactions with physical objects are required at an extended distance. As LeapTrack uses the Leap Motion, hand tracking can be used in conjunction with the peripheral tracking capabilities.
Gaze input has the potential to become an integral part to future HMD usage, but at this point in time the hardware support is missing. The Leap Motion used in LeapTrack has a horizontal field of view of 132 degrees, whereas that of the Rift or Vive is 110 degrees. Objects detected by LeapTrack can thus be telegraphed before they fully enter the field of view of the user.
Speech recognition is useful for simple commands, and only where the command is not time critical. For instance, if the user tells the system to turn left by 90 degrees, chances are the avatar will have hit a wall by the time the user finishes speaking. Complex commands are often not supported, as they either could be phrased or interpreted in multiple ways, thus

requiring the user to memorize the accepted commands.
Unfortunately, it is often challenging to make interfaces intuitive to use with body input. There are cases where traditional input is preferred and unfortunately bodily awareness is not applicable to input devices once separated from touch. This further illustrates the point that a reliable way to identify the position of said input devices is often not a luxury, but a necessity.

Some form of physical feedback is often desirable which pure body input is incapable of delivering. Wearable input devices capable of eyes-free input might offer a suitable compromise, as they provide the physical feedback that classic input devices deliver, while being significantly harder to lose track of, similar to body input.

2.3 Eyes-free input

In this subchapter related works are examined that discuss wearable input devices capable of eyes-free input. For each work, the input device and their workings are described. At the end of the subchapter, the advantages and disadvantages of the discussed input methods are compared to those of LeapTrack.

While it is generally impossible to blindly determine the position of an input device, users are aware of any devices touching their body. **Nenya: Subtle and Eyes-Free Mobile Input with a Magnetically-Tracked Finger Ring** [47] leverages this property by providing a magnetically tracked ring for the user to utilize as an input device. The position and rotation of this magnetic band-style ring is measured by a 3-axis magnetometer worn as a bracelet on the same hand. The magnetometer used here allows for sampling at 25 Hz. These measurements are then communicated via Bluetooth, making the entire setup wireless. By adding a small disc magnet to the ring as a landmark, it can be used entirely by touch. Options can be selected by rotating the ring to the position corresponding to the option, then confirming the selection by shifting the ring away from the bracelet.

In **Belt: An Unobtrusive Touch Input Device for Head-worn Displays** [48] another worn user interface is considered. A common leather belt is extended with up to 288 small metal rivets, each wired to one of six battery-powered touch sensing units woven onto the inside of the belt. The touched rivets are communicated via Bluetooth and can be used to detect swiping and tapping gestures on the belt.

Wearable input devices need not be limited to accessories. **iSkin: Flexible, Stretchable and Visually Customizable On-Body Touch Sensors for Mobile Computing** [49] introduces skin-worn sensors for touch input on the body. These sensors are able to distinguish between touching and pressing actions and are flexible enough to be worn on various body locations. They can be created in all manner of shapes and sizes, allowing for the customization of touch sensors for specific applications, visual appearances and button layouts.

In conclusion, while these input devices certainly succeed in their main objective of being near impossible to lose and providing the physical feedback body input is incapable of delivering, the one-dimensional nature of the input means they lack as an input device for all but the most basic of VR applications. They do offer an advantage when combined with hand tracking, as there is no need to put down a peripheral to switch to gesture controls as is the case when using the hand tracking provided by the Leap Motion used in LeapTrack.

2.4 Overview of related work

Work	Tracking method	Required hardware	Tracking target	Commodity hardware only	Generic	Optimal refresh rate (>=90 Hz)	Tracking limitations
The Virtual Tricorder	- Ultrasonic	- Logitech Flymouse	- Logitech Flymouse				- Wired - Ultrasonic interference
3D Palette: A Virtual Reality Content Creation Tool	- Electro-magnetic	- Polhemus Fastrak	- Polhemus sensors		✓	✓	- Wired - Electromagnetic interference
A new Optical Tracking System for Virtual and Augmented Reality Applications	- Infrared	- Infrared cameras - Infrared LED array	- Infrared retroreflective markers				- Area constricted to overlapping field of view of static cameras - Infrared interference
The Wii Remote as an input device for 3D interaction in immersive head-mounted display virtual reality	- Infrared	- 2 Wii Remotes - 2 infrared LEDs	- Wii Remote	✓		✓	- Area constricted to field of view of ceiling mounted Wii Remote - Infrared interference

Ring-shaped Haptic Device with Vibrotactile Feedback Patterns to Support Natural Spatial Interaction	- Infrared	- 2 infrared cameras - Ring with infra-red LED	- Infra-red LED				- No 6 DOF - Area constrained to overlapping field of view of static cameras - Infrared interference
A Dose of Reality: Overcoming Usability Challenges in VR Head-Mounted Displays	- Chromakey - Infrared	- Microsoft Kinect	- Objects not in chromakey - Skeleton	✓	✓		- Area constrained to field of view of static camera - Green screen environment needed - No actual tracking (visualization only) - Infrared interference
Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences	- Infrared	- Microsoft Kinect	- Marked cube - Skeleton	✓			- No 6 DOF on cube - Area constrained to field of view of static camera - Infrared interference
Snake Charmer: Physically Enabling Virtual Objects	- Infrared	- Microsoft Kinect	- Skeleton		✓		- Object must be attachable to robot arm - Area constrained to robot arm range - Infrared interference
LeapTrack	- Infrared	- Leap Motion	- Infrared markers - Hands	✓	✓	✓	- Area constrained to field of view of user - Infrared interference

Table 1: Overview of the most important peripheral tracking methods mentioned as related work

Chapter 3

Concept

This chapter provides the reader with a high-level overview of the workings of LeapTrack. Firstly, the required hardware will be discussed. Subsequently, an overview of the core loop of the tracking solution will be covered, after which the three main steps of the tracking process will be discussed. The first step will cover detecting the infrared markers in the individual camera feeds. The second step describes the process of matching the found markers and using this information to find its location in 3D space. The third and final step covers identifying a stored marker pattern and estimating the rotation and translation compared to the original pattern.

3.1 Tracking setup

This subchapter covers both the required and optional hardware required for LeapTrack, as well as the configuration in which these need to be set up.

The hardware setup consists of a Leap Motion attached to the front of the HMD using the Universal VR Developer Mount. The Leap Motion uses two factory calibrated monochromatic infrared cameras with a resolution of 640 x 240 pixels and a field of view of 132 x 115 degrees capturing at up to 150 Hz. It supplies its own lighting through 3 infrared LEDs emitting at a wavelength of around 850 nm on the electromagnetic spectrum. The Leap Motion comes bundled with hand tracking software optimized for VR usage. This setup can be applied as is to any HMD compatible with the Developer Mount.

In order to identify the peripheral to be tracked, a non-isogonal pattern of at least three infrared markers will need to be applied to it. As the Leap Motion supplies its own infrared light any kind of infrared marker can be utilized. While attaching infrared LEDs or spherical retroreflective markers would provide the optimal results, any material capable of sufficiently reflecting infrared light can be used.

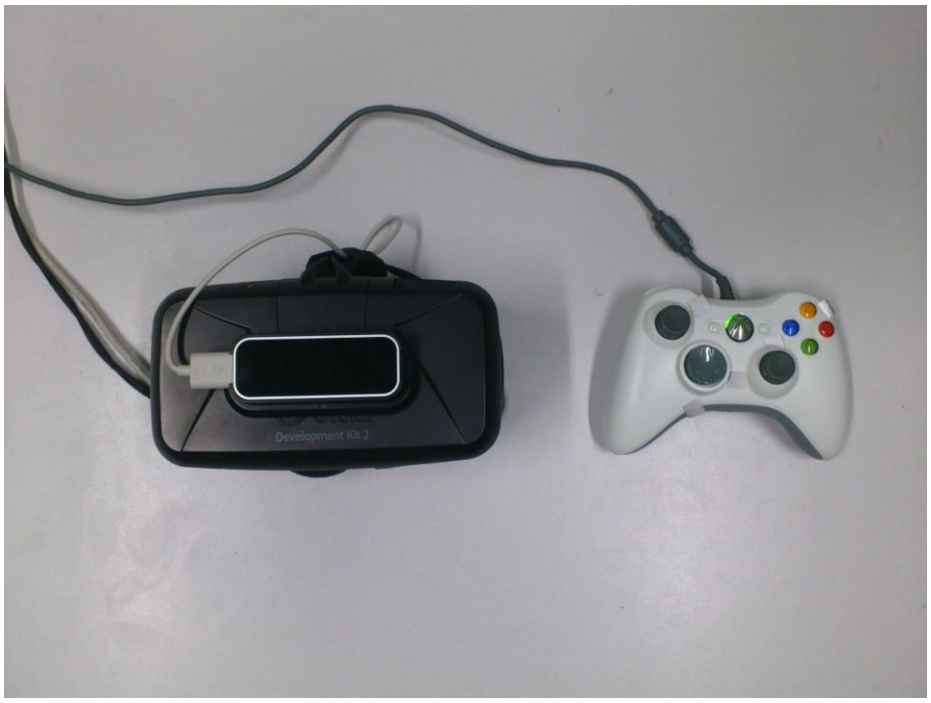

Figure 11: A possible hardware setup consisting of a Leap Motion mounted to an Oculus Rift Development Kit 2 and an Xbox 360 Controller marked with retroreflective stickers

3.2 Overview

A brief overview of the core loop of the tracking solution will be provided in this subchapter.

The core loop is named as such not because it is a loop of instructions, but because it is a series of instructions intended to be executed during the main update loop of the application. As LeapTrack is intended to make optimal use of the high capture rate of the Leap Motion, this loop should be executed at least once per rendered frame.

The first step of the core loop is acquiring the latest image data captured by the Leap Motion. This image data is available as a pair of images, one for each camera, with a single channel containing the sensor brightness values. Additionally, the camera calibration maps are available as well, which will be used to correct the lens distortion in the images.

The markers placed on the peripheral are then identified by their high brightness values. A contour tracing algorithm is used to locate the centroids of the markers. The points identified in the left image are then paired with their corresponding point in the right image. The coordinates of each pair of points can then be used to triangulate the location of their marker in 3D space.

