



CHALMERS



Maskin-till-maskin-kommunikation för Augmented Reality

Samverkan mellan HoloLens och en högprecisions IR-kamera
Examensarbete inom Data- och Informationsteknik

JOAKIM JÄRLEKLINT, STEFAN VUKMAN

EXAMENSARBETE 2018:04

Maskin-till-maskin-kommunikation för Augmented Reality

Samverkan mellan HoloLens och en högprecisions IR-kamera

JOAKIM JÄRLEKLINT, STEFAN VUKMAN



Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2018

Maskin-till-maskin-kommunikation för Augmented Reality
Samverkan mellan HoloLens och en högprecisions IR-kamera
JOAKIM JÄRLEKLINT, STEFAN VUKMAN

© JOAKIM JÄRLEKLINT, STEFAN VUKMAN, 2018.

Handledare: Sakib SisteK, Institutionen för Data- och Informationsteknik
Examinator: Peter Lundin, Institutionen för Data- och Informationsteknik

EXAMENSARBETE 2018:04
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telfon: +46 31 772 1000

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Omslag: Illustration av en operatör vid monteringsarbete, samt MQTT kommunikationen mellan HoloLens och Nexonarsystemet.

Institutionen för Data- och Informationsteknik Göteborg 2018

Maskin-till-maskin-kommunikation för Augmented Reality
Samverkan mellan HoloLens och en högprecisions IR-kamera
JOAKIM JÄRLEKLINT, STEFAN VUKMAN
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola

Sammanfattning

Detta arbete är gjort på företaget Virtual Manufacturing och ämnar att redogöra hur en kommunikation mellan en positionsbestämmande infraröd-kamera och augmented reality-glasögon kan fungera. För att undersöka detta har en applikation tagits fram som möjliggör för användaren att utföra monteringsarbeten utan någon större tidigare erfarenhet. Monteringsarbetet görs med hjälp av instruktioner och hologram som visas i AR-glasögonen samtidigt som IR-kameran övervakar att arbetet blir utfört på ett korrekt sätt. Applikationen är utvecklad för AR-glasögonen Microsoft HoloLens[1] samt en IR-kamera från företaget Nexonar[3] som med hjälp av IR-trackers håller koll på användarens händer och verktyg. Dessa kommunicerar genom MQTT-protokollet[4] och skapar tillsammans en väl fungerande applikation som underlättar vid upplärning samt ökar kvalitetssäkring.

Keywords: MQTT, HoloLens, Nexonar, IR camera

Maskin-till-maskin-kommunikation för Augmented Reality
Samverkan mellan HoloLens och en högprecisions IR-kamera
JOAKIM JÄRLEKLINT, STEFAN VUKMAN
Department of Computer science and engineering
Chalmers University of Technology

Abstract

This work has been carried out at Virtual Manufacturing and is destined to narrate how an communication between a position determining infrared camera and augmented reality glasses can be done and used. To investigate this, an application has been built to help a person with assembly work without any further knowledge in beforehand. This is made possible with the help of instructions and holograms from the AR glasses and with the surveillance of the IR camera confirming that the work is done properly. The application is built for the Microsoft HoloLens[1] and an IR camera from Nexonar[3] that can, with the help of IR-trackers, keep track of the users hands or tools. The communication between these two are carried out with the MQTT protocol[4] and together they make a well functioning application which helps in the process of teaching and raises quality assurance.

Keywords: MQTT, HoloLens, Nexonar, IR camera

Förord

Vi vill tacka vår handledare på Chalmers, Sakib Sistek, samt vår handledare på Virtual Manufacturing, Daniel Muhr.

Joakim Järleklint, Stefan Vukman, Göteborg, Maj 2018

Innehåll

Terminologi	xiii
Figurer	xv
Tabeller	xvii
1 Inledning	1
1.1 Bakgrund	1
1.2 Syfte	1
1.3 Mål	1
1.4 Problem	2
1.5 Avgränsningar	2
2 Teori	3
2.1 Augmented Reality	3
2.2 Scrum	3
2.2.1 Trello	4
2.3 Microsoft Hololens	4
2.3.1 Hårdvara	5
2.3.2 Inbyggda funktioner	5
2.3.2.1 Gaze	6
2.3.2.2 Gester	6
2.3.2.3 Röstkommando	6
2.4 Nexonar	6
2.4.1 IR SCT	6
2.4.2 Hard Probe	7
2.5 Mjukvara	7
2.5.1 Unity	7
2.5.2 Viusal Studio	8
2.5.3 MTM-tracker/Assembly Scout	8
2.5.3.1 Workspace	8
2.5.3.2 Recepie	9
2.5.4 Motion Visualizer	9
2.5.5 MQTT	10
2.5.5.1 M2MQTT	10
2.5.6 Microsoft HoloLens Emulator	11
2.5.7 Vuforia Augmented Reality SDK	12

2.5.7.1	ImageTarget	12
2.5.7.2	Bildigenkänning	12
3	Metod	15
3.1	Arbetsmetod	15
3.2	Tidsplanering	15
3.3	Informationssökning	15
3.4	Verktyg	15
3.5	Laboration	16
3.5.1	Hololens och MQTT	16
3.5.2	Sammankoppling av Hololens och Nexonar	16
3.6	Utveckling av produkten	16
4	Genomförnade	17
4.1	Förstudie och utveckling av arbetsmoment	17
4.2	MQTT	17
4.2.1	Upprättande av MQTT-broker	17
4.2.2	MQTT kompatibilitet Nexonar	18
4.2.2.1	MQTT med Assembly scout	19
4.3	IR-kameran, Nexonar	20
4.3.1	Utformande av arbetsstationen	20
4.3.2	Utveckling i MTM-tracker	20
4.3.2.1	Inställning av IR-trackers	20
4.3.2.2	Skapande av workspace	21
4.3.2.3	Skapande av recept	22
4.3.3	Motion Visualizer och 3D-koordinater i realtid	23
4.4	Unity	24
4.4.1	Systemfiler för MQTT och Unity	24
4.4.2	Implementering av Vuforia SDK	25
4.4.2.1	ARCamera	25
4.4.2.2	ImageTarget	26
4.4.2.3	Val av ImageTargets	27
5	Resultat	29
6	Diskussion och slutsats	31
6.1	Vad som hade kunnat göras annorlunda	33
7	Miljö och etik	35
8	Vidareutveckling	37
	Referenser	39

Terminologi

- AR - Augmented reality
- VR - Virtual Reality
- HL - Hololens
- MQTT - Message Queuing Telemetry Transport
- VS - Visual Studio
- MV - Motion Visualizer
- VM - Virtual Manufacturing
- TCP - Transmission Control Protocol
- IP - Internetprotokoll
- IR - Infraröd
- UI - User interface

Figurer

2.1	Scrumboard i Trello	4
2.2	Microsoft HoloLens	5
2.3	IR-trackers av två olika storlekar i två olika utföranden, på armband och verktyg	7
2.4	Blå ruta: Workspace. Röd ruta: Panel. Grön cirkel: Position	8
2.5	Ett recept i MTM-tracker	9
2.6	Hololens Emulator	11
2.7	Jämförelse av olika bilder. Bild 1 till 3, från vänster till höger	13
2.8	QR-kod	14
4.1	Mosquitto installation	18
4.2	MQTT inställningar i MTM-tracker	19
4.3	Plockstationen till vänster och monteringsstationen till höger	20
4.4	Substep	22
4.5	.NET framework 3.5	24
4.6	Inspector fönstret	25
4.7	Inställningar för Vuforia	25
4.8	Inställningar för vuforia	26
5.1	Ett kors för anvisning av monteringsposition	29
5.2	Plockandet av artiklar	30

Tabeller

4.1	Monteringsytans positioner:	21
4.2	Plockstationens positioner:	21

1

Inledning

1.1 Bakgrund

Projektet kommer utföras på företaget Virtual Manufacturing som arbetar med leanbaserade produktions- och logistiktjänster. Deras mål är att optimera kundernas produktion med hjälp av nytänkande och innovativa tjänster. Detta kan förslagsvis uppnås genom att modern teknik vävs in i företaget både virtuellt och fysiskt, före och efter produktionsstart. Arbetet innehåller augmented reality glasögonen Hololens[1] och en IR-kamera från Nexonar[3]. HoloLens är utvecklad av Microsoft[2] och programmeringen görs med hjälp av spelmotorn Unity. IR-kameran är utvecklad av Nexonar och är ett verktyg för att avläsa precisa rörelser i rummet. Huvuduppgiften är att undersöka hur dessa två system skulle kunna kommunicera med hjälp av kommunikationsprotokollet MQTT[4].

1.2 Syfte

Projektets syfte är att visa hur Microsoft HoloLens och Nexonars IR-kamera kan hjälpas åt att skapa en väl fungerande applikation för att assistera en montör, både genom ett informationsflöde av anvisningar och positionsövervakning av händer och verktyg. Med hjälp av dessa två system vill uppdragsgivaren se huruvida kommunikationen mellan systemen är möjlig samt utvärdera hur bra dess eventuella tillämpningar i verkstadsindustri är.

1.3 Mål

Målet är att skapa en kommunikation mellan Hololens[HL] och Nexonar för en applikation där HL instruerar användaren och Nexonar verifiera att uppgiften utförs korrekt. Applikationen ska steg för steg ge användaren anvisningar om vad som ska utföras och stega vidare i processen först när ett godkännande från Nexonar är skickat. För att detta ska fungera ska ett lämpligt kommunikationsprotokoll användas. Möjligheten att skapa instruktioner för nya monteringsprogram via HoloLens ska utvärderas.

1.4 Problem

HoloLens har inte möjlighet att avgöra positionering med tillräckligt hög precision för att övervaka vissa arbetsmoment. Nexonar är högpresterande inom positionering men användargränssnittet och möjlighet att ge användaren löpande information på ett smidigt sätt är begränsat.

För att lösa dessa problem ämnas följande punkter att besvaras:

1. Kan HoloLens och Nexonar samarbeta på ett bra sätt?
2. Är MQTT-protokollet ett lämpligt kommunikationsprotokoll för samarbetet?
3. Är teknikens prestanda tillräcklig?
4. Är HoloLens ett lämpligt verktyg för att instruera en operatör?

1.5 Avgränsningar

Följande punkter kommer projektet inte att täcka:

- VR-teknologi eller andra AR-plattformar än Hololens egna
- Andra applikationer än den som är beskriven i 1.3
- Arbete med hårdvaruutveckling
- Användandet av fler än en styck Hololens och en styck Nexonar
- Andra kommunikationsprotokoll än MQTT

2

Teori

Detta kapitel avser att beskriva hårdvarna och mjukvaran som kommer att behandlas under projektets gång. För att inte skapa förvirring är många engelska ord ej översatta.

2.1 Augumented Reality

Augmented reality[AR] är när digital information kombineras med användarens verklighet i realtid. Själva uttrycket Augmented reality myntades 1990 av Tom Caudell[5] och översättningen som ofta används på svenska är förstärkt verklighet. Definitionen av AR innehåller tre kriterier. Sinnesintrycken från den reella världen och den digitala ska komma till användaren på samma gång, de båda världarna ska överensstämma med varandra geometriskt och allt detta ska ske i realtid[6]. Tekniken har länge varit något som många forskare velat utveckla men det är först på senare år som den faktiskt börjat bli så pass bra att det finns rimliga användningsområden för den.

Spelet Pokemon Go, som släpptes under 2016, är ett spel för mobiltelefoner där AR-tekniken används. Med hjälp av mobiltelefonens kamera projiceras monster i den verkliga världen på telefonens display. Detta spel gav publiciteten kring AR ett stort uppsving där den bredare allmänheten fick uppleva och förstå vad tekniken har att erbjuda.

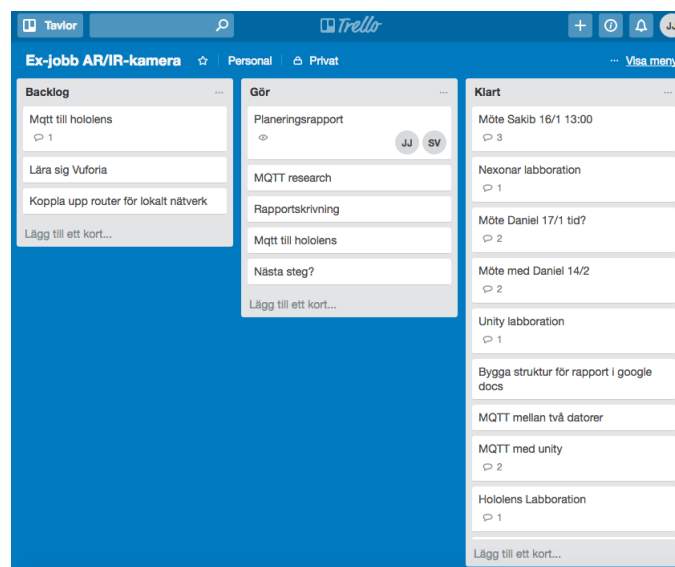
2.2 Scrum

Scrum är ett arbetssätt utvecklat för systemutveckling i form av grupparbete för att lösa komplexa projekt[12]. Arbetssättet fokuserar på vad som ska utvecklas mer än hur det ska utvecklas. Arbetet kretsar runt en så kallad backlog som fungerar som en önskelista för vad som ska göras i projektet. Det finns ingen begränsning på antal önskemål i backlogen, önskemålen prioriteras dock olika högt. Sakerna i backlogen kan både strykas och läggas till under projektets gång vilket gör det dynamiskt. För att gå vidare och utföra saker ur backlogen hålls sprint-möten för att tillsammans bestämma hur en ska genomföra dessa. En tidsram sätts för när uppgiften ska vara löst och detta kallas för en sprint. Varje dag hålls ett möte där uppgifterna i sprinten kan diskuteras och se till att dessa blir gjorda. När sprinten är klar ska produkten vara potentiellt säljbar till kund. Nästa steg blir då ha en genomgång av sprinten för kollegor eller kunder för att alla ska få förståelse. Sist görs en återblick av hela

processen för att utvärdera vad som gick bra och dåligt under arbetets gång. Om det finns negativa saker väljs några ut och åtgärdas i kommande sprint.

2.2.1 Trello

För att utföra Scrum finns det diverse hjälpmedel, ett populärt verktyg är online-plattformen Trello som är en online scrum board. Här går det att skapa allt som behövs och enkelt lägga till kommentarer och annan info på alla uppgifter. Saker ur backloggen kan flyttas till Gör som är sprinten för att sedan hamna i Klart för arkivering och senare vara en del i uppföljningsprocessen.



Figur 2.1: Scrumboard i Trello

2.3 Microsoft Hololens

HoloLens är augmented reality glasögon, utvecklade och producerade av Microsoft. HL är i stora drag en dator som är inbyggd i ett par glasögon/headset som låter användaren se, höra och interagera med en förstärkt verklighet. Den förstärkta verkligheten är en blandning av det rum användaren befinner sig i och dom hologram som projiceras av HL. Operativsystemet som används är windows 10 och där används Microsofts egna plattform Windows mixed Reality.

2.3.1 Hårdvara

Glasögonen är större än vanliga glasögon och kan ses som ett headset. Detta möjliggör att en kraftig mikrodator med tillhörande komponenter får plats för att skapa helhetsintrycket som är AR. Den display som projicerar hologrammen uppdateras 240 gånger i sekunden uppdelat på fyra olika rutor med färg vilket resulterar i en FPS(frames per second) på 60[21]. Microsoft rekommenderar att avståndet mellan användaren och ett virtuellt objekt skall vara minst 0,85m[21].

HoloLensen innehåller bland annat[7]:

- Genomskinliga holografiska linser för att visa projektionerna
- Inertial measurement unit (IMU) som innehåller gyrometer, accelerometer och magnetometer för att kartlägga förflyttning av användaren
- Fyra omgivningsförstående kameror
- En djupseendekamera
- En kamera med 1 megapixels upplösning för foto och HD-video inspelning
- Fyra mikrofoner
- En omgivningsljussensor

Mikrodatorn i sig innehåller följande komponenter:

- Intel Atom 1.04 GHz processor
- Holographic Processing Unit, en för HL specialbyggd grafikprocessor
- 2 GB ram
- 64 GB flashlagringsdisk



Figur 2.2: Microsoft HoloLens

2.3.2 Inbyggda funktioner

För att hjälpa användaren att manövrera HL på bästa sätt finns det fördefinierade funktioner som beskrivs nedan.

2.3.2.1 Gaze

Detta är motsvarande en muspekare på en dator och kan ses som en laserpekare eller vektor som är riktad rakt fram i riktning av användarens huvud[23]. Gaze används för att bestämma vad användaren vill göra med ett objekt, likt i den riktiga världen där man vanligtvis tittar på det objekt man har för avsikt att interagera med. När HL vet vad användaren tittar på kan detta användas i applikationer för att integrera med hologram eller användas av spatial mapping.¹

2.3.2.2 Gester

Detta kan ses som ett klick på vänster muspekare på en PC. Gester arbetar tillsammans med Gazefunktionen för att bestämma vad användaren vill göra[24]. Genom att användaren riktar sin sikt mot ett hologram och använder gesten Air Tap kan hologrammet flyttas, förstöras och förminskas. En annan användbar gest är Bloom som fungerar som en hem knapp, likt windows-knappen på en PC.

2.3.2.3 Röstkommando

Röstkommandon kan användas som komplement till Gaze och Gestures och kan programmeras att utföra önskade funktioner[25]. Det finns bland annat en dikteringsfunktion där man kan tala in den text man vill få dokumenterad eller kommunicerad.

2.4 Nexonar

Nexonar är en produktserie som är utvecklad av soft2tec GmbH. Produkten är ett verktyg för att läsa av positioner i ett tredimensionellt plan. En infraröd kamera gör detta genom att övervaka så kallade IR-trackers. Dessa bärs av användaren på armen, verktyg eller vad som än är intressant att övervaka. Med hjälp av tillhörande mjukvara kan man i sin tur bestämma hur denna information ska behandlas. Verktygets användningsområden återfinns i sport, medicin, industri och kvalitetskontroll[10].

2.4.1 IR SCT

Nexonar IR SCT(Single Camera Tracking) är en kamera som identifierar infraröda dioder i 3D-rummet med hög noggrannhet. Kameran kan följa upp till 16 olika IR-trackers samtidigt med en noggrannhet upp till 0.2mm i X- och Y-led, och 0.5mm i Z-led. Kameravinkeln i förhållande till avståndet till ytan uppgår till 79 grader i X-led och 63 grader i Y-led. Det maximala avståndet är specificerat att kunna arbeta mellan 0.2-3 m i Z-led, beroende på ljusförhållanden kan kameran arbeta på avstånd längre än 3 m. Kameran kopplas in till en dator som kör någon av Nexonars tillhörande mjukvaror.

¹Med spatial mapping menas den funktion i HL som kartlägger specificerad information om den riktiga världens ytor

2.4.2 Hard Probe

Hard Probe är en IR-tracker som hjälper IR-kameran att läsa 3D-positioner i rummet. Modulen bärs av användaren, eller av ett verktyg vars positionsdata skall läsas av och senare utvärderas. Mätningens noggrannhet uppgår till runt 0.3mm. För att uppnå en bra precision måste modulen kalibreras inom kameravinkeln, detta görs med hjälp av någon av dom tillhörande mjukvarorna.



Figur 2.3: IR-trackers av två olika storlekar i två olika utföranden, på armband och verktyg

2.5 Mjukvara

2.5.1 Unity

Unity är en multiplattformens spelmotor utvecklad av Unity Technologies. Här kan tvådimensionella såväl som tredimensionella spel skapas till över 25 olika plattformar[9], däribland Microsoft HoloLens. Utvecklingen sker med en blandning av drag-and-drop funktioner, UI-baserade inställningar och C# programmering.

För att kunna utföra ett effektivt arbete i grupp inom Unity-projekt erbjuder Unity deras tjänst Unity Teams[20]. Det är en cloud-baserad tjänst där projekt kan sparas, delas och synkroniseras. Förutom en cloud backup av projektet erbjuder den kollaborativa delen av Unity Teams en komplett projekthistorik med ändringar som gjorts genom projektets gång. När ändringar är gjorda i såväl script som objektbaserade inställningar laddas den nya version av projektet upp för andra att ladda ner och synkronisera med sina versioner.

2.5.2 Viusal Studio

Visual Studio[VS] är en IDE-(integrated development enviroment)mjukvara för utveckling av datorprogram, hemsidor, internetappar, internetjänster och mobilappar[11]. VS har stöd för flera olika programmeringsspråk, bland annat C# som används i Unity och alltså i detta projekt.