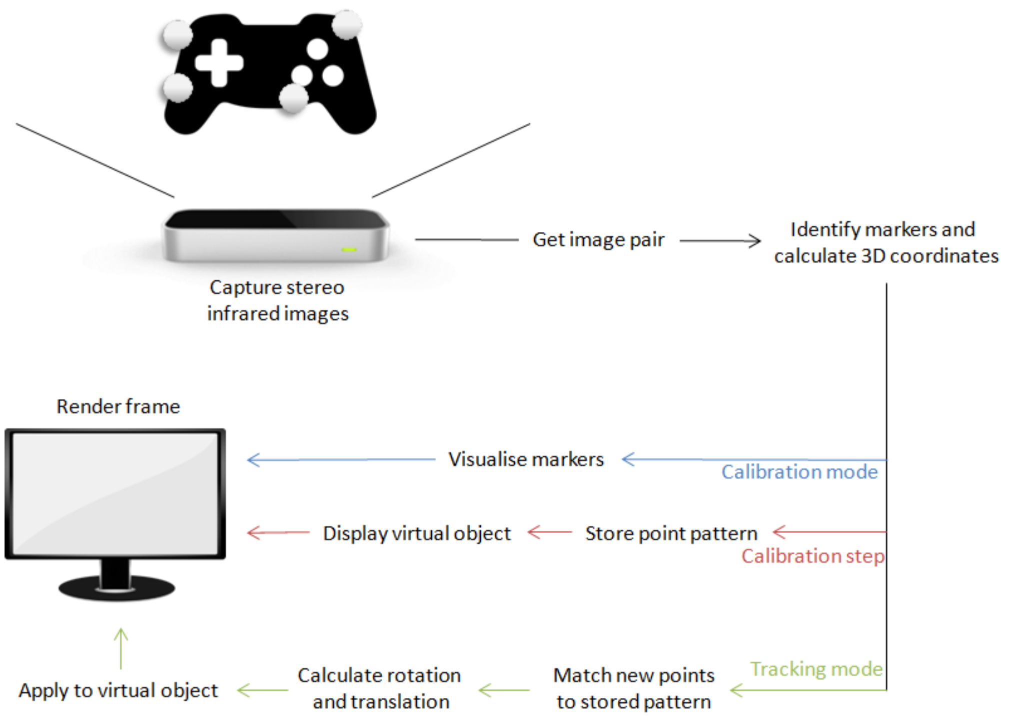

Figure 12: An overview of the core loop of the solution

The manner in which this list of 3D points is further used depends on the current mode of operation of the application. Due to the universal nature of the tracking application, a pattern to track must first be established. During this calibration process, the user should align the peripheral to be tracked with its virtual representation. In order to assist in this matter, the list of 3D points can be visualized in order to inform the user of the position of the physical peripheral in relation to the virtual one.

Once the peripheral is in the correct position, the calibration step can occur. During this step, the pattern in the list of 3D points perceived by the Leap Motion is stored as the default position for the virtual peripheral. Whenever a rotated or translated version of this pattern is observed, the difference with the pattern stored in this step will be calculated and applied to the virtual peripheral. Any visual aid displayed during the calibration mode can be disabled after the completion of the calibration step.

With the calibration completed, LeapTrack can enter the tracking mode, in which it will be spending the rest of its runtime. First, a voting algorithm is used to identify as many points of the original pattern in the new list of 3D points as possible. The difference in translation between the centroid of the identified points and the centroid of the matching subset of the original pattern is used as the translation of the virtual peripheral. To calculate the rotation between the two patterns, the Kabsch algorithm is employed.

With both the rotation and translation in relation to the calibrated position of the physical peripheral known, it is possible to transform the virtual peripheral to the matching position.

3.3 Marker detection

This subchapter provides an in-depth explanation of the detection of the infrared markers and the conversion to pixel coordinates.

The Leap Motion supplies us with an image containing the sensor brightness values and a buffer containing the camera calibration map, which can be used to correct lens distortion and other optical imperfections in the image data, for each of the two cameras. During the marker detection step only the sensor brightness values are considered in order to determine the location of the infrared markers placed on the peripheral.

LeapTrack uses a contour tracing approach using the Moore-Neighborhood tracing algorithm. The image is scanned until a pixel is found with a brightness value over a set threshold. This pixel is added to the boundary and defined as the current boundary pixel. The previous pixel is defined as the entry point. The Moore neighborhood of the current boundary pixel is then evaluated, which is the set of 8 pixels that share an edge with the current pixel, starting clockwise from the entry point. As soon as another pixel above the threshold is found, this pixel is added to the boundary and set as the new current boundary pixel. The previous boundary pixel is designated as the entry point and another sweep is performed. This is repeated until the original pixel is revisited.

As the following steps of the tracking algorithm require a single point rather than a shape, the center of the shape is used. The center of the shape is approximated by calculating the centroid of the traced boundary, which is the pixel obtained by averaging the horizontal and vertical coordinates of all pixels contained in the boundary.

Once a boundary is traced and its attributes determined, the scan continues at the starting pixel of the shape. Any pixels that lie within a found boundary are skipped. Once another pixel is found above the brightness threshold, the boundary tracing process is repeated. The scan ends once the final pixel of the image has been evaluated.

As the shape and form of the infrared marker is unknown, techniques such as shape recognition cannot be relied on in order to filter out false positives. Instead, any boundary for which the surface area is below a certain threshold is discarded.

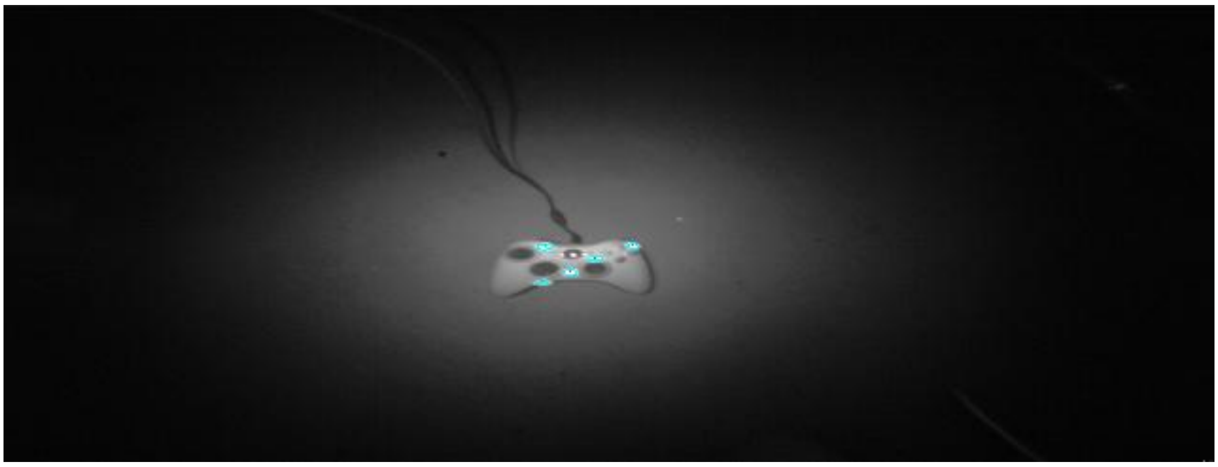

Figure 13: Detected boundaries and their center points on a marked Xbox 360 controller

3.4 Deriving 3D coordinates

In this subchapter the marker coordinates from the previous step from both camera feeds are used to determine the coordinates in 3D space for each of the markers. First the process of matching the 2D coordinates is discussed, and then a description is provided on how this information is used to calculate the 3D coordinate.

The problem of finding the matching marker in the second image for each of the markers in the first image can be constrained using epipolar geometry. Consider the triangle formed by connecting the centers of projection of the two cameras with each other and a point captured by both cameras. This epipolar plane intersects the images formed by the cameras such that the point captured by the first camera must lie on the line formed by the intersection of the epipolar plane and the second image. This limits the search area to this epipolar line.

As the Leap Motion uses a pair of parallel cameras, determining this epipolar line is simplified. For parallel cameras, images of any point must lie on same horizontal line in each image plane. In practice this will not be an exact match, but the potential matches of each marker can be limited to the markers on the other image whose y value fall within a threshold of that of the marker.

Four scenarios are possible for each marker. If no possible matches are found, the marker is discarded. If a single possible match is found, the pairing is kept. If several markers are near the epipolar line in the second image and the same amount is on the epipolar line in the first image, all markers are paired in the same horizontal order. If a different amount of markers are detected, the most promising pairing is calculated based on the surface size of each marker and the horizontal order.

Once the markers are paired, their position in 3D space can be triangulated. First, the corrected camera rays intercepting the marker position on their respective image are acquired for both of the marker coordinates. Knowing that the distance between the two cameras is 40 mm, it is then possible to determine the position in the Leap Motion coordinate system (Figure 14) with the following formulas.

- $y_m = -40/(\tan(\alpha_r) - \tan(\alpha_l))$
- $x_m = (-y_m) * \tan(\alpha_r) - 20$
- $z_m = (-y_m) * \tan(\beta_r)$

With (x_m, y_m, z_m) the 3D coordinates of the marker and $(\alpha_r, \beta_r)\ and\ (\alpha_l, \beta_l)$ the horizontal (α) and vertical (β) slopes to the pixel of the marker after correcting for lens distortion of the right and left image respectively.

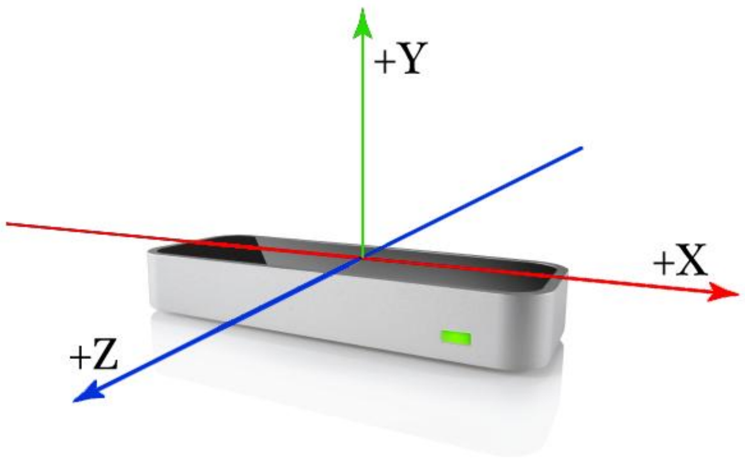

Figure 14: The Leap Motion coordinate system
[Image by Leap Motion]

3.5 Pattern matching

In this subchapter the storing of the detected 3D points as a reference pattern is described, followed by the detecting this pattern in other point formations and calculating the rotation and translation required to transform the reference pattern to the newly detected formation.