2.5.3 MTM-tracker/Assembly Scout

Assemby Scout är en programvara skapad av soft2tech GmbH i produktfamiljen Nexonar. Programmet kan med hjälp av IR kameran läsa av IR-trackers rörelse i rummet i realtid. För att ge instruktioner till användaren om hur modulen ska förflyttas i rummet använder programmet sig av steg för steg-instruktioner kallade recept. Med hjälp av förutbestämda positioner kan användaren då arbeta enligt en instruktionslista och gå vidare i receptens olika steg genom att nå bestämda positioner.

2.5.3.1 Workspace

Ett workspace är den yta där arbetsuppgifterna sker inom. Inuti ett workspace skapas paneler som fungerar som ett paket av koordinater. Alla koordinater ställs in med dess panel som referens, detta gör att om en panel flyttas räcker det att panelen omkalibreras för att dom nya positionerna ska bli uppdaterade. Positioner läggs till med hjälp av en IR-tracker genom att hålla dess spets mot önskad punkt och trycka Teach Position. Då koordinatens exakta position i praktiken är oändligt liten skapas en sfär runt positionen, denna sfärs radie kan ställas in för att öka eller minska hur exakt trackern måste nå koordinaterna.



Figur 2.4: Blå ruta: Workspace. Röd ruta: Panel. Grön cirkel: Position

2.5.3.2 Receptie

Ett Receptie innehåller arbetsinstruktioner, kontrollerar att arbetsprocessen blir utförd och definierar vad som ska hända när den blir utförd. Ett recept innehåller Steps som är steg i arbetet, dessa Steps måste i sin tur innehålla minst ett Substep. Används flera Substeps går det att välja om dessa ska utföras i en viss ordning och om alla måste utföras eller om det räcker att bara vissa blir utförda (and- eller or-funktion). När villkoren för ett Step är utfört går receptet vidare till nästa. När ett nytt recept skapas finns det redan ett step inlagt, NIO_Step vilket är felsteget som kan aktiveras av olika anledningar. NIO steget kan inte tas bort. Bekräftelsen om att stegen är utförda går att skicka över MQTT.

Assembly Scout - Teach Recipe nexonar®

RECIPE Data

Load Recipe New Recipe Translation Calibration Forbidden Sources

Recipe name: Exjobb_projekt

Workspace: Exjobb_projekt.xml

Pause Permit always Repeat recipe Calibrate at start

STEP Data

ID	Step name	Active	#Substeps
NIO	NIO_Step	1	1
1	Plocka skruv	1	2
2	Plocka enkelkrok	1	1
3	Plocka dubbelkrok	1	1
4	Lägg skruv	1	1
5	Lägg enkelkrokar	1	1
6	Lägg dubbelkrok	1	1
7	Plocka enkelkrok	1	1
8	Fäst på position 1	1	1
9	Ta en skruv	1	1
10	Skruva fast	1	1
11	Plocka dubbelkrok	1	1
12	Fäst på position 2	1	1
13	Ta en skruv	1	1
14	Skruva fast	1	1
15	Plocka dubbelkrok	1	1
16	Fäst på position 2	1	1
17	Ta en skruv	1	1
18	Skruva fast	1	1

Step name: [] active ☐

Step text (description): [] Step picture: []

Set Picture Delete Picture

☐ deny pause ☐ critical

☐ allow calibration

Substeps condition: ☐ OR ☐ AND ☐ synchronous

Duration ms: 20

Input: English - English

Save and exit Cancel

Figur 2.5: Ett recept i MTM-tracker

2.5.4 Motion Visualizer

Motion Visualizer är en programvara skapad av soft2tech GmbH i produktfamiljen Nexonar. Detta program är utvecklat för att vara ett laborationsverktyg vid rörelseanalyser. Videor kan spelas in samtidigt som rörelsemönster från IR-trackers spåras och spelas in. Programmet stödjer export av data i CSV-form för databehandlingsprogram såsom MatLab och som en redigerbar dataström via TCP/IP. Det går även upprätta direktanslutningar för att styra programmet via CAN bus, I/O och via Nexonars egenutvecklade Motion SDK.

2.5.5 MQTT

MQTT är ett klientserver baserat kommunikationsprotokoll som bygger på en prenumerera- och publiceraprinzip. Protokollet är enkelt, lättviktigt, öppet och enkel att implementera. Detta gör protokollet idealt inom användningsområden där maskinkommunikation krävs eller vid projekt av typen Internet of Things, där små meddelandepaket skickas mellan flera enheter[8]. Användning av prenumeration och publikation av meddelanden tillåter protokollet att skicka ett meddelande till en eller flera mottagare, oberoende om dessa är uppkopplade eller inte. Meddelanden skickas med tre olika kvalitetssäkringar, vars betydelse beskrivs nedan:

- At most once, för denna inställning skickas data på snabbt och resurssnålt sätt, där förlust av meddelanden kan uppstå, då ingen verifikation från prenumeranten krävs. Detta är bra för applikationer där ny data ständigt skickas.
- At least once, för denna inställning skickas data till prenumeranten, som sedan verifierar till publiceraren att meddelandet mottagits.
- Exactly once, för denna inställning skickas data på det säkraste sättet, genom digital handskakning verifierar både prenumeranten att den mottagit ett specifikt meddelande, som publiceraren sedan bekräftar, förutsatt att rätt meddelande gått fram. Detta är särskilt bra i lösningar där data inte får dupliceras eller gå förlorad, då datautbytet är viktigt för prenumeranterna.

2.5.5.1 M2MQTT

För att använda MQTT protokollet i .NET/C# lösningar skapades M2MQTT, en MQTT-klient som arbetar med C# programspråket. Detta är en gratis och enkel lösning för utvecklare, med stöd för alla .NET plattformar och WinRT plattformar[13].

M2MQTT är ett bibliotek fyllt med klasser och funktioner som gör det möjligt att kommunicera via MQTT. Genom att föra in önskad brokers IP-adress eller host name, port, eventuella användarnamn och lösenord samt vald topic kan kommunikation upprättas.

För att starta instansen för klassen `MqttClient`, där brokers IP eller host name ska skrivas in:

```
MqttClient client = new MqttClient("172.16.4.94");
```

För att registrera mottagna meddelanden:

```
client.MqttMsgPublishReceived  
+=client_MqttMsgPublishReceived:
```

Koppla upp mot broker:

```
string clientId = Guid.NewGuid().ToString();  
client.Connect(clientId);
```


Koppla ner från broker:

```
client.Disconnect();
```

Unsubscribe:

```
client.MqttMsgUnsubscribed += client_MqttMsgUnsubscribed;
ushort msgId = client.Unsubscribe(new string[] { "/topic_1",
"/topic_2" });
```

Publicera meddelanden till vald topic, vad som ska publiceras och välja Quality of Message. Den sista parameteren som här är benämnd som false går att sätta till true för att publicera ett mottaget meddelande:

```
ushort msgId = client.Publish("topic",
Encoding.UTF8.GetBytes("MyMessageBody"),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, false);
```

Subscribe till vald topic och välja Quality of Message:

```
client.Subscribe(new string[] { "topic" }, new byte[] {
MqttMsgBase.QOS_LEVEL_AT_MOST_ONCE });
```

2.5.6 Microsoft HoloLens Emulator

HoloLens Emulator är en programvara skapad av Microsoft. Mjukvaran tillåter utvecklaren att testa applikationer som är byggda för att köras med den fysiska HoloLensen, direkt på PC. Detta bidrar till en smidigare och snabbare testning av produkten. De sensorer som läser av rörelsen på användaren simuleras istället från användarens tangentbord, mus eller Xbox-handkontroll.

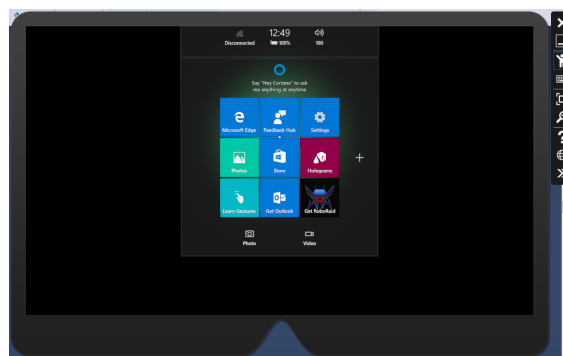


Figure 2.6: Hololens Emulator

2.5.7 Vuforia Augmented Reality SDK

Vuforia är ett av de mest avancerade verktygen inom området för datorseende för AR hårdvara. Utvecklat av Qualcomm, som sedan såldes till PTC i november 2015[15]. Verktöget är gratis för privat utveckling, och enkelt för användare att lära sig, vilket har skapat en stor community kring den. I dagsläget tillhandahåller Vuforia modell- och bildigenkänning för utvecklare. Genom att importera bilder, eller 3D-modeller i en applikation, kan användaren projicera och manipulera hologram. Detta realiserar genom att utveckla en applikation i Unity och använda funktionen ARCamera som ingår i utvecklingsverktöget.

2.5.7.1 ImageTarget

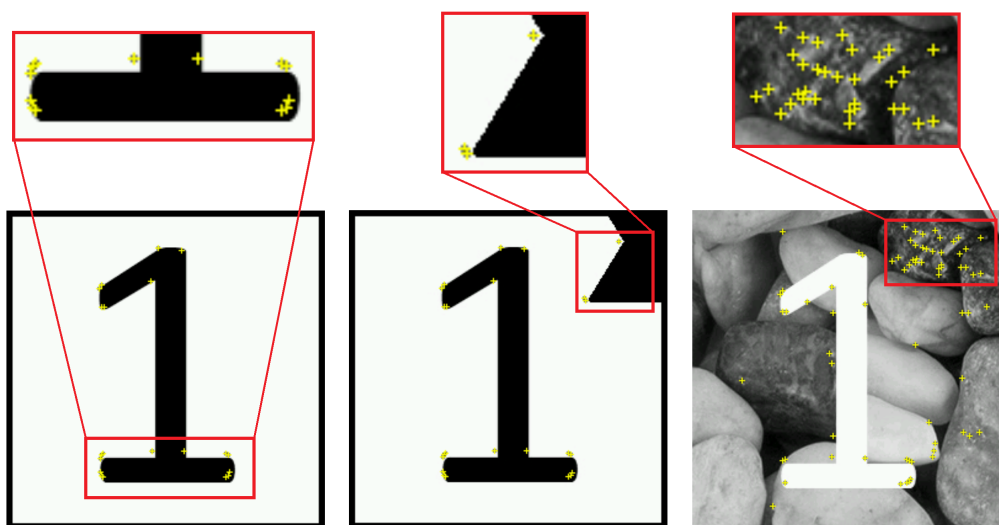
ImageTarget är Vuforias mest populära verktyg. Funktionen tillåter applikationer att känna igen bilder med hjälp av HoloLens kamera. Vid igenkänning kan ett eller flera hologram visualiseras för användaren på olika positioner, oavsett från vilken position användaren tittar från. ImageTarget bilden fungerar som ett ankare i den virtuella världen, där koordinaterna för placeringen av ett holografiskt objekt bestäms i Unity.

2.5.7.2 Bildigenkänning

Bildigenkänningsfunktionen som Vuforia tillämpar är gjord genom att analysera bilder för att sedan hitta kontraster att skapa kännetecken med. Kvaliteten på dessa kännetecken varierar beroende på följande variabler[19].

- Detaljrikedom och dess utspridning
- Bra kontrast
- Hög bredd av gråskala
- Huruvida det finns repetitiva mönster eller inte

För att säkerställa att bilden blir lättigenkänd ska dessa kännetecken vara många och utspridda över hela bilden. När kameran tittar på en bild kommer den ständigt försöka urskilja och hitta dessa kännetecken. Dessa kommer jämföras med de kännetecken som definierats av Vuforias algoritmer. När kameran hittat de som stämmer överens med de fördefinierade, har bildigenkänning uppnåtts. Hur bra kameran känner igen dessa punkter varierar mycket med ljusförhållanden och avståndet mellan objektet och kameran. Avståndet och ljusförhållanden påverkar hur stor andel av kännetecknen kameran kan uppfatta. En bildigenkänningsalgoritm har en procentsats på hur många kännetecken som behöver existera för att lyckas.



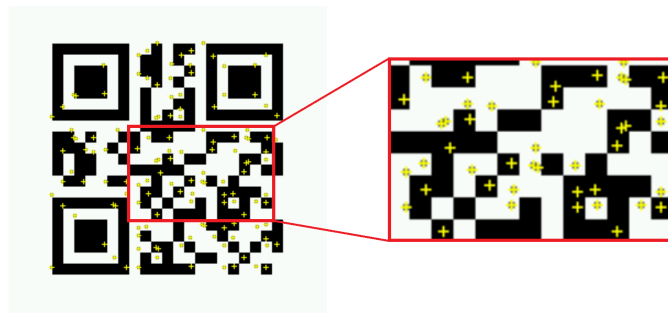
Figur 2.7: Jämförelse av olika bilder. Bild 1 till 3, från vänster till höger

I figur 2.7 illustreras tre olika ettor som används i försök till att skapa en bra bild för igenkänning. I dessa bilder kan små gula markeringar ses, dessa är bildens kännetecken.

Bild 1 innehåller en vanlig etta. Ur den genererades endast ett fåtal kännetecken, detta då bilden endast innehåller några enstaka kanter. Detaljrikedomen är väldigt låg och endast de kanter som ettan utgör genererar kännetecken. Kontrastskillnaden är optimal då bilden enbart består av två olika färger, svart och vit. Bilden innehåller inte en bred varians av gråskala, och därmed genereras inte heller en utspridd detaljrikedom.

Bild 2 är näst intill identisk med bild 1. Skillnaden är additionen av en figur med två kanter i det övre högra hörnet. Den geometriska formen genererar ytterligare två kännetecken, vilket är bättre men inte tillräckligt. Om det istället skulle läggas till flera olika geometriska former med kanter runt hela bilden, skulle antalet kännetecken öka markant. Ett exempel på en sådan lösning är en QR kod, som diskuteras senare i detta kapitel.

Bild 3 innehåller samma etta som i de två tidigare bilderna. Bakgrunden är istället för vit, en bild på stenar. Det mänskliga ögat kan enkelt urskilja att detaljrikedomen är mer intensiv medans Vuforia inte uppfattar det på samma sätt. Gråskalan i bilden är hög, och flera nivåer av grått existerar. Kontrasten mellan de olika grå är däremot inte tydlig, vilket gör det svårt för algoritmen att urskilja hörn och kanter i bilden. Detaljrikedomen är högre här, trots detta inte optimal. Majoriteten av kännetecknen är lokaliserade i det övre högra hörnet, vilket gör de kännetecknen i det nedre hörnet mer vitala för en lyckad igenkänning.



Figur 2.8: QR-kod

Figur 2.8 är en lösning som fungerar optimalt för bildigenkänning. En analys av denna QR-kod visar en hög detaljrikedom med många kanter och optimalt kontrast då bilden endast innehåller svart och vitt. Kännetecknen är tätt utspridda över hela bilden, vilket underlättar för algoritmen att identifiera bilden.

3

Metod

3.1 Arbetsmetod

Projektet kommer att utföras med hjälp av den agila arbetsmetoden Scrum och detta genom online-plattformen Trello. En gång i veckan kommer ett utvärderingsmöte hållas där framsteg och problem diskuteras. För att spara ner och dela programkod kommer Unity Collaborative användas, detta för att alltid ha tillgång till att dela och ladda ner den senaste koden för att kunna arbeta vart projektgruppen än befinner sig.

3.2 Tidsplanering

Innan det praktiska arbetet påbörjas ska så många olika arbetsmoment som möjligt identifieras för att skapa ett gantt-schema. Detta för att enkelt kunna följa upp hur arbetet ligger till tidsmässigt genom projektets gång. Genom att överskådligt läsa dokumentation för dom olika huvuddelarna, HoloLens, Nexonar-kameran och kommunikationsprotokoll kommer en uppfattning om tidsåtgången bildas för att sedan konsultera handledare på Chalmers och Virtual Manufacturing.

3.3 Informationssökning

För att hitta den information som krävs kommer det först undersökas vilken information Virtual Manufacturing har att erbjuda. Därefter genomsöks tillverkarnas hemsidor samt olika diskussionsforum på internet. Den mer allmänna C# programmeringen hittas i lämpliga böcker samt databaser online. Denna information kommer sedan användas för att definiera olika sprinter som ska utföras, samt vilka utvecklingsverktyg som krävs för utföra dom.

3.4 Verktyg

För att utveckla applikationen kommer Unity att användas. Här kommer allt visuellt skapas för HL så som hologram och texttrutor, där Visual Studio används som programmeringsmiljö för scripten, mer specifikt i språket C#. VS kommer även användas till att ladda över kod till HL-emulatorn och till den fysiska HoloLens. För att integrera MQTT-protokollet i C# koden kommer M2MQTT användas. Nexonars

MTM-tracker kommer användas för att skapa arbetsytan och arbetsinstruktionerna där Nexonar-kameran kommer övervaka.

3.5 Laboration

Laborationer är en viktig del i processen för att skapa förståelse och för att skaffa dom grundkunskaper som krävs för att genomföra projektet på ett smidigt sätt.

3.5.1 Hololens och MQTT

För att lära känna HoloLens finns det två saker som ska göras, att använda HL med befintliga appar och att arbeta i Unity där all utveckling kommer ske. Unity-laboration är den del som kommer ta längre tid. Där läggs grunderna i hur skapandet fungerar i Unity, inte nödvändigtvis åt Hololens utan till spel eller annan 3D-miljö. HL ska i slutet av projektet kommunicera med Nexonar via MQTT, därför krävs att detta protokollet undersöks. En applikationen ska skapas och utformas på det sättet att ett script i Unity manipuleras med hjälp av meddelanden skickade över MQTT.

3.5.2 Sammankoppling av Hololens och Nexonar

När en grundlig förståelse för hur varje teknik fungerar är det dags att koppla samman dessa. Att skapa en arbetsinstruktion i MTM-tracker som skapar data, skicka denna via MQTT och sedan få HL att ta emot. Denna laborationen är ett viktigt och stort steg mot det slutgiltiga mål.

3.6 Utveckling av produkten

När tillräckligt med kunskap är insamlad för att få alla funktioner att fungera kommer fokusen ligga på att bygga den produkt som projektet handlar om, upprätta en god kommunikation mellan HL och Nexonar samt skapa mjukvara som visar detta. Detta moment kommer att innebära arbete i Unity Collaborate.

4

Genomförnade

Detta kapitel handlar om hur projektgruppen gick till väga för att skapa den applikation i HoloLens som kommunicerar med Nexonar-tekniken och dess recept. Här beskrivs också hur detta program fungerar. För att inte skapa förvirring är många engelska ord ej översatta.

4.1 Förstudie och utveckling av arbetsmoment

För att bygga en bra och funktionell applikation bestämdes ett fiktivt arbetsmoment som tekniken kan användas för. Genom att diskutera möjliga fall utefter Virtual Manufacturing kundbas föll valet på en montering åt exempelvis Volvo. Resultatet blev ett arbetsmoment med en plockstation för delar och en arbetsstation för montering. Ett moment som kan ses som en simulering av att plocka och montera ihop komponenter till en bil, då exempelvis en motor.

Det arbetsmoment som valdes börjar vid plockstationen. Där ska användaren plocka dom artiklar som behövs, i detta fall tre skruvar, två dubbelkrokar och en enkelkrok. När varje steg i plocksekvensen är godkända, avlastas dessa i lådor i direkt anslutning till monteringsytan för en smidig access. Nu påbörjas monteringen genom att användaren blir ombedd att fästa första kroken på den första positionen, för att därefter plocka en skruv och skruva fast den. Detta upprepas sedan för två andra positioner på monteringsytan.

4.2 MQTT

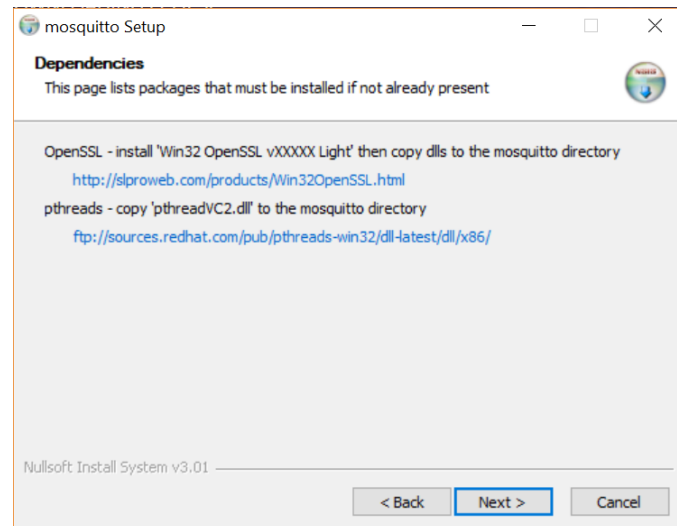
I detta kapitel återfinns viktiga moment direkt kopplat till projektets huvudsyfte, att koppla samman Nexonar-kameran med HL via MQTT.