Once all the markers have been detected as 3D points, the pattern can be stored. This is intended to be done during a calibration step, where the user aligns the markers with the virtual representation of the object. This ensures tracking is functional regardless of the pattern applied by the user. If the exact pattern is known, the calibration step can be skipped by using these points directly. In addition to the 3D coordinates of the points, the distance and angle between the points are also calculated and stored as part of the reference pattern.

The first step in matching the reference pattern to other point formations is identifying which points of the other point formation correspond to which points of the reference pattern. To achieve this, a double pass voting algorithm is used. During the first pass an attempt is made to identify as many points as possible based on the inter-point distances. If one of the measured distances matches a known distance, both points gain a vote towards the points involved in the known distance. The second pass attempts to identify any point with a tied vote from the previous pass. The second pass uses the inter-point angles of the tied point and all combinations of two points identified during the first pass as the voting metric. As the aim is to identify at least three points for 6 DOF movement, it is possible to conclude no match is found if less than two points are identified after the first pass, and less than three points are identified after the second pass.

Once a sufficient amount of corresponding points are found, the Kabsch algorithm is employed to calculate the rotational matrix between the two paired point sets. This algorithm works in three steps. In the first step both point sets are translated to the origin by subtracting their centroid, the average position of all points in the set, from each point. The second step consists of calculating the 3x3 covariance matrix by multiplying the matrix consisting of the point coordinates of the first set with the transpose of the matrix consisting of the point coordinates of the second set. The third and final step consists of calculating the singular value decomposition of the covariance matrix. The rotation matrix can then be attained by multiplying the resulting right singular vectors with the transpose of the left singular vectors. It is possible that the resulting rotation matrix is in the left-handed coordinate system. This can be detected by calculating the determinant of the rotation matrix. If it is negative, the rotation matrix will need to be mirrored by multiplying the third row by -1.

The translation between the two point sets is found by calculating the difference between the centroid of each point set. The rotation is extracted from the rotation matrix using the following formulas.

- $\theta_x = arctan(r_{32} \,/\, r_{33})$
- $\theta_y = -arcsin(r_{31})$
- $\theta_z = arctan(r_{21} \,/\, r_{11})$

With $(\theta_x, \theta_y, \theta_z)$ as the three Euler angles and r as elements of the 3x3 rotation matrix.

The pattern matching algorithm used gives us the translation and rotation between the centroid of the reference pattern and the centroid of the detected pattern. During the calibration step the translation between the centroid of the reference pattern and the anchor point of the virtual representation is stored so it can be reapplied after transforming the virtual representation with the results of the pattern matching algorithm. This ensures that the mapping between the physical peripheral and the virtual representation remains accurate even if the centroid of the markers does not match the centroid of the peripheral.

Chapter 4

Implementation

The tracking algorithm was implemented in a Unity demo application intended to measure its effectiveness. In this chapter the implementation of the algorithms described in the previous chapter and their integration in the Unity game engine is discussed.
First, the implementation of the algorithms is covered, along with any optimizations and external libraries used. Then, the process and additional scripting required to implement these techniques in the Unity game engine are described.

4.1 Tracking implementation

This subchapter covers the implementation of the tracking algorithms discussed in chapter 3. The steps will be covered in the same order, starting with the identification of infrared markers from the stereo images provided by the leap motion, then triangulating the 3D coordinates of the detected markers and finally identifying the tracking pattern applied to the peripheral and calculating the translation and rotation required to align the virtual representation with the physical peripheral. The implementation was written in C# in order to ensure compatibility with the Unity game engine. Version 2 of the Leap Motion SDK was used. Version 3 of the SDK with improved hand tracking from a head-mounted perspective was available at the time of writing, but was not used due to inconsistencies in the API concerning retrieval and processing of the raw sensor data.

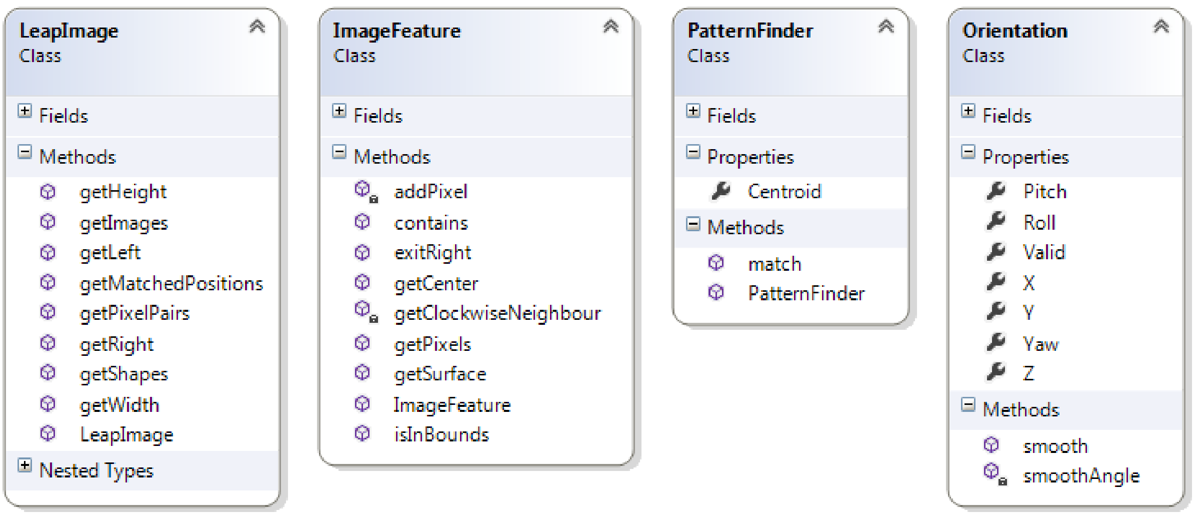

Figure 15: The class diagram of the tracking implementation

Four classes were created for the implementation. The LeapImage class is used to convert a frame of image data from the Leap Motion to a list of 3D points. A LeapImage instance is created using the constructor which takes an ImageList as argument. This ImageList can be acquired from the Leap Motion API and contains two Image objects, one for each camera. These Image objects allow access to the raw image data and contain functions to compensate for lens distortion. Requesting the list of 3D points from the LeapImage object via the getMatchedPositions method starts a call to the getPixelPairs method, which in turn first calls the getShapes method for each lens. The getShapes method scans the raw image data for any values above the brightness threshold. Once such a value is found, an ImageFeature object is created. This class requests this starting pixel in its constructor and executes a contour tracing algorithm to determine the dimensions and centroid of the detected marker. Once the getShapes method completes, this list of ImageFeature objects is returned. The getPixelPairs method can then use this list to find each corresponding ImageFeature object in the list from the other camera. A list containing the coordinates of the centroids of each pair of corresponding ImageFeature objects is then returned. Finally, these coordinates are then used to calculate the 3D coordinate of each pair of points, adjusting for lens distortion. This list of 3D points is then returned.
During the calibration step this list of 3D points is used to create a PatternFinder instance. This class accepts a list of 3D points as the argument for its constructor. This list of points is then stored as the reference pattern. The match method can then be used to compare another list of points to the reference pattern stored in the PatternFinder instance. This method returns the result of the attempted matching as an Orientation object. The Valid property of this object indicates whether or not the matching was successful. Furthermore it contains the rotation and translation necessary to transform the reference pattern to the new pattern and has a smooth method which accepts a previous Orientation as argument to help minimize the impact of noise.

The following subchapters elaborate on the described functionality of the classes, starting with the marker detection of the LeapImage and ImageFeature classes, followed by the 3D point conversion of the LeapImage class and finally closing with the pattern matching of the PatternFinder class.

4.1.1 Marker detection

During the marker detection step the image data provided by the Leap Motion cameras is scanned for any blobs with a brightness value above a certain threshold. These blobs are defined using a contour tracing algorithm and stored in an instance of the ImageFeature class for future use.

First, the image data from the left camera is acquired. This data is presented as a 1D byte array, indicating the measured brightness for each pixel from top-left to bottom-right. The elements of this array are then scanned for any pixels with a brightness value over a certain threshold. Whenever such a pixel is found, it is first checked if it is part of an ImageFeature detected earlier in the scan. If this is found to be so, the scan can be skipped forward to the first pixel not part of the ImageFeature on the same horizontal line. If the pixel is not part of any previously found ImageFeature, a new ImageFeature is created with the pixel as starting point.

When an ImageFeature object is created, the contour tracing algorithm is started. The algorithm keeps track of the current pixel and the previous pixel visited. At the start of the algorithm, the pixel used as the starting point is assigned to the current pixel and the (fictional) pixel directly to the left of the current pixel is assigned as the previous pixel. The eight pixels surrounding the current pixel are then evaluated, going clockwise starting with the pixel after the previous pixel. Once the first pixel with a brightness value over the threshold is found, this pixel is added to the contour and assigned as the new current pixel, with the old current pixel being assigned as the previous pixel. This is repeated until the first pixel is reached once more.

The pixels are stored in a dictionary of lists using the y coordinate as key. This allows for the quick retrieval of the horizontal exit point when entering an ImageFeature during the scan. As the pixels are stored, their x and y values are added to a sum which will be divided by the number of points in the boundary at the end of the scan to determine the coordinates of the centroid, and a check takes place to see if the pixel qualifies as the most extreme pixel of each of the four directions. In case the newly added pixel ties with the previous extreme for that direction, the pixel closest to the previous direction is used. For example, if there is a tie for the Northern-most pixel, the pixel furthest North-West will be selected. By using these four pixels to define a rectangle around the ImageFeature, one can quickly estimate if a pixel could be part of the ImageFeature. They are also used to estimate the surface of the ImageFeature. In order to reduce false positives, any ImageFeature consisting of an estimated area of less than four pixels is discarded.

This entire process is repeated for the right camera. While not done in this implementation, it is easy to acquire a small performance gain by processing both camera images in parallel. The results of this step are two lists of ImageFeatures, one from each camera. These will be further processed in the next step.