4.2.1 Upprättande av MQTT-broker

Det första steget för att koppla samman två eneter med MQTT är att en broker behövs, alltså en dator eller server som agerar som förmedlare. Det finns online brokers att tillgå, dock är vår applikation tänkt att användas på slutna industriella nätverk utan internetuppkoppling. Därför behövdes en broker som arbetar på ett lokalt nätverk. Denna broker behöver finnas på nätverket eller till exempel på den dator som Assembly scout körs på, vi valde det senare alternativet då den datorn

alltid kommer vara igång när applikationen används.

Vi valde att använda oss av Mosquitto broker men det är även möjligt att använda andra MQTT brokers[17]. Det första vi gjorde var att ladda ner klienten från Mosquittos hemsida[18], vid exekvering av installationsfilen hänvisar utvecklarna till länkar på hemsidor där två systembibliotek ska laddas ner. Se Figur 4.1.



Figur 4.1: Mosquitto installation

För att starta mosquitto-brotern måste mosquitto.exe exekveras. För att verifiera att mosquitto brotern är igång, användes kommandot `netstat -an` i kommandotolken. Kommandot visar datorns nätverksstatistik, här skall en lokal adress vid 0.0.0.0.1883 finnas, där 1883 är standardporten som används för MQTT-protokollet. För att ta reda på den publika broker-adressen använde vi oss av datorns IPv4-adress. Denna hittas enkelt via kommandotolken med kommandot `ipconfig`.

IP-adressen är broterns adress och används för att kunna publicera och prenumerera till olika topics i ett lokalt nätverk.

Mosquittio installerades sedan och de nedladdade biblioteken extraherades. Filer-na placeras in i mosquitto applikationens programmap. De efterfrågade systemfiler som krävs hittas i samma mapp, med namnet `readme-windows.txt`.

4.2.2 MQTT kompatibilitet Nexonar

Då Assembly scout är programvaran som huvudsakligen använts av Virtual Manufacturing för applikationer med Nexonar valde vi att fokusera på just denna mjukvara, Assembly scout har också direkt stöd för MQTT. En upptäckt långt senare i projektet visade dock att syster-programvaran Motion Visualizer skulle kunna vara en bättre lämpad programvara för ändamålet. Detta för att Assembly scout enbart kan skicka godkännanden av genomförda receptsteg (med tillhörande positionsnamn), medans Motion Visualizer kan skicka 3D-koordinater av IR-trackern i realtid. Mer information om detta går att hitta i Vidareutveckling kapitel 8.

4.2.2.1 MQTT med Assembly scout

Assembly scout är skapat med direkt stöd för MQTT, något som först vilseledde oss. När vi började laborera med tekniken försökte vi koppla upp till publika brokers som hittades online utan någon större framgång. Det vi senare insåg är att det krävs att Mosquitto är installerat på datorn för att den ska kunna kommunicera via MQTT, även om datorn inte ska användas som broker. Hur installationen gick till är beskrivet i föregående kapitel, 4.2.1.

För att ställa in MQTT i Assembly scout hittades inställningarna under Admin logon - Device configuration - MQTT configuration. Där visas följande inställningar, se figur 4.2.

Assembly Scout - MQTT Configuration nex)))onar®

MQTT Settings

Use MQTT: ☒

MQTT Host Name: MQTT Host Port:

MQTT Topic Name:

MQTT ClientID:

Limit Message Frequency: ☒ Minimum Message Interval (ms):

Channel Content: Channels ☒ Source ☒ Position ☒ Trim Position after

[Back](#)

Figur 4.2: MQTT inställningar i MTM-tracker

Channel Content-rutan är inte redigerbar i applikationen, detta var förvirrande. Det ingav tankar om att detta var felkällan till att inget skickades via MQTT. Insikten om att Mosquitto behövdes installeras för att detta skulle gå löste problemet.

Beskrivning av inställningar:

- Use MQTT: För att aktivera funktionen.
- MQTT Host Name: Datorns lokala IP-adress ställdes här in då datorn ska agera broker.
- MQTT Host Port: 1883 är standard-porten för MQTT, det går även bra att använda andra portar.
- MQTT Topic Name: Exjobb är namnet på den topic vi valde för projektet. Valfritt namn går att använda, så länge alla maskiner som ska kopplas upp använder samma.
- MQTT ClientID: Namnet Exjobb valdes även här. Detta fält är inget måste då en maskin inte behöver ha något specifikt namn utöver sin IP-adress i ett MQTT nätverk. Detta används bara för att förtydliga.
- Limit Message Frequency: Används ej.
- Maximum Message Interval(ms): Används ej.
- Channel Content: Denna dialogruta är endast för att visa vad som skickas, den är alltså inte redigerbar.

4.3 IR-kameran, Nexonar

4.3.1 Utformande av arbetsstationen

Arbetet som applikationen ger instruktioner för utövas på en arbetsstation. Denna innehåller en ställning med lådor som agerar som plockstation, en avlastningsyta för det plockade materialet och slutligen en vertikal arbetsyta där arbetet utförs. Se figur 4.3.



Figur 4.3: Plockstationen till vänster och monteringsstationen till höger

IR-kameran behöver fri sikt över hela arbetsytan och till viss del skydd mot dagsljus då solens IR-strålning kan störa. Därför placerades arbetsytan ungefär 5 meter in i lokalen för skydd mot direkt solljus och kameran fästes på en hög hylla precis intill ytan, snett vinklad in för optimal översikt. För att få till den mest optimala vinkeln för kameran utvärderade vi genom att kolla i Assembly scout vart kameran lyckades se IR-trackern. Här framhövdes IR-kamerans pyramidliknande synfält och vinkeln ställdes in med hänsyn till detta.

Två stycken IR-trackers används i arbetet. En som sitter fast på skruvdragaren för att lokalisera skruvspetsens position och en som sitter på handleden av användaren för att övervaka handens position.

4.3.2 Utveckling i MTM-tracker

4.3.2.1 Inställning av IR-trackers

Det finns två olika storlekar av trackers, den mindre är smidigare och är därför avsedd att bäras på handleden för att övervaka användarens hand. Den initiala planen var därför att använda en mindre tracker på handen. Det negativa med den mindre är dock att kameran har lite svårare att se den i snedställda vinklar, framförallt i det yttre spektrat av dess synfält. Därför valdes den större varianten även på handen, då den är mer stabil.

Inställning av dessa IR-trackers gjordes via ‘Device Configuration’ i MTM-tracker. Väl där valdes ‘Hard Probe Manager’. Här inne går det att se alla tillgängliga trackers, redigera dom, ta bort och kalibrera. Här lades två trackers till genom att trycka på Add, välja Hard probe Lite då några djupare inställningar inte behövdes och Tip Diameter ställdes in. Tip Diameter är spetsens diameter, alltså i skruvdragarens fall en uppskattad diameter av skruvbitsens huvud, i handtrackerns fall en uppskattad diameter av fingrarna.

När dessa inställningar är gjorda behövs en kalibrering för att MTM-tracker ska veta exakt vart varje trackerns spets är. Det som räknas fram är en offset från trackerns position. Detta gjordes genom att trycka på Calibrate, då öppnas fönstret Hard Probe Calibration. För att mjukvaran ska räkna ut spetsens offset hålls spetsen mot en fast position. Efter att ha tryckt Start Hardware och Start Calibration ska trackern lutats runt om i olika positioner med spetsen helt stilla på samma fasta situation. Det rekommenderas att 10-20 punkter över 20 graders vinkel fångas genom att trycka capture point. När detta är gjort genomförs en liknande process genom att trycka på Start Optimizing för att finjustera. Denna gång förs trackern runt i cirklar, fortfarande med spetsen på en fast punkt, för att fånga 200 olika positioner.

4.3.2.2 Skapande av workspace

Två paneler skapades, en för monteringsytan och en för plockstationen. Detta medför att plockstationen enkelt kan flyttas för att sedan enkelt kalibreras in med dess positioner intakta genom Teach Workspace funktionen.

Från huvudfönstret valdes Teach Workspace för att sedan välja New för att skapa en helt ny. Här valdes först tre referenspunkter för respektive yta, monteringsytan och plockstationen, och dessa tre punkter märktes ut fysiskt för att inte glömmas bort. När referenspunkterna för panelerna var satta, ställdes samtliga positioner in för vardera panel.

Tabell 4.1: Monteringsytans positioner:

<i>2_sBox</i>	Låda för skruvar
<i>2_ekBox</i>	Låda för enkelkrokar
<i>2_dkBox</i>	Låda för dubbelkrokar
<i>2_ekPos_1</i>	Enkelkrokens monteringsposition
<i>2_dkPos_1</i>	Dubbelkrok 1 monteringsposition
<i>2_dkPos_2</i>	Dubbelkrok 2 monteringsposition

Tabell 4.2: Plockstationens positioner:

<i>1_sBox</i>	Låda för skruvar
<i>1_ekBox</i>	Låda för enkelkrokar
<i>1_dkPos</i>	Låda för dubbelkrokar

4.3.2.3 Skapande av recept

Receptet är uppbyggt av de positioner som beskrivs i föregående kapitel och med hjälp av dessa skapas listan av punkter som ska genomföras i den arbetsbeskrivning som togs fram i förstudien, kapitel 4.1.

För att skapa receptet användes Teach Recipe för att sedan välja New Recipe, receptet fick namnet Exjobb_projekt. Därefter valdes dom två workspace som skapats i föregående kapitel 4.3.2.2. När ett nytt steg läggs till är första steget att skriva in den information som ska visas under steget och om det ska visas någon speciell bild. All information som ges till användaren i detta projektet ges dock av HL, därför skrevs bara kortare texter in här för att hjälpa till vid utveckling.

På varje steg skapades ett Substep där den aktuella positionen lades in. Alternativet Position valdes för att godkännandet ska komma vid uppnådd position. Den valda positionen ställdes in under Source samt vilken Device Channel, alltså om det skulle vara skruvdragaren eller handen som nådde positionen.

The screenshot displays the 'Assembly Scout - Teach Substeps' software interface. At the top, the title bar reads 'Assembly Scout - Teach Substeps' on the left and 'nex))onar' on the right. The main window is divided into several sections:

- SUBSTEP Data Table:** A table with two columns: 'Substep Name' and 'Type'. It lists 'Substep 1' with type 'Position (1)' and 'Substep 2' with type 'Position (1)'. Below the table are 'Add' and 'Delete' buttons.
- Substep Form:** A form for editing a substep. It includes:
 - Substep name:** A text field containing 'Substep 1'.
 - Dead Time after Last [ms]:** A numeric input field set to '0'.
 - Substep active:** A checked checkbox.
 - Substep text:** A large text area for entering instructions.
 - Substep picture:** A placeholder image area with 'Select Picture' and 'Delete Picture' buttons.
- Configuration Panels:**
 - Position:** Includes a 'Source' dropdown set to '1_sBox', a 'Motion Device channel' dropdown set to '2', and a 'Mode' section with radio buttons for 'Valid when source is touched' (selected), 'No validation. Permanently send source is "touched" or "not touched" to Digital OUT channel 1...16', and 'and signal "touched" as High/Low'.
 - Sound:** Includes a 'Sound file' dropdown, 'Select' and 'Play' buttons, and a 'Play asynchronous' checkbox.
 - Digital IN:** Includes a 'Digital channel' dropdown set to '1', a 'Value' input set to '1...16', and a 'Mode' section with radio buttons for 'Substep is valid when value is set' (selected) and 'Substep aborts step on wrong value'.
 - Digital OUT:** Includes a 'Digital channel' dropdown set to '1', a 'Value' input set to '1...16', and a 'Mode' section with radio buttons for 'High/Low'.

At the bottom of the interface, there is a 'Step name' field containing 'Plocka skruv' and two buttons: 'Apply changes' and 'Cancel'.

Figur 4.4: Substep

4.3.3 Motion Visualizer och 3D-koordinater i realtid

Sent in i projektet upptäcktes programmet Motion Visualizer[MV], ett program som med hjälp av Nexonar-kameran kan läsa av och visualisera 3D-koordinater av en IR-tracker i realtid. Ursprungsidén var att använda sig av denna data som ett komplement till HoloLens. Mer om detta går att läsa i vidareutveckling, kapitel 8.

MV distribuerade rörelsedata via en TCP utgång under den valda porten i det lokala nätverket. Den data som skickades var i form av x-, y- och z-koordinater. En mottagare för dessa TCP-paket utvecklades i C och Visual Studio[VS] med hjälp av Oboolnet[22], ett TCP bibliotek som avser att enkelt tillämpa TCP-nätverkning. I samma projekt byggdes en MQTT klient med hjälp av M2MQTT, vars syfte var att kunna publicera datan från MV till en mottagare.

Mottagaren som utvecklades blev en väldigt simpel applikation i Unity vars syfte är att ta emot data från TCP servern och översätta koordinat-data och uppdatera ett objekts position i programmet. Därav skulle en IR-trackers position i verkligheten projiceras av ett objekt i Unity. Programmet använde sig av en `Regex.Split` funktion för att sortera en sträng in i ett fält. Fältet används sedan av en funktion som placerar dessa koordinater i tre olika variabler för att därefter uppdatera ett objekts 3D-positioner i Unity.

Applikationen lyckades ta emot dessa koordinater utan större problem. Frekvensen av positioner som MV skickade till VS blev dock för hög. När datan skulle skickas vidare via MQTT klarade scriptet inte av att förmedla datan tillräckligt snabbt. Detta resulterade i att mer data från MV mottogs än vad MQTT klarade av att skicka vidare. Som ett resultat av detta bildades en kö av positioner inuti mottagningsscriptet. När positionerna sedan skickades via MQTT innehöll varje meddelande för mycket data, vilket Unity-applikationen inte kunde läsa av.

Försök gjordes att försöka minska dataströmmen från MV, dock utan någon större framgång. Slutsatsen blev att koden behöver optimeras samt att dataströmmen behöver strypas på ett effektivt sätt i VS applikationen. Då tidsbrist, och bristande kunskaper i C# var ett faktum valde vi att inte fokusera på att fortsätta utveckla denna möjlighet.

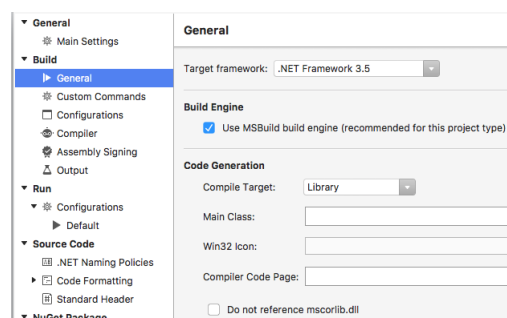
4.4 Unity

I detta kapitel behandlas de inställningar i Unity som krävdes för att skapa en väl fungerande applikation.

4.4.1 Systemfiler för MQTT och Unity

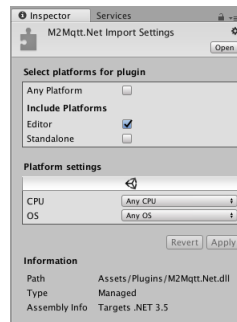
För att kunna använda MQTT i Unity används ett tillvägagångssätt där funktioner läggs in som assets, för att göra detta ska två .DLL-filer skapas. För att skapa dessa .DLL-filer användes filer från GitHub som heter paho.mqtt.m2mqtt[27] och som är fritt fram att använda.

Filerna från GitHub laddades ner och därefter öppnades M2Mqtt.sln i Visual Studio. Innan skapandet av .DLL-filerna påbörjas krävs det att rätt Framework ställs in för M2Mqtt.Net projektet. Detta görs genom att högerklicka på M2Mqtt.Net och välja Options. Under fliken Build och General väljs .NET framework 3.5 under fliken Target Framework. Projektet sparades ned och kompilerades sedan, därifrån genererades en M2Mqtt.Net.dll fil. Den andra systemfilen skapades genom att kompilera M2Mqtt.WinRT projektet. Båda filerna hittas sedan i debugmappen. När projekten byggdes kom diverse varningar upp men dessa förbisågs då det inte skulle komma att påverka något.



Figur 4.5: .NET framework 3.5

För att lägga in filerna i Unity skapades en mapp som hette Plugins. I den mappen lades sedan M2Mqtt.dll in. Från Inspector-fältet är det viktigt endast kryssa i Editor under Include Platforms för denna plugin, se figur 4.6. På liknande sätt skapas en ny mapp vid namn WSA under Plugins, i denna mapp läggs M2Mqtt.dll in. I Inspector fönstret kryssas sedan endast WSAPlayer i.



Figur 4.6: Inspector fönstret

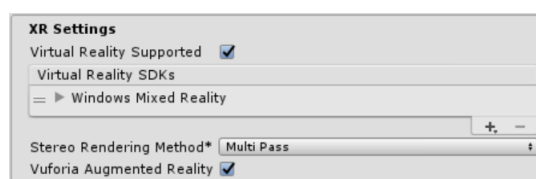
4.4.2 Implementering av Vuforia SDK

Vi valde att använda Vuforia ImageTarget funktionen i syfte att visualisera för användaren vart den ska interagera. Ett objekt ska tändas och släckas över en bild på arbetsstationen. Detta ska vägleda användaren i sin process.

4.4.2.1 ARCamera

Vuforia ARCamera är grundstenen i utvecklingen av AR applikationer som utnyttjar bildigenkännande egenskaper. Den kan ses som en modifierad digital kameralins som tillåter applikationen att se och upptäcka olika objekt som Hololens ser. Vuforia kommer inbyggt i Unity, och funktionerna är tillgängliga som objekt. För att kunna använda Vuforia SDK krävdes att standard kameran Main Camera i scenen byttes ut till Vuforias ARCamera.

Vuforia aktiverades genom att gå in i File/Build Settings/Player Settings/XR Settings. Under XR Settings behövde rutan Vuforia Augmented Reality kryssas i. Se figur 4.8.



Figur 4.7: Inställningar för Vuforia

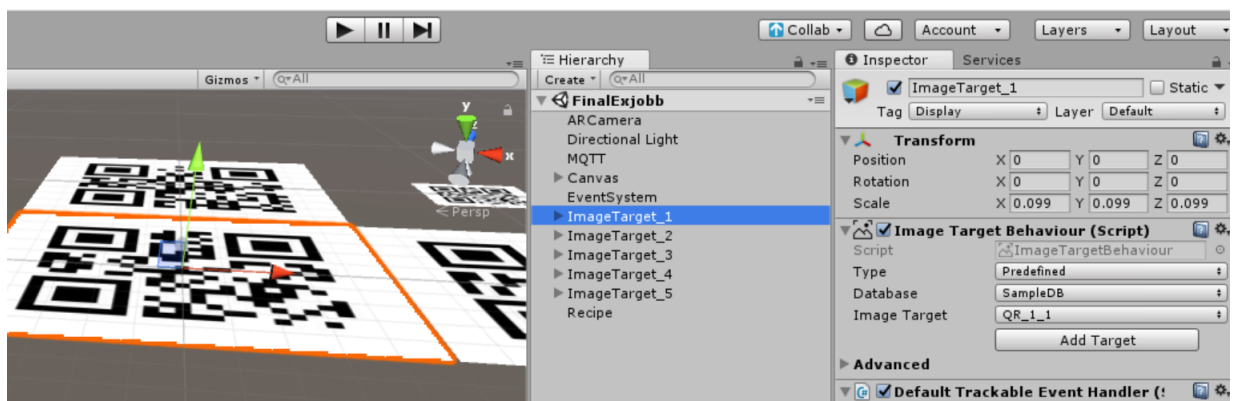
Då Vuforia är utvecklat av ett annat företag än Unity, krävdes det att applikation registrerades i Vuforias utvecklarportal. Därifrån hämtades en licensnyckel under License Manager. Nyckeln måste genereras varje gång ett nytt projekt ska påbörjas i Unity där Vuforia är tänkt att användas.

För att nyckeln skulle fungera krävde Vuforia att Unity-projektet och nyckelregistreringen hade samma namn. Nyckeln klistrades sedan till Unity projektet, under ARCamera och VuforiaConfiguration. Med installationen klar, var Unity projektet färdigkonfigurerat för att ta vara på Vuforias funktioner.

4.4.2.2 ImageTarget

Fem olika bilder togs fram för mjukvaran att känna igen. Dessa skulle sedan laddas upp i Vuforias utvecklarportal under Target Manager. Här skapades en ny databas genom att trycka på Add Database, där bilderna sedan laddades upp. Det var väldigt viktigt att den utskrivna bilden som skulle placeras på arbetsstationen mättes upp, och skrevs sedan in i enheten meter i portalen. Om detta inte gjordes riskerades instabilitet i projiceringen av hologramet. När de önskade bilder var uppladdade, markerades de bilder som skulle användas i projektet och laddades sedan ner som ett Unity-paket genom att trycka på Download Database.