4.1.2 Deriving 3D coordinates

During this step the ImageFeatures detected in the image from the left camera are matched with the ImageFeatures detected in the image from the right camera. Following this, the paired positions are used to triangulate the position of the associated marker in 3D space.

As the Leap Motion uses parallel cameras, the ImageFeature detected in the left image will match an ImageFeature detected in the right image on the same horizontal line. Rather than attempting to match the entire ImageFeature, the calculated center points are compared instead. As there might be a slight difference in the shape of the detected marker in each camera which can result in the center point being on a different horizontal line, a certain amount of lines above and below the horizontal line are checked as well.

First the center points from the ImageFeature list of each camera are sorted on their y coordinate. The matching loop is then started where all points on the horizontal area of the next point of the left camera are selected. Then all points in the same horizontal area of the right camera are selected. There are three scenarios possible at this point. In the first scenario, no matching points were found from the right camera. The points from the left camera are removed from the list. In the second case, there are an equal amount of points from both cameras, in which case they are matched in the same horizontal order. In the third case, a different amount of points are found from the left and right camera. All possible combinations of points that preserve the horizontal ordering of the points are considered and a score is assigned to each combination. This score consists of the sum of the difference in surface size between the matched points. The combination with the lowest score is selected as the best match. Once any kind of matching has occurred, the matched points are removed from the list. When the list of points from the left camera is empty, the loop ends and the resulting point pairings are returned.

These point pairings are then used to triangulate the point in 3D space they represent. The Leap Motion documentation provides a code snippet for this purpose.

```
Vector slopes_left = image_left.Rectify(left_camera_pixel);
Vector slopes_right = image_right.Rectify(right_camera_pixel);

//Do the triangulation from the rectify() slopes
float cameraZ = 40/(slopes_right.x - slopes_left.x);
float cameraY = cameraZ * slopes_right.y;
float cameraX = cameraZ * slopes_right.x - 20;
Vector position = new Vector(cameraX, -cameraZ, cameraY);
```

Given a point on the image, rectify() corrects for camera distortion and returns the true direction from the camera to the source of that image point within the Leap Motion field of view. The tangent of the horizontal and vertical ray angle are returned by this method as a vector object with the z value being 0. The distance between the two cameras is 40 mm, which is used in the calculation of cameraZ and for cameraX, where it corrects for the offset of the right camera. Once each of the paired points has been processed in this way, a list of the calculated 3D points is returned.

4.1.3 Pattern matching

In this step, a PatternFinder instance containing the pattern of 3D points to track is first defined. Then, this reference pattern is compared to the newly detected 3D points of each frame. When the reference pattern is detected, the translation and rotation required to move the reference pattern onto the newly detected pattern is calculated.

To create a PatternFinder instance, a list of 3D points in the Leap Motion coordinate system needs to be supplied, such as the one acquired in the previous step. In addition to storing the set of points, the centroid is calculated as well, which is the average 3D coordinate of the points in the list. A 2D matrix is then filled with the distance between each of the points, and finally a 3D matrix is filled with the angle between any 3 points. These values can be easily calculated using the distanceTo() and angleTo() methods supplied by the Leap Motion API. The distance and angle properties will be used to identify the sampled points during pattern detection.

Once a pattern is established in a PatternFinder, it becomes possible to detect this pattern in a list of supplied points. The established pattern will be referred to as the original pattern, while the list of points that could contain a possible match will be referred to as the sampled pattern. A double pass voting algorithm is used to identify the points of the original pattern in the sample pattern.
In the first pass, possible matches are evaluated solely on inter-point distance. The distance between every pairing of sample points will be compared to every value in the 2D distance matrix created during the construction of the PatternFinder instance. If the difference between the distances is lower than the error margin, both of the sample points will receive a vote towards both of the original points. Once all point pairings have been considered, the votes are counted. Sampled points with a clear winning vote are matched to that original point. If less than two points are identified in this pass, the matching process is aborted.
The purpose of the second pass is to identify any sample points with tied results on the first pass. This is done by comparing the angle between the unidentified sample point and every combination of two identified sample points with the angle between the possible original points and those same identified points from the 3D angle matrix. If the difference between the sample angle and the original angle is within the error margin, the unidentified sample point will receive a vote towards that original point. Once all possible matches are considered, the sample point will be identified as the original point that gained the most votes, or ignored if the highest votes are tied once more.

If at least three points have been identified, it is possible to calculate the rotation matrix. The Math.NET Numerics library is used for any matrix related calculations performed in the implementation of the Kabsch algorithm. First, two lists of 3x1 matrixes are created, one for the original points and one for the sample points. Each identified sample point and their corresponding original point are then added in the Math.NET matrix format to their respective list. The resulting two lists thus contain the subset of identified points of both the original and the sample points. The centroid is then calculated for the points of each list by taking the sum of their coordinates and dividing them by the number of points. The translation to origin and the calculation of the covariance matrix is combined in a single step using the following loop.

```
Matrix<float> h = Matrix<float>.Build.Dense(3, 3, 0);
for(int i = 0; i < numPoints; i++) {
    h += (originalPoints[i] - originalCentroid) *
          (samplePoints[i] - sampleCentroid).Transpose();
}
```

With h the covariance matrix, numPoints the number of identified points, originalPoints the list of used points from the reference pattern and originalCentroid the centroid of these points, and samplePoints the list of identified sample points and sampleCentroid the centroid of these points.
The singular value decomposition of the covariance matrix is then calculated. Math.NET offers the functionality to decompose the matrix into the form $U * S * V^T$. To calculate the rotation matrix, V^T is transposed back to V, which is then multiplied with the transpose of U. A check is then performed for the reflection case by calculating the determinant of the rotation matrix. Should the determinant be negative, the rotation matrix is corrected by multiplying the third column of the matrix by -1.

With the rotation matrix available, the rotation in Euler angles can be extracted from it. This is done using the following equations.

```
orient.Yaw = Mathf.Rad2Deg * Mathf.Atan2(r[2, 1], r[2, 2]);
orient.Pitch = Mathf.Rad2Deg * Mathf.Atan2(-r[2, 0],
               Mathf.Sqrt(Mathf.Pow(r[2, 1], 2) + Mathf.Pow(r[2, 2], 2)));
orient.Roll = Mathf.Rad2Deg * Mathf.Atan2(r[1, 0], r[0, 0]);
```

With orient the Orientation object used to return the movement and r the 3x3 rotation matrix. The Mathf class is provided by the Unity game engine. The Atan2() method returns the angle whose tangent is the quotient of two specified numbers, while Rad2Deg is a constant used to convert the radians returned by Atan2() to degrees. Sqrt() and Pow() are square root and power functions respectively.
The translation is calculated by subtracting the centroid of the subset of the original points from the centroid of the identified sample points.

4.2 Integration into Unity

This subchapter covers the implementation of LeapTrack in the Unity game engine. First, the scene setup needed to facilitate the tracking script is outlined. Then, the content of the script required to apply the tracking solution to the virtual representation of the peripheral is discussed.

4.2.1 Scene setup

The Leap Motion Unity asset package contains a camera rig with VR support. This camera rig moves according to the head tracking of the VR HMD and displays the hands of the user when tracked by the Leap Motion. It also has the option of displaying the video feed of the Leap Motion. When added to the scene, the hierarchy of the camera rig as illustrated in Figure 16 becomes visible. The LMHeadMountedRig is a top-level management object, allowing for a quick setup of the camera rig by providing several presets. The TrackingSpace object it contains is responsible for any head-tracking related movement of the cameras and their alignment with the images provided by the Leap Motion. It contains an anchor point for a camera for each eye, and a central camera anchor. The cameras used differ based on the preset selected on the LMHeadMountedRig. The left and right camera are used if video pass-through from the Leap Motion is required, otherwise the center camera is used. This central anchor contains the HeadMount, an empty object containing a BugReporter and the LeapHandController, which instantiates the virtual hands used to represent the tracked hands. The QuadBackground renders the camera images captured by the Leap Motion.

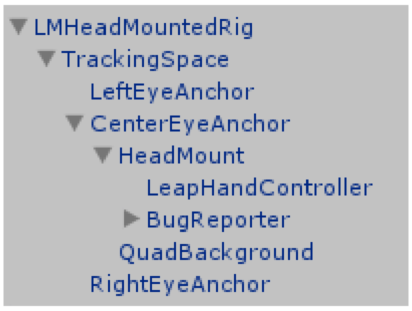
```
▼ LMHeadMountedRig
  ▼ TrackingSpace
      LeftEyeAnchor
    ▼ CenterEyeAnchor
      ▼ HeadMount
          LeapHandController
        ▶ BugReporter
        QuadBackground
      RightEyeAnchor
```

Figure 16: The Leap Motion VR camera rig hierarchy

A LeapTracker object is added to the hierarchy under the CenterEyeAnchor. This object will contain a script linking the tracking solution with the Unity game engine. By adding the tracking solution at this position of the hierarchy, it is guaranteed that the distance between the virtual representation of the tracked peripheral and the virtual hands is always concurrent with the distance between the physical peripheral and the physical hands of the user. Under the LeapTracker object an Anchor object is added. The virtual representation of the object to be tracked is added under this Anchor object. The Anchor is placed within the reach of the arms of the user in front of the camera, yet not too close that it could cause eyestrain. During the calibration phase, the user can then line up the tracked object with the virtual representation. Once the placement is confirmed, the Anchor object will be repositioned depending on the tracking data and the difference between the centroid of the reference pattern and the original Anchor position.

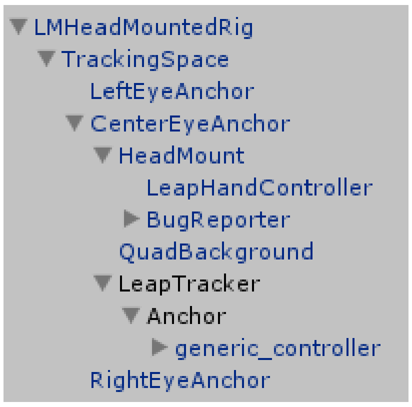
```
▼ LMHeadMountedRig
  ▼ TrackingSpace
      LeftEyeAnchor
    ▼ CenterEyeAnchor
      ▼ HeadMount
          LeapHandController
        ▶ BugReporter
        QuadBackground
      ▼ LeapTracker
        ▼ Anchor
          ▶ generic_controller
      RightEyeAnchor
```

Figure 17: The camera rig expanded with the components of the tracking solution

4.2.2 The LeapTracker script

The script attached to the LeapTracker is responsible for acquiring the image data from the Leap Motion and supplying them to the tracking solution, displaying the markers identified by the tracking solution during the calibration phase, moving the Anchor during the tracking phase and converting the coordinates from the Leap Motion right-handed coordinate system to the Unity left-handed coordinate system by using the following conversion.

```
Vector3 converted = new Vector3(vector.x * -0.001f, vector.z * 0.001f,
                                vector.y * 0.001f);
```

As the script starts up, it is necessary to indicate that the raw image data from the Leap Motion is intended to be used by enabling the relevant policy. If the user has disabled the "Allow Images" option in their Leap Motion settings, no images will be supplied regardless. During the update cycle the most recent image data is first acquired from the Leap Motion controller object. This data is then used to create a LeapImage object, from which the list of 3D points identified in the image pair is requested.

If the script is in the calibration phase, these points will be visualized using green spheres. When the user has lined these spheres up with the virtual representation of the controller and confirmed the position, a new PatternFinder object is created with the current list of points as the reference pattern. The difference between the centroid of the reference pattern and the Anchor is stored as well. In order to limit false positives, any point outside of a configurable range of the Anchor is removed from the list before creating the PatternFinder. The calibration can be reverted by setting the PatternFinder object to null and restoring the position and orientation of the Anchor to its default position.

When in the tracking phase, the list of points is used as an argument for the match() method of the PatternFinder. This method returns an Orientation object which tells us whether or not the reference pattern was recognized and the translation and rotation required to match the patterns if it was. In order to counteract jitter, a simple smoothing factor is applied (0.4 x the new value + 0.6 x the old value) if the distance between the current and previous Orientation is small enough. To apply the transformation, the local position of the anchor is first set to the sum of the centroid of the reference pattern and the transform of the Orientation. The Anchor is then rotated with the rotation from the Orientation. The anchor is then translated in the direction of the vector between the original Anchor position and the centroid of the reference pattern stored during calibration. If no pattern was recognized, the reference object is retained in its current position for up to three update calls to compensate for false negatives. If no new pattern is recognized during this timeframe, the Anchor object and its children are put in an inactive state until a pattern is recognized again.

Chapter 5

Evaluation

This chapter outlines the user study conducted to evaluate the impact of LeapTrack on the ability to locate a peripheral device and the level of immersion of the user. First, the recruitment process is described. Then, an overview of the tasks the participants were asked to complete is given. Following this, the apparatus is discussed, covering both the hardware and software components. The metrics measured are then listed. Finally, the results are presented, followed by a discussion of the collected data.

The aim of the experiment was to test the ability to access a peripheral device, such as a game controller, while being immersed in a VR scene. We tested two different conditions: Without any tracking infrastructure and with LeapTrack. LeapTrack enables the users to see a visual representation of the peripheral device in the VE.

5.1 Participants

Participants for the user study were gathered by approaching students on campus and asking if they would be interested in participating in a user study concerning VR. 12 students opted to participate, consisting of 7 males and 5 females ranging from 20 to 25 years.

5.2 Task

Participants were asked to try and access a peripheral device seven times during a short VR game. This game was run twice for each participant, the first run being executed without additional support through LeapTrack and the second run with support through LeapTrack.

When the participant arrived, they were first seated in front of a desk area upon which a game controller and VR headset were placed. The participant was then instructed to put on the VR headset and to adjust the head strap for a comfortable fit. The VE was then loaded and the user was encouraged to look around the VR scene to confirm the head tracking of the HMD was working correctly. The participant was then handed the controller and told to move forward through the VE at a pace they were comfortable with using the left analog stick. Movement directions other than forward were disabled.

After a short distance, the participant encountered the first “roadblock”.
The participants were instructed within the VR scene to look to the right and put up their hands into a designated virtual area. Once both hands are in the correct place, the

designated areas fill up in a green color. Once completely filled, the areas disappear and a confirmation message is displayed. These instructions were verbally relayed during the first roadblock along with an additional instruction to place the controller in front of them before turning to place their hands into the designated areas and to pick up the controller and continue forward once the confirmation message is displayed. These instructions were only repeated verbally during later roadblocks if the participant asked for a reminder or was unsure of what to do.

The goal of this chain of action is to incentivize the user to put down the controller without breaking immersion. The 90-degree rotation to the right is intended to disorient the user, making location the peripheral less reliant on memory. The search action is repeated at each of the seven roadblocks. The first roadblock is intended as a tutorial guided by the experimenter, the next three are performed as expected and during the final three roadblocks the controller is moved by the experimenter while the participant is turned away. As the user can rely on spatial memory for the location of the controller during the second, third and fourth search actions, this will be labeled “aware” performance, since the participant is aware of the location of the controller. Conversely, the final three search actions will be labeled “unaware” performance, since the user is no longer aware of the location of the peripheral device as it has been moved without their knowledge.

Once the test environment was “cleared” by performing these actions for each of the 7 roadblocks, the participant was asked to rate their immersion on a scale of 1 to 10. This scale was described with a 10 feeling as if they were present in the VE, a 5 feeling as if they were playing a video game and a 1 feeling as if they were in the room watching a screen on a HMD. Afterwards, the VE was restarted for the second run with the addition of the LeapTrack. Before being instructed to move, the user was told to perform a calibration step, consisting of aligning the physical controller with a virtual representation, then pressing a button to confirm the calibration. This calibration could be cancelled and retried until the participant was content with the tracking performance. After the calibration phase, the participant was asked to complete a SUS questionnaire concerning the calibration procedure. The remainder of the run then proceeded identically to the first run, albeit with the peripheral device tracking of LeapTrack enabled.

After the completion of the second run, the participant was again asked to rate their immersion on a scale of 1 to 10. Finally, they were asked to complete another SUS questionnaire, this time concerning the performance of the tracking solution.

5.3 Apparatus

This subchapter provides an overview of the hardware setup used during the user test and the software environment created to perform the experiment in.

5.3.1 Hardware

An Oculus Rift Development Kit 2 (DK2) is used as the VR headset and an Xbox 360 controller as the peripheral to track. The participant is equipped with the DK2, to which a Leap Motion is mounted using the Leap Motion VR Developer Mount. They are then seated in a swivel chair in front of a desk covered in a cardboard surface in order to minimize infrared reflection. The DK2 tracking camera is placed on an elevated surface on the desk facing the user. A cage has been constructed around the Xbox 360 controller using K'NEX from which an "antenna" extends holding an OptiTrack hand rigid body to which 4 spherical retroreflective markers are attached in a non-isogonal pattern. A monitor mirroring the images displayed on the HMD is available to track the progress of the participant. Figure 18 demonstrates the setup used in the experiment.

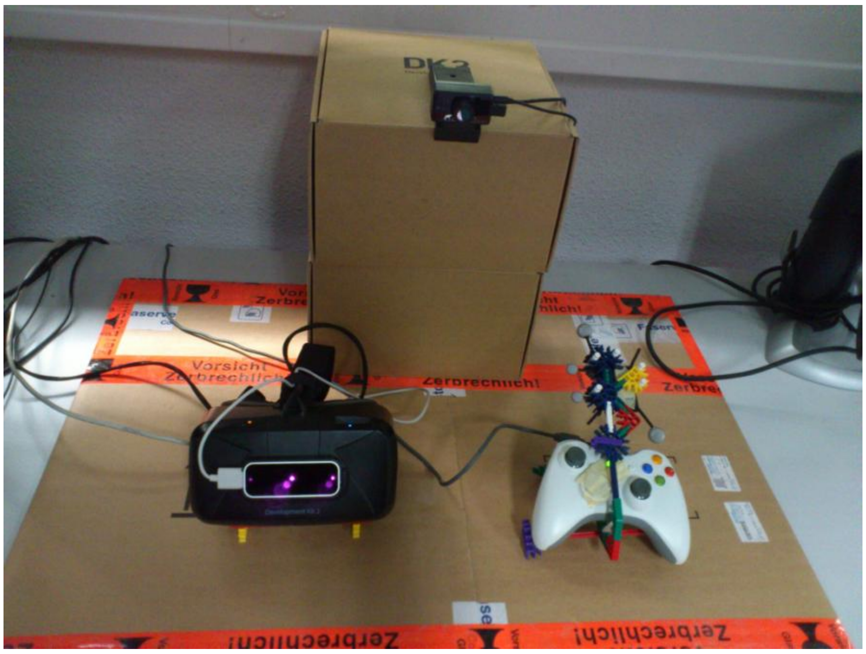

Figure 18: The hardware setup used for the user study

5.3.2 Software

A scene has been created in the Unity Game engine to serve as the virtual testing environment. A pre-fabricated scene from the Crystal Cave asset by Almgp was modified to provide a linear testing experience. The participant starts in a closed room placed on a device approximating a hovering segway, based on the SciFi Jet asset by Righteous Games. The participant is only able to move forward. In order to minimize cyber sickness, this may be done at a pace of their choice. Head tracking is not coupled to movement and the user is able to look around at will. During the course of the scene the user will traverse the starting room, a bridge through a crystalline cave and two additional rooms before ending at a grand circular room housing a giant crystal. A total of seven roadblocks will be encountered during the test, the first of which is intended as a tutorial. A layout of the level is provided in Figure 19, showing the starting point as a green dot, the roadblocks as red dots and the ending point as an orange dot.