Unity-paketet sparades sedan ner och importerades genom att klicka på Assets/Import Package/Custom Package. Ett nytt objekt under GameObject-fliken i Unity skapades, och därifrån kunde en ImageTarget-lösning upprättas, genom att välja Vuforia/Image.



Figur 4.8: Inställningar för vuforia

I figur 4.8 ses objekten under Hierarchy-fönstret i mitten, till vänster om den ses en del av scenen som arbetades med, samt den QR-kod som valdes för ImageTarget-lösningen. Längst till höger ses Inspector-fönstret. Under Image Target Behaviour hade den tidigare skapade databasen SampleDB och den uppladdade bilden QR_1_1 valts för just detta ImageTarget objektet.

För den bildigenkännande hanteringen i Hololens användes Extended Tracking, vilket hittades under Advanced under Image Target Behaviour. Extended tracking betyder att Hololens sensor kameror kommer hålla koll på ImageTarget-bilden, trots att den är utanför Hololens skärm. Detta medför i en bättre upplevelse för användaren.

4.4.2.3 Val av ImageTargets

I de tidigare försöken användes ImageTargets med en diameter på cirka 3,5 cm. Avsikten var att använda siffror som bilder. HL hade dock problem att känna igen dessa sifferbilder. Detta problem identifierades som en brist av kännetecken för Vuforia kameran, ett ämne som diskuteras mer under 2.5.7.2. Slutsatsen ur denna studie var att använda QR-koder istället, av samma storlek på 3,5 cm i diameter. Bytet till QR-kod resulterade i en mer stabil bildigenkänning där HL mycket enklare upptäckte de olika bilderna. Efter några tester upptäcktes att HL-kameran tenderade att blanda ihop de olika QR-koderna, vilket ledde till att ett hologram kunde tändas ovanför fel QR-kod. Stabiliteten av hologrammen var också bristande, där dessa tenderade att byta position tygellöst, samt vibrera vilket gjorde de suddiga och jobbiga för användaren att titta på.

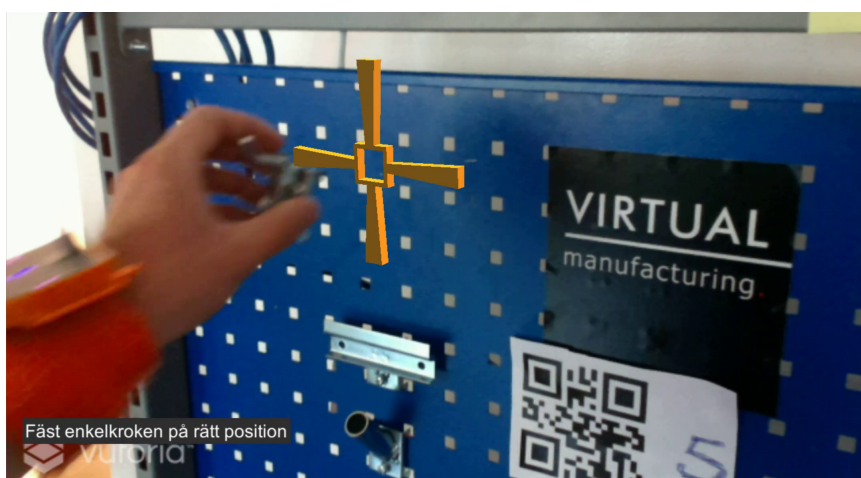
Efter tester drogs slutsatsen att de små QR-koderna skapade positionsrelaterade konflikter för Vuforia och HL. Det beslutades att skapa större QR-koder, då detta enligt teorin förenklade bildigenkänningen. Diametern ökades från 3,5 cm till 9.9cm. Storleksändringen resulterade i en markant stabilisering av bildigenkännandet, där hologram alltid tändes över rätt ImageTarget.

Inställningarna Allow MSAA och Occlusion Culling under ARCamera inaktiverades. Detta löste problemen där hologramen vibrerade och bytte positioner.

5

Resultat

En AR-applikation för Microsoft HoloLens har utvecklats som med hjälp av Nexonars IR-kamera instruerar en användare under ett arbetsförlopp, från att plocka nödvändiga artiklar till att dessa monteras i rätt ordning. Se figur 5.1.



Figur 5.1: Ett kors för anvisning av monteringsposition

Text och hologram hjälper användaren att plocka rätt artikel eller montera en artikel på rätt plats. Med ett positionsgodkännande av Nexonar, skickas ett meddelande via MQTT och applikationen stegar vidare med nya instruktioner. Vid detta utförandet, att använda MTM-tracker som mjukvara för Nexonar, fanns inte möjligheten att skapa nya arbetsinstruktioner via HoloLens då MTM-tracker enbart kan skicka information och inte kan ta emot och behandla. Därför lades ingen tid på att utvärdera hur detta hade kunnat fungera.

Texten visas i nedre vänstra hörnet och är kort men koncis för att ge en tydlig instruktion. Texten följer med användarens synfält för att det ska vara möjligt att läsa och utföra uppgiften samtidigt. För att visa användaren vilken position som är aktuell i varje steg projiceras hologram vid eller över denna position. Det hologram som används är en pil för hänvisning av att plocka eller placera artiklar i rätt låda. Ett kors används för att visa vart artiklar ska monteras. En konfirmation i form av en tumme upp visas i nedre högra hörnet av synfältet vid genomförd uppgift. Detta för att ge användaren en snabb och positiv feedback vid avslutad uppgift.



Figur 5.2: Plockandet av artiklar

Dessa tre olika former av assistans finns i tre olika sammansättningar, detta för att användaren själv ska få välja vilken mängd av information denne behöver. Nivåerna består av:

- Nivå 1: Text, hologram och konfirmering av genomfört steg.
- Nivå 2: Hologram och konfirmering av genomfört steg.
- Nivå 3: Konfirmering av genomfört steg.

6

Diskussion och slutsats

Detta projekt visar att det är fullt möjligt för Microsoft HoloLens och Nexonars IR-kamera att arbeta tillsammans på ett bra sätt. Det visar även att MQTT-protokollet är väl fungerande teknik i detta samband.

IR-kameran används via Nexonars mjukvara MTM-tracker som kan skicka godkännande av genomförda steg i arbetsprocessen. Mjukvaran gör detta felfritt och med hög precision. Men, slutsatsen blir ändå att programvaran Motion Visualizer är ett bättre alternativ för denna typ av applikation. Detta för att MTM-tracker enbart kan leverera godkännanden i recept och inte aktuell positionering vilket MV kan. Mer om detta går att läsa i kapitel 8.

AR-tekniken har överlag stora möjligheter att bidra till en bättre och effektivare arbetsprocess inom dom flesta arbetsområden, däribland verkstadsindustrin. Att kunna instruera en operatör på ett effektivt sätt förkortar upplärningsprocessen avsevärt och blir mindre betungande för ett företag, det gör det också enklare för operatören att bredda sina kunskaper och utföra arbeten den inte redan kan. Något som påvisats under detta arbete är dock att tekniken inte riktigt är där den borde vara för att AR-inläring ska ta fart. En utvärdering av HL's prestanda visar på att det finns utrymme för förbättring och utveckling. HL har ett snävt synfält vilket hämmar upplevelsen och känslan en hel del då det är fullt möjligt att låta blicken hamna utanför detta synfält, speciellt på närmare håll. Detta leder till att synfältet kan ses mer som en transparent skärm, istället för ett naturligt komplement av en virtuell verklighet. Dess högtalare och röstigenkänning är däremot väldigt bra och det finns stora möjligheter inom dessa områden. Ett problem är också priset på hårdvaran. Om företag, och framförallt privatpersoner, ska göra sig bekväma med tekniken kräver det att tillgängligheten ökas, likt datorkunskaper ökade i takt med hem-pc reformen på slutet av 1990-talet[28].

Projektet visar HL's begränsning i synfältet väldigt tydligt för att arbetet utförs på relativt nära håll. Ursprungsvisionen var att skapa en sömlös virtuell värld för användaren att interagera i. För att sätta problemet i perspektiv, motsvarar synfältet av HL endast det en människa ser när den fokuserar på ett specifikt objekt, periferin kring det fokuserade objektet saknas helt. Det begränsade seendet drev en underliggande tankegång under projektets gång, där frågan kring hur mycket beskrivande information som faktiskt behöver visas för användaren. En produkt av problemet var att HL tvingar användaren att vrida huvudet för mycket, eller ta ett steg tillbaka under närgående monteringsarbete. Detta för att kompensera för

avsaknaden av information utanför synfältet. Dessa tilläggsmoment skapade en diskontinuitet i arbetsflödet och hämmar slutprodukten. Därav dras slutsatsen att HL för närvarande inte är redo för en fullskalig implementering vid monteringsarbete i verkstadsindustrin.

Kommunikationsverktyget MQTT användes i en mindre skala under detta projekt. Dess potential öppnar däremot upp för kommunikation mellan flera maskiner samt informationsflöde av olika ämnen. Den data som i detta fall skickades från MTM-tracker var av en förutbestämd struktur, där innehållet kunde ändras. Detta möjliggjorde skapandet av en algoritm där informationen kunde styckas upp i flera olika strängar och som sedan avkodades beroende på innehåll. Som följd av detta kunde en mer dynamisk algoritm byggas. Efter tester av MQTT-funktionen, konstaterades att meddelandet från MTM-tracker endast skickades när ett steg utförts. Alltså var informationen i strängen inte relevant i detta sammanhang. Strängen kunde istället betraktas som en indikation för algoritmen att det var dags att visa ny information. Därför byggdes en stegorienterad algoritm istället för en som agerar efter innehållet.

Arbetsmetoden Scrum har under detta projekt fungerat som det är tänkt. De olika sprintar som definierades i backloggen under projektets gång korrelerade alltid med gantt-schemat. Metoden hjälpte till att på ett strukturerat sätt hålla projektet innanför tidsramen. Utvärdering av vad som gjorts under tidigare arbetspass blev enkelt överskådligt, samtidigt som det åskådliggjorde vad som behövde göras närmast. Detta gav avsedd tid för att ventilera tankar och åsikter, vilket hjälpt till att utveckla projektet på ett effektivt sätt. Arbetet skedde parallellt med separata delmål i projektet som definierades vid tidigare backlog möten. När en uppgift var avklarad laddades en ny version av applikationen upp till Unity teams. Arbetet kunde därför fortlöpa utan större problem, backloggen hindrade projektet från att inriktas på irrelevanta delmål. Samt, om någon körde fast kunde den andre enkelt förstå detta genom att se hur mycket som gjorts i backloggen.