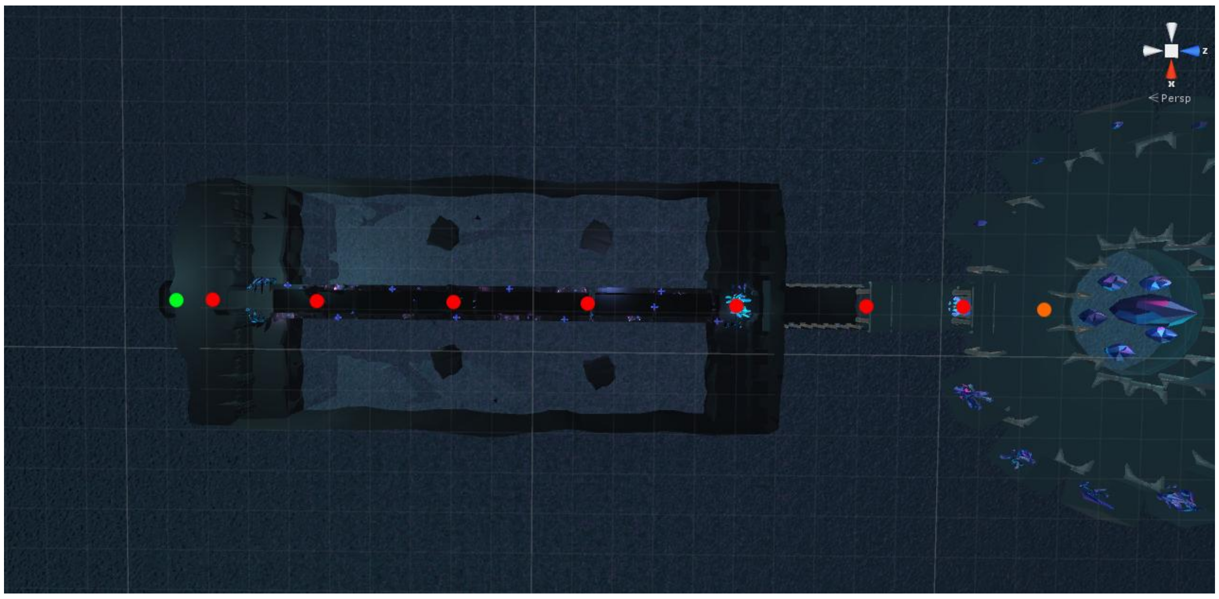

Figure 19: The layout of the virtual testing environment

At the start of the second run of experiment, the participant is tasked with calibrating the tracking solution. To assist in this task, a blended overlay of the Leap Motion camera feed is enabled and the tracked markers are visualized by green spheres. To complete the calibration, the participant must align the physical controller with the virtual representation, and then press a button on the controller to confirm the calibration. The tracking solution will provide tracking based on the markers recognized during this confirmation. At this point the participant may cancel the calibration if they find the tracking unsatisfactory and retry, or accept the calibration and continue the experiment. The number of attempts the user request to calibrate is logged, as well as the amount of markers recognized each attempt.

As the participant moves forward, they will encounter their first roadblock. At a roadblock, movement is disabled and a message is displayed asking the participant to "charge the lazers" (Figure 20). The participant is instructed to put down the controller and look to their right. Here they will see two orange boxes surrounding a text "Charge lazers" (Figure 21). When the participant places a hand in each of the boxes, they will slowly turn green. Once both boxes are completely green, the boxes disappear and the text changes to "Target destroyed" (Figure 22). The roadblock is removed and unbeknownst to the user a timer is started. The user is instructed to continue moving as soon as possible. Once the controller is found and the VE detects movement input, the timer is stopped and the time saved. This time will later be used to determine the impact on search performance of the tracking solution. Once all seven roadblocks are passed and the user reaches the end of the VE, a message is displayed thanking them for their participation.

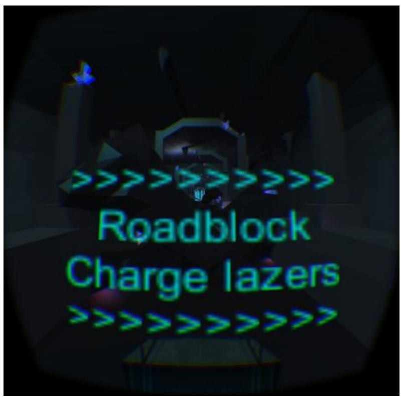

Figure 20: The sign displayed in front of the user when a roadblock is encountered

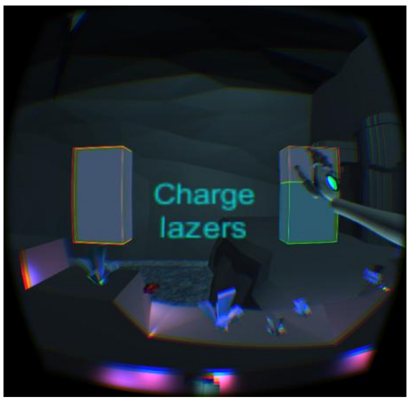

Figure 21: The charging gesture required at the roadblock

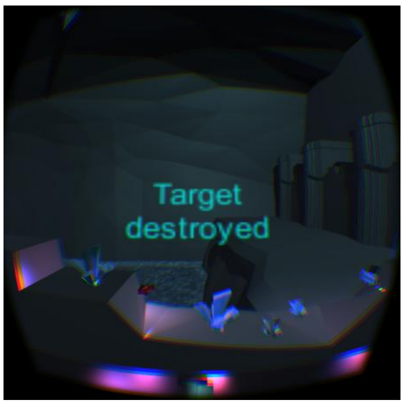

Figure 22: The confirmation message displayed once the task is completed

5.4 Measures

7 metrics were used to track the impact of LeapTrack on both search performance and user experience.

The metrics used to evaluate the influence of LeapTrack on search performance were the time taken to complete the search action at a roadblock and the number of attempts at grabbing the controller at each roadblock. The time taken was automatically measured between the completion of the charging gesture at a roadblock and the first forward input after the completion of said roadblock event. The number of grabbing attempts was counted manually. A motion was considered a grabbing attempt when one or both hands moved to a position with the intent of coming into contact with the controller. This includes the act of picking up the controller.

The impact on the user experience was measured by asking the participant to rate their immersion on a scale of 1 to 10 after both runs, along with a SUS questionnaire on the performance of the tracking solution. The impact of the calibration step was evaluated separately by automatically measuring the number of calibration attempts used and the number of infrared markers that were identified during each attempt, along with a SUS questionnaire.

5.5 Results

This subchapter first presents the data as it was gathered, and then provides an interpretation for notable measured events.

5.5.1 Data

The data gathered from each of the 7 roadblocks encountered during the user test is split into two categories. Roadblocks 2, 3 and 4 are considered aware performance, where the controller remains in the location where the user has left it. Roadblocks 5, 6 and 7 are considered unaware performance, where the controller has been moved to an unexpected location. The entries from Roadblock 1 are discarded as this is intended as a tutorial. Entries where the user did not rely on sight during the search action are not evaluated for performance with tracking enabled. This resulted in 36 entries for performance without tracking, 30 entries for aware performance with tracking and 28 entries for unaware performance with tracking.

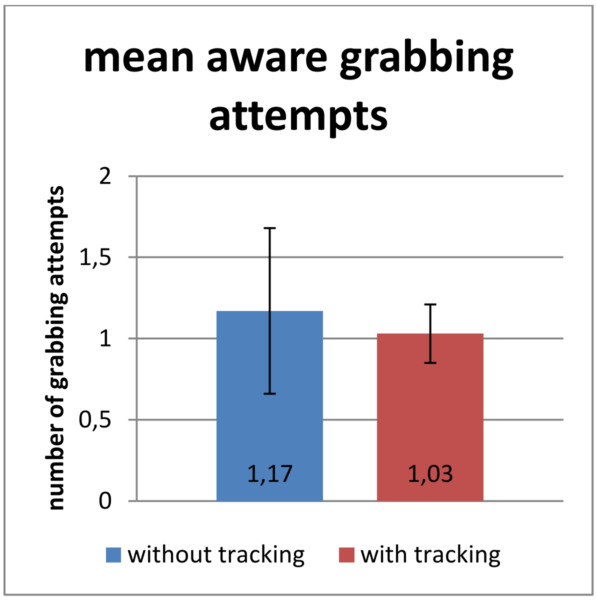

Figure 23: The mean number of grabbing attempts made during aware performance (roadblocks 2, 3 & 4)

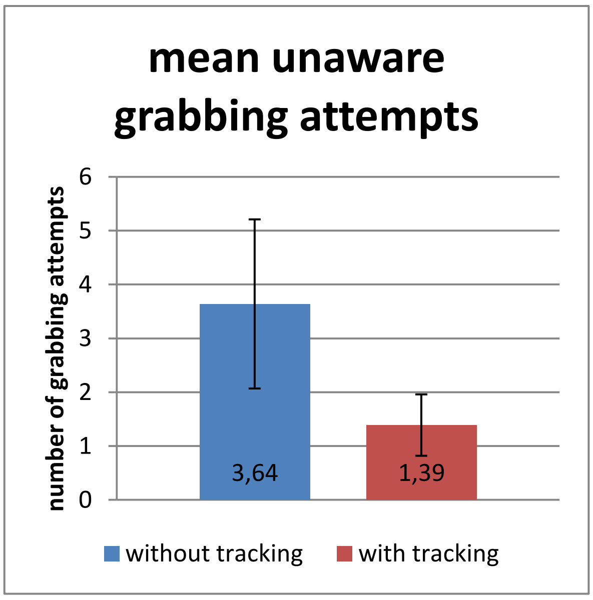

Figure 24: The mean number of grabbing attempts made during unaware performance (roadblocks 5, 6 & 7)

The mean number of grabbing motions required to find the controller decreased slightly from 1.17 to 1.03 grabbing attempts when using LeapTrack during aware performance, where the peripheral device remained in the location where the participant expected it to be. A T-test determined this decrease to not be statistically significant ($p = 0.1765$).
However, during unaware performance, where the controller was moved to a location unknown to the participant, using LeapTrack decreased the mean number of grabbing attempts much more noticeably from 3.64 to 1.39 grabbing attempts. A T-test determined this to be statistically significant ($p < 0.01$).

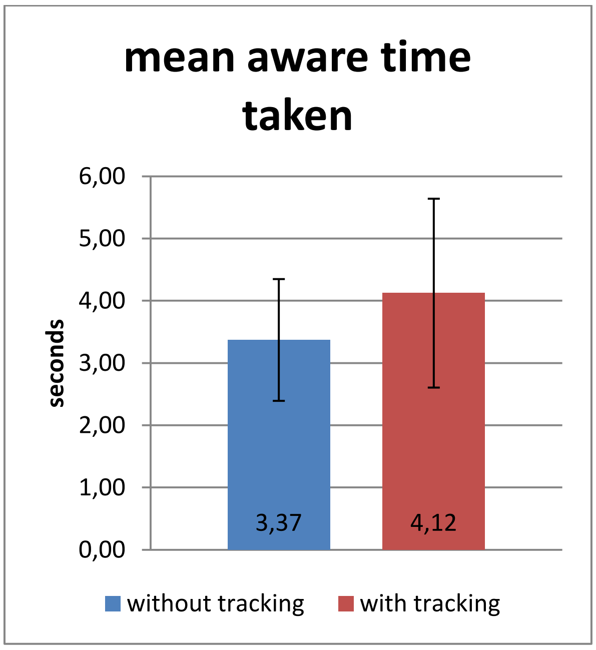

Figure 25: The mean amount of time needed to complete a the search action during aware performance

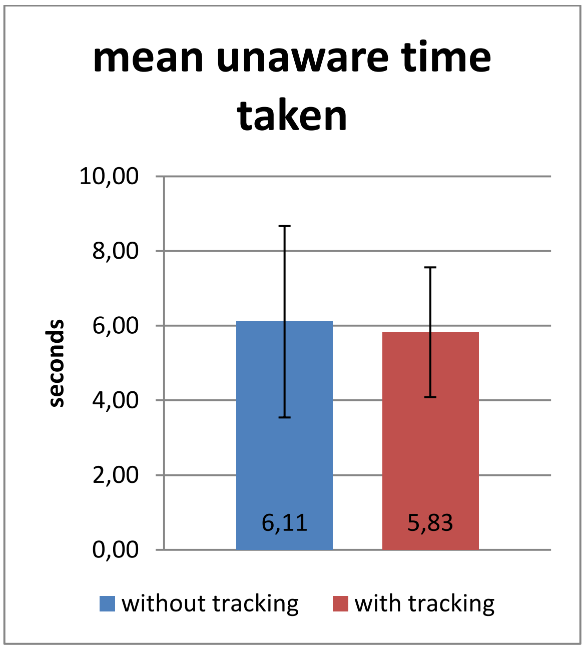

Figure 26: The mean amount of time needed to complete the search action during unaware performance

Using LeapTrack during aware performance increased the mean time taken to complete the search action from 3.37 seconds to 4.13 seconds. A T-test indicated that this was statistically significant ($p < 0.05$).
During unaware performance, LeapTrack was found to slightly decrease the mean time taken from 6.11 seconds to 5.83 seconds. Running a T-test found this to not be statistically significant ($p = 0.2810$).

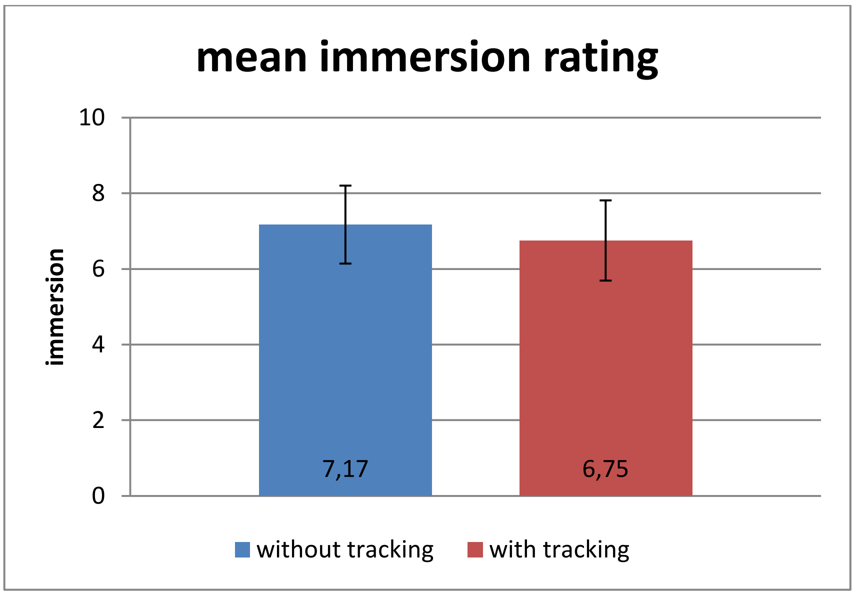

Figure 27: The mean immersion ratings of a run with LeapTrack disabled and with LeapTrack enabled

The immersion indicated by the participants was on average lower after a run with LeapTrack than without the tracking solution, dropping from an average of 7.17 without tracking to an average of 6.75 using LeapTrack. A T-test found that this was not statistically significant (p = 0.3383).

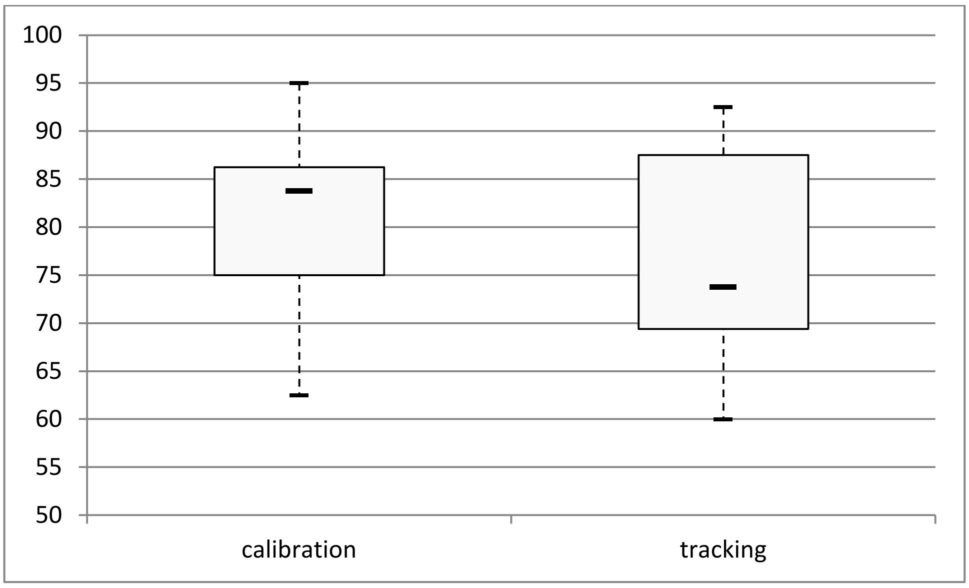

Figure 28: SUS scores of the calibration step and the overall tracking

The calibration step was generally received positively, scoring a median of 83.75 on the SUS questionnaire. 7 out of the 12 participants were content with the calibration after their first attempt and 4 continued after their second attempt. The only participant opting for more than two attempts did so because one of their fingers was being identified as an additional marker. This was fixed by increasing the brightness threshold for marker detection, after which the calibration was completed in a satisfactory manner in a single attempt. All calibration attempts succeeded with all four infrared markers identified by LeapTrack.
The tracking was less positively received, but still scored above average with a median SUS score of 73.75.

5.5.2 Discussion

Users were capable of consistently finding the controller during aware performance. The head tracking of the Oculus Rift allowed the user to utilize the virtual environment to orient themselves back to the position from where they laid down the controller, making it easy to locate the input device from spatial memory.
Introducing tracking in this situation increased the time needed to complete the pickup from an average of 3.37 seconds without tracking to an average of 4.13 seconds with tracking.

This increase can be attributed to the users first attempting to get a visual confirmation of the location of the controller before initiating the pickup attempt.
The approach most users opted for to locate the controller without tracking was to sweep over the desk area with their hands and fore-arms in an attempt to make contact with the input device. This strategy can be detrimental to the user in non-test environments, as objects that usually clutter desks could be knocked over this way, possibly damaging the object itself, the surrounding objects or even injuring the user. The chosen strategy of the user changes when the tracking solution is available, instead opting to first visually locate the object, before making a calculated grabbing attempt.

When asked to rate their immersion with tracking and without, participants reported no significant difference between the two. Some participants did remark that the presence of the virtual controller took away from the immersion, but this is likely because a standard controller model was used, rather than one customized for the VE.
Participants did remark that while the presence of the virtual controller model might have a slight negative effect on immersion, they still preferred the model be present.

The most common issue participants had with the tracking was inconsistency, likely caused by the virtual controller not immediately disappearing as the physical controller exits their field of view. As the virtual controller is linked to the head position of the user rather than to the virtual space, the controller would remain at the edge of their vision for a short amount of time. Quick head movements could thus for example result in the controller being detected to the right of the user, which would leave an afterimage in front of the user if the user made a head movement from right to left. This afterimage is a side-effect of an optimization that displays the virtual controller in the last detected location if no match is found on the current frame, for up to a certain amount of time. This optimization prevents a flickering effect from occurring if the peripheral is not found in a few frames at a time. This inconsistency can be remedied by displaying the virtual controller at the last found position in world coordinates, rather than in a position relative to the head orientation of the HMD.
Another cause of inconsistency was jitter on the virtual controller. The small amount of jitter naturally present was magnified by the use of a makeshift antenna used to suspend the infrared markers away from the controller. While this ensured that the markers would be unobscured from all viewing angles, this also had the repercussion of exaggerating any inaccuracy in the calculation of the rotation of the controller. This jitter could be reduced by moving the infrared markers closer to the controller or by employing more aggressive smoothing.

Overall the tracking solution was received positively. A recurring comment of the participants was that being able to see the controller inside the VE was something they found useful, as the feeling of where they had put down the controller had been partially lost.

Chapter 6

Conclusion

In this thesis, an infrared tracking solution intended for virtual reality head-mounted displays using the Leap Motion was presented. LeapTrack requires a single calibration to align the physical peripheral with its virtual representation after which it is able to track the peripheral with six degrees of freedom. Infrared markers of any kind can be applied to the peripheral, assuming they are able to emit or reflect a sufficient amount of infrared light. A minimum of three markers is required, placed in a non-isogonal pattern.

It was shown how to implement the tracking solution in the Unity game engine, requiring only three calls to the tracking algorithm, namely one to request a list of recognized markers, one to confirm the marker pattern to track and a final call to determine the translation and rotation of the marker list compared to the reference pattern. It was also shown how to use the data provided by the tracking solution to apply the transformation to the virtual representation of the tracked peripheral.

LeapTrack does have its limitations. Tracking is only available while the peripheral is in the field of view of the user. As no persistent tracking is available, the virtual representation of the tracked peripheral cannot be relied on for continuous effects such as lighting. It also requires markers to be added to the physical peripheral, which might be difficult or unwanted depending on the peripheral. Finally, as an infrared tracking solution, it is vulnerable to the lighting conditions of the tracking environment. Tracking might not be accurate if used in an environment flooded with infrared light or reflective surfaces.