Slutprodukten gränssnitt och utformning skulle varit mer utvecklad än vad den blev. Den huvudsakliga förklaringen till att detta område inte fick tillräckligt med tid var den tidsbrist som uppstod när utvecklingen av detta skulle påbörjas. Projektgruppens förkunskaper inom Unity och C# var begränsade, och fokuset låg inte på gränssnitt relaterade områden under informationssökningen. I början av projektet spenderades en stor del av tiden på att lära sig och att experimentera med MQTT och Unity.

Att följa det bestämda gantt-schemat blev här till en nackdel då mer tid hade kunnat spenderas på att studera grunderna som krävs för att skapa ett snyggt gränssnitt. Därför bestämdes att gränssnittet skulle ha en lägre prioritet än projektets huvudsyfte, alltså kommunikationen och utvärdering av möjligheterna mellan HL och Nexonar. Designen av detta är väldigt grundläggande, och följer ingen specifik grafisk röd tråd. Den information som visualiseras för användaren görs på ett konkret och tydligt sätt.

6.1 Vad som hade kunnat göras annorlunda

I början av projektet lades för mycket fokus på MTM-tracker. Anledningen till detta var att denna mjukvara har ett mer uttalat stöd för just MQTT vilket syns tydligt i Nexonars kommunikation kring produkten. Om ett mer öppet sinne hade applicerats i denna fråga kanske Motion Visualizer hade upptäckts i ett tidigare skede, därmed hade applikationen kunnat byggas med detta som grund.

7

Miljö och etik

Den allmänna användningen av AR-teknik är definitivt ett steg i rätt riktning miljömässigt då det ger möjligheten att projicera saker som annars skulle vara producerat eller tryckt.

Det går att diskutera vad en applikation av detta slag har för inverkan på arbetskultur. Det finns två sidor på myntet, det ena är att det kan göra anställda mer ersättningsbara, det andra är att anställda enklare kan lära sig fler saker inom företaget på kortare tid vilket breddar deras kunskaper. I Sverige är dock risken att ersättas väldigt låg på grund av goda anställningsskydd, därför är detta inte ett allt för starkt argument.

Att för en arbetare bära ett verktyg likt HoloLens med guidning och övervakning av arbetet kan öppna upp för personlig resultatuppföljning av arbetsförmågan hos den enskilda anställda. Detta kan ge en känsla av övervakning hos den anställde vilket lätt kan upplevas negativt.

8

Vidareutveckling

Det vi tycker är den mest konkreta och viktiga vidareutvecklingen av detta projekt är att testa använda Motion Visualizer istället för MTM-tracker som mjukvara för Nexonar. MV ger möjlighet att läsa av den exakta positionen av IR-trackers och skicka dessa i realtid via TCP/MQTT. En applikation som behandlar handens koordinater i realtid skulle ge HoloLensen en mer dynamisk, och hjälpsamt roll. En algoritm som är baserad på avståndet mellan hand och punkt skulle kunna utvecklas. Genom att analysera vart handen befinner sig, kan algoritmen vägleda användaren under hela förloppet. Skulle användaren till exempel arbeta på fel arbetsyta kan algoritmen varna för detta. Även i fall där användaren inte rör hand eller verktyg på en stund, kan olika interaktioner genereras. Exempelvis kan detta betyda att användaren inte vet vad den ska utföra, och därav få bättre förklaringar.

I vår applikation har vi använt HoloLens-teknologin endast som en typ av skärm. Den information som visas för användaren är statisk, och beror redan på förbestämda koordinater i MTM-tracker.

Bytet till MV gör att applikationen inte längre är beroende av två olika mjukvaror för att skapa arbetsbeskrivningar, istället hade dessa kunnat skapas i Unity endast. En artikels utformning hade kunnat läsas in via dess CAD-ritning och kritiska positioner i form av svetspunkter, skruvförband eller dylikt hade kunnat framställas och användas i programmet. Skalbarheten blir då mycket högre och tiden för att skapa en arbetsinstruktion förkortas avsevärt.

Applikationens användargränssnitt är just nu väldigt simpelt och hade kunnat utvecklas med en meny för användaren att välja arbetsinstruktion, informationsnivå och andra relevanta inställningar. Menyn hade även kunnat innehålla en inloggning för olika användare, på så sätt kan varje användare ha sparade inställningar för sin profil.

Texten och bilden är i nuläget ankrad till HL kameran i Unity. Detta medför att den upplevs aningen skakig när användarens huvud rör sig, vilket påverkar helhetsupplevelsen negativt. I vissa situationer måste användaren stanna upp och fixera huvudet för att läsa texten. Därför anser vi att en stabiliseringsfunktion borde kunnat skapas för att minska skakningarna.

Möjlighet för användaren att öppna upp informationsrutor tillhörande artiklar eller positioner hade varit en bra egenskap. Här hade information om vart artikeln ska och varför, olika tekniska egenskaper, lagersaldon och annan relevant information visas. Detta och flera saker inom applikationen hade förslagsvis kunnat styras med röstkommandon.

Referenser

- [1] Microsoft, "Microsoft HoloLens" [Online]. Tillgänglig: <https://www.microsoft.com/sv-se/hololens>, hämtad: 2018-02-12.
- [2] Microsoft Corporation, WA, 2018. [Online]. Tillgänglig: <https://www.microsoft.com/sv-se/>, hämtad: 2018-05-22
- [3] Nexonar, "IR Single Camera - Nexonar," [Online]. Tillgänglig: <http://www.nexonar.com/de/produkte/ir-single-camera-tracking/ir-single-camera/>, hämtad: 2018-02-12.
- [4] MQTT, "MQTT," [Online]. Tillgänglig: <http://mqtt.org/>, hämtad: 2018-02-12.
- [5] C. Nessman, "A brief history of augmented reality," 21 November 2017. [Online]. Tillgänglig: <http://adsreality.com/history-of-augmented-reality-infographic/>, hämtad: 2018-02-12.
- [6] DING, "Augmented Reality - Allt du behöver veta om AR," [Online]. Tillgänglig: <http://augmentedreality.se/>, hämtad: 2018-02-12.
- [7] Microsoft, "HoloLens hardware details," [Online]. Tillgänglig: https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details, hämtad: 2018-02-12.
- [8] Oasis Open, "MQTT Version 3.1.1", [Online]. Tillgänglig: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>, hämtad: 2018-02-20
- [9] Unity Technologies, "Features," [Online]. Tillgänglig: <https://unity3d.com/unity>, hämtad: 2018-02-21.
- [10] soft2tec GmbH, "Nexonar Product & Applications", [Online] Tillgänglig: http://www.nexonar.com/files/nex_broschuereanwendprod-2017-web.pdf, hämtad 2018-02-21
- [11] Microsoft, "Visual Studio IDE overview," [Online]. Tillgänglig: <https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide>, hämtad: 2018-02-21.
- [12] Scrum.org, "What is Scrum?," [Online]. Tillgänglig: <https://www.scrum.org/resources/what-is-scrum>, hämtad: 2018-02-21.
- [13] Microsoft, "Using the HoloLens Emulator", [Online]. Tillgänglig: https://developer.microsoft.com/en-us/windows/mixed-reality/using_the_hololens_emulator, hämtad: 2018-02-22
- [14] P. Patierno, "M2MQTT", [Online]. Tillgänglig: <https://m2mqtt.wordpress.com>, hämtad: 2018-02-22
- [15] PTC, "PTC Adds Augmented Reality Leader Vuforia to Portfolio", 2015, [Online]. Tillgänglig: <https://www.ptc.com/en/about/history/vuforia>, hämtad: 2018-04-03

- [16] H. Habbak och D. Cushnan, Developing AR Games for iOS and Android, Vuforia SDK and how it helps delivering the AR experience, Birmingham, United Kingdom, 2013. [Online]. Tillgänglig: <https://doc.lagout.org/programmation/Android/Developing%20AR%20Games%20for%20iOS%20and%20Android%20%5BCushnan%20%26%20El%20Habbak%202013-09-24%5D.pdf>, hämtad: 2018-04-04
- [17] Nexonar Assembly Scout - MQTT Setup Instructions, Rüsselsheim, Germany, Muhammad Ali Shahid (Nexonar), 2018-02-01.
- [18] Mosquitto, "Download", [Online]. Tillgänglig: <https://mosquitto.org/download/>, hämtad: 2018-04-18
- [19] Vuforia, "Optimizing Target Detection and Tracking Stability", [Online]. Tillgänglig: <https://library.vuforia.com/articles/Solution/Optimizing-Target-Detection-and-Tracking-Stability.html>, hämtad: 2018-04-18
- [20] Unity3d.com, "Unity Teams," [Online]. Tillgänglig: <https://unity3d.com/teams>, hämtad: 2018-05-06.
- [21] docs.microsoft.com, "Hologram Stability," [Online]. Tillgänglig: <https://docs.microsoft.com/en-us/windows/mixed-reality/hologram-stability>, hämtad: 2018-05-06.
- [22] Phyyl, "OboolNet", 2015, [Online]. Tillgänglig: <https://github.com/Phyyl/OboolNet>, hämtad: 2018-05-06
- [23] docs.microsoft.com, "Gaze," [Online]. Tillgänglig: <https://docs.microsoft.com/sv-se/windows/mixed-reality/gaze>, hämtad: 2018-05-06.
- [24] docs.microsoft.com, "Gestures," [Online]. Tillgänglig: <https://docs.microsoft.com/sv-se/windows/mixed-reality/gestures>, hämtad: 2018-05-06.
- [25] docs.microsoft.com, "Use your voice with HoloLens," [Online]. Tillgänglig: <https://support.microsoft.com/sv-se/help/12647/hololens-use-your-voice-with-hololens>, hämtad: 2018-05-06.
- [26] docs.microsoft.com, "Hologram Stability," [Online]. Tillgänglig: <https://docs.microsoft.com/en-us/windows/mixedreality/hologram-stability>, hämtad: 2018-06-03.
- [27] github.com, "M2MQtt," [Online]. Tillgänglig: <https://github.com/eclipse/paho.mqtt.m2mqtt>, hämtad: 2018-06-03.
- [28] internetmuseum.se, "Hem-pc-reformen kickar igång: Nu ska alla kunna ha en dator hemma," [Online]. Tillgänglig: <https://www.internetmuseum.se/tidslinjen/hem-pc-reformen-snabbare-utvecklingen/>, hämtad: 2018-06-03.