Finally, a user study was conducted requiring the participants to perform a search action of the peripheral, both with and without LeapTrack. Users responded positively to both the tracking and the calibration step, and preferred the tracked peripheral be present in the virtual environment. While the participants did take longer to find the peripheral with tracking enabled, they were able to pick up the peripheral with much greater accuracy.

References

[1] "Definition of VIRTUAL REALITY." [Online]. Available: http://www.merriam-webster.com/dictionary/virtual+reality.

[2] I. E. Sutherland, "A Head-mounted Three Dimensional Display," in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, New York, NY, USA, 1968, pp. 757–764.

[3] Plato, *The Republic*. Athens.

[4] T. A. DeFanti, D. Acevedo, R. A. Ainsworth, M. D. Brown, S. Cutchin, G. Dawe, K.-U. Doerr, A. Johnson, C. Knox, R. Kooima, F. Kuester, J. Leigh, L. Long, P. Otto, V. Petrovic, K. Ponto, A. Prudhomme, R. Rao, L. Renambot, D. J. Sandin, J. P. Schulze, L. Smarr, M. Srinivasan, P. Weber, and G. Wickham, "The future of the CAVE," *Cent. Eur. J. Eng.*, vol. 1, no. 1, pp. 16–37, Nov. 2010.

[5] K. J. Fernandes, V. Raja, and J. Eyre, "Cybersphere: The Fully Immersive Spherical Projection System," *Commun ACM*, vol. 46, no. 9, pp. 141–146, Sep. 2003.

[6] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1993, pp. 135–142.

[7] "Amiga History Guide." [Online]. Available: http://www.amigahistory.co.uk/virtuality.html. [Accessed: 14-May-2016].

[8] C. E. Engler, "Affordable VR by 1994," no. 100, pp. 80–81, Nov-1992.

[9] "Gaming History: arcade, video games, slots and more." [Online]. Available: http://www.arcade-history.com/. [Accessed: 14-May-2016].

[10] M. Zachara and J. P. Zagal, "Challenges for success in stereo gaming: a Virtual Boy case study," 2009, p. 99.

[11] "Oculus Rift: Step Into the Game," *Kickstarter*. [Online]. Available: https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game.

[12] "Oculus Rift | Oculus." [Online]. Available: https://www.oculus.com/en-us/rift/.

[13] "Vive | Home." [Online]. Available: http://www.htcvive.com/eu/.

[14] "PlayStation VR," *Playstation*. [Online]. Available: https://www.playstation.com/en-us/explore/playstation-vr/.

[15] "Gear VR | Samsung US," *Samsung Electronics America*. [Online]. Available: http://www.samsung.com/us/explore/gear-vr/.

[16] D. Peterson and C. Robertson, "UTILIZING VIRTUAL REALITY IN TEACHING AND TRAINING: PROGRESS TOWARD VIRTUAL SURGICAL SIMULATIONS," *INTED2013 Proc.*, pp. 3285–3291, 2013.

[17] M. L. Bymer, "Virtual reality used to train Soldiers in new training simulator | Article | The United States Army," 01-Aug-2012. [Online]. Available: http://www.army.mil/article/84453/.

[18] F. P. Brooks, "What's real about virtual reality?," *IEEE Comput. Graph. Appl.*, vol. 19, no. 6, pp. 16–27, Dec. 1999.

[19] A. Gorini and G. Riva, "Virtual reality in anxiety disorders: the past and the future," *Expert Rev. Neurother.*, vol. 8, no. 2, pp. 215–233, Feb. 2008.

[20] M. Teras and S. Raghunathan, "BIG DATA VISUALISATION IN IMMERSIVE VIRTUAL REALITY ENVIRONMENTS: EMBODIED PHENOMENOLOGICAL PERSPECTIVES TO INTERACTION," *ICTACT J. Soft Comput.*, vol. 5, no. 4, pp. 1009–1015, Jul. 2015.

[21] R. Sacks, J. Whyte, D. Swissa, G. Raviv, W. Zhou, and A. Shapira, “Safety by design: dialogues between designers and builders using virtual reality,” *Constr. Manag. Econ.*, vol. 33, no. 1, pp. 55–72, Jan. 2015.

[22] J. Seibert, “An exploratory study on virtual reality head mounted displays and their impact on player presence.,” Thesis, 2014.

[23] J. Mattiasson and D. Lu, *How does Head Mounted Displays affect users’ expressed sense of in-game enjoyment*. 2013.

[24] W. Wirth, T. Hartmann, S. Böcking, P. Vorderer, C. Klimmt, H. Schramm, T. Saari, J. Laarni, N. Ravaja, F. R. Gouveia, F. Biocca, A. Sacau, L. Jäncke, T. Baumgartner, and P. Jäncke, “A Process Model of the Formation of Spatial Presence Experiences,” *Media Psychol.*, vol. 9, no. 3, pp. 493–525, May 2007.

[25] M. Mine, “Towards Virtual Reality for the Masses: 10 Years of Research at Disney’s VR Studio,” in *Proceedings of the Workshop on Virtual Environments 2003*, New York, NY, USA, 2003, pp. 11–17.

[26] R. Pausch, J. Snoddy, R. Taylor, S. Watson, and E. Haseltine, “Disney’s Aladdin: First Steps Toward Storytelling in Virtual Reality,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1996, pp. 193–203.

[27] “The New Revolution Virtual Reality Coaster |.” [Online]. Available: https://www.sixflags.com/magicmountain/attractions/vr/overview. [Accessed: 16-May-2016].

[28] M. Marozas, V. Marozas, S. Stanaitis, I. Ulozienė, V. Ulozas, M. Šileikaitė, and R. Liutkevičienė, “Virtual reality approach for testing dynamic visual acuity,” *Biomed. Eng. 2015*, vol. 19, no. 1, Feb. 2016.

[29] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, “Augmented reality: a class of displays on the reality-virtuality continuum,” 1995, pp. 282–292.

[30] L. Motion, “Leap Motion.” [Online]. Available: https://www.leapmotion.com/.

[31] M. M. Wloka and E. Greenfield, “The Virtual Tricorder,” Brown University, Providence, RI, USA, 1995.

[32] M. Billinghurst, S. Baldis, L. Matheson, and M. Philips, “3D Palette: A Virtual Reality Content Creation Tool,” in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, New York, NY, USA, 1997, pp. 155–156.

[33] M. Ribo, A. Pinz, and A. L. Fuhrmann, “A new optical tracking system for virtual and augmented reality applications,” in *Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference, 2001. IMTC 2001*, 2001, vol. 3, pp. 1932–1936 vol.3.

[34] Y.-W. Chow, “The Wii Remote as an input device for 3D interaction in immersive head-mounted display virtual reality,” *Fac. Inform. - Pap. Arch.*, pp. 85–92, Jan. 2008.

[35] O. Ariza, P. Lubos, F. Steinicke, and G. Bruder, “Ring-shaped Haptic Device with Vibrotactile Feedback Patterns to Support Natural Spatial Interaction,” in *Proceedings of the 25th International Conference on Artificial Reality and Telexistence and 20th Eurographics Symposium on Virtual Environments*, Aire-la-Ville, Switzerland, Switzerland, 2015, pp. 175–181.

[36] M. McGill, D. Boland, R. Murray-Smith, and S. Brewster, “A Dose of Reality: Overcoming Usability Challenges in VR Head-Mounted Displays,” 2015, pp. 2143–2152.

[37] M. Azmandian, M. Hancock, H. Benko, E. Ofek, and A. D. Wilson, “Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences,” in

Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, 2016, pp. 1968–1979.

[38] B. Araujo, R. Jota, V. Perumal, J. X. Yao, K. Singh, and D. Wigdor, "Snake Charmer: Physically Enabling Virtual Objects," in *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*, New York, NY, USA, 2016, pp. 218–226.

[39] S. Greuter and D. J. Roberts, "SpaceWalk: Movement and Interaction in Virtual Space with Commodity Hardware," in *Proceedings of the 2014 Conference on Interactive Entertainment*, New York, NY, USA, 2014, pp. 1:1–1:7.

[40] P.-W. Lee, H.-Y. Wang, Y.-C. Tung, J.-W. Lin, and A. Valstar, "TranSection: Hand-Based Interaction for Playing a Game Within a Virtual Reality Game," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA, 2015, pp. 73–76.

[41] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa, "The Go-go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR," in *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 1996, pp. 79–80.

[42] B. Petry and J. Huber, "Towards Effective Interaction with Omnidirectional Videos Using Immersive Virtual Reality Headsets," in *Proceedings of the 6th Augmented Human International Conference*, New York, NY, USA, 2015, pp. 217–218.

[43] A. Esteves, E. Velloso, A. Bulling, and H. Gellersen, "Orbits: Gaze Interaction for Smart Watches Using Smooth Pursuit Eye Movements," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, New York, NY, USA, 2015, pp. 457–466.

[44] M. Stengel, S. Grogorick, M. Eisemann, E. Eisemann, and M. A. Magnor, "An Affordable Solution for Binocular Eye Tracking and Calibration in Head-mounted Displays," in *Proceedings of the 23rd ACM International Conference on Multimedia*, New York, NY, USA, 2015, pp. 15–24.

[45] R. Xiao and H. Benko, "Augmenting the Field-of-View of Head-Mounted Displays with Sparse Peripheral Displays," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2016, pp. 1221–1232.

[46] Z. Zhongxiang, L. Shanshan, Y. Jiali, and S. Zhenghe, "Application of Speech Recognition Technology to Virtual Reality System," 2015.

[47] D. Ashbrook, P. Baudisch, and S. White, "Nenya: Subtle and Eyes-free Mobile Input with a Magnetically-tracked Finger Ring," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2011, pp. 2043–2046.

[48] D. Dobbelstein, P. Hock, and E. Rukzio, "Belt: An Unobtrusive Touch Input Device for Head-worn Displays," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, 2015, pp. 2135–2138.

[49] M. Weigel, T. Lu, G. Bailly, A. Oulasvirta, C. Majidi, and J. Steimle, "iSkin: Flexible, Stretchable and Visually Customizable On-Body Touch Sensors for Mobile Computing," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, 2015, pp. 2991–3000.

[50] P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, *Usability Evaluation In Industry*. CRC Press, 1996.