CHALMERS

Augmented Reality med fysikalisk simulering

Kandidatarbete vid Signaler och System

Adam Lindkvist
Madeleine Arnason
Liam Håkansson
Marcus Hilding Södergren

Institutionen för Signaler och System
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2017

Abstract

The project aims to explore the possibitiy and practicality of constructing an application for the HoloLens to visualize a robot cell using the physics engine AGX Dynamics. By attempting to develop this application the potential of the tools at hand are examined and eventual problems with the technology are identified. AGX Dynamics and HoloLens are incompatible at the present time due to the early developmental stage of the latter. Instead two applications are built - One for visualizing with the HoloLens and one to showcase the advantages of using AGX Dynamics for the simulation.

Sammandrag

Arbetets syfte är att undersöka möjligheten samt utvärdera hur givande det är att konstruera en applikation till HoloLens för att visualisera en robotcell med hjälp av fyiskmotorn AGX Dynamics. Genom att försöka konstruera denna applikation undersöks potentialen i vektygen och problem med tekniken identifieras. AGX Dynamics och HoloLens är inkompatibla i dagsläget till följd av den senares tidiga utvecklingsstadium. Istället byggs två applikationer - En för att visualiseras med HoloLens och en för att visa fördelarna med användandet av AGX Dynamics i simuleringen.

Innehåll

Begreppslista

AGX Dynamics Fysikmotor tillverkard av Algoryx. Fokuserar på realtidssimulering.

AGX Unity Insticksmodul till AGX Dynamics för att använda komponenter baserat på dess fysikmotor i utvecklingsmiljön Unity.

Algoryx Svenskt företag baserat i Umeå. Tillverkar mjukvara för simulering, däribland fysikmotorn AGX Dynamics.

AR *Augmented Reality*, förstärkt verklighet. Innebär att digital förstärkning projiceras ovanpå verkligheten för att ge användaren extra information.

HoloLens AR-glasögon framtagna av Microsoft.

IDE *Integrated Development Environment*, Utvecklarmiljö för programmering med funktioner som förenklar arbetet.

json Ett filformat som sparar objekt och bevarar variabler.

Prefab *Prefabricate*, en sorts "ritning" för objekt i Unity som skapas och används som mall.

Process Simulate Simuleringprogram för produktionsmiljöer.

Unity Utvecklingsmiljö för datorspel och andra virtuella miljöer.

VR *Virtual Reality*, virtuell verklighet. Visualiering där åskådarens synfält helt ersätts med en digital bild.

VS *Visual Studio*, IDE för `C#` med integrerat stöd för Unity och HoloLensutveckling.

1 Inledning

En utmaning inom industrin är att vid utveckling och konstruktion av produktionsmiljöer på förhand förutse hur resultatet kommer att bli. Detta är viktigt då det är kostsamt både att konstruera produktionsmiljöer och att ha produktionsanläggningar stillastående. För att hantera utmaningen finns idag programvaror för att planera och visualisera dessa miljöer, till exempel Process Simulate[1]. Idag kan man dock se att visualiseringstekniken har gjort stora framsteg, inte minst i och med AR, Augmented Reality eller Förstärkt Verklighet. I framkanten av den tekniken ligger Microsofts AR-glasögon HoloLens som skulle kunna innebära en revolution för visualisering av just produktionsmiljörer.

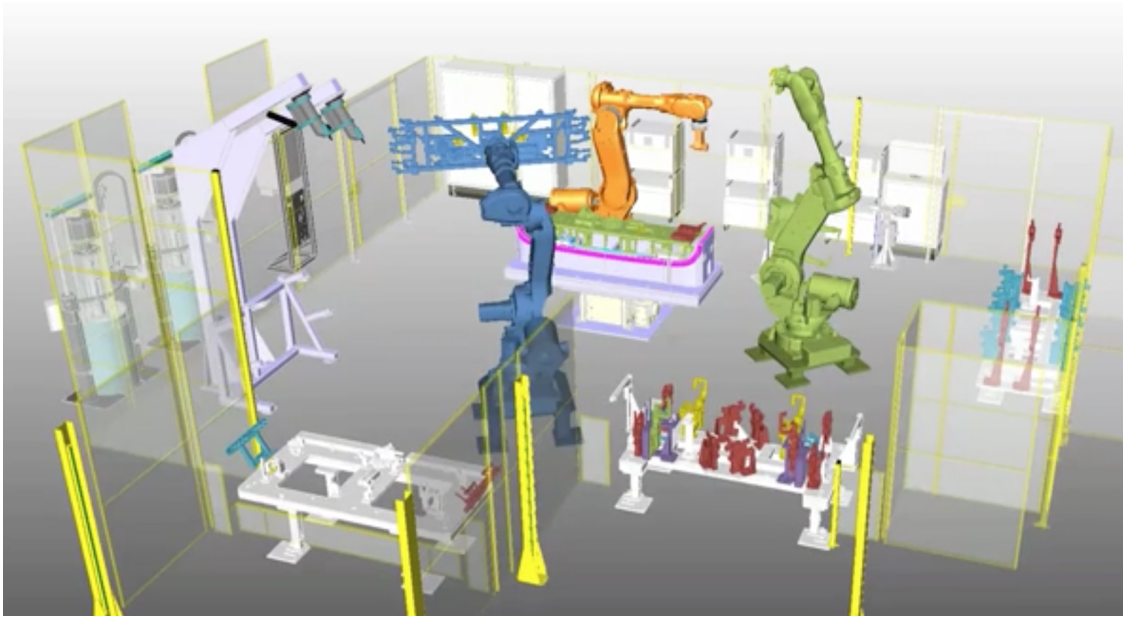
Att genomföra fysikaliska simuleringar finns det också bra teknik för och detta har gjorts i flera år nu, exempelvis med Algoryx fysikmotor AGX Dynamics [2]. Genom att kombinera dessa tekniker för planering, simulering och visualisering kan stora förbättringar och effektiviseringar ske inom såväl planeringsstadiet som utvecklingsstadiet av produktionsanläggningar. Utmaningen här är att hitta ett sätt att smidigt kunna arbeta med information från olika programvaror för att kunna bygga en sammankopplad enhet.

En sammansättning uppbyggt på detta sätt öppnar upp nya möjligheter att få ut nyckelinformation ur systemet utan att behöva göra en specifik simulering för varje problem. Ett tillämpningsområde är att kunna se hur tillverkningsmiljön kommer fungera rent praktiskt, men möjligheterna som kommer med att ha hela systemet representerat och visualiserat på ett bra sätt är många fler. Fysikaliska simuleringar kan modellera de krafter som uppstår, vilket är viktig information till exempel för att kunna beräkna slitage och därmed kunna planera för underhåll. Att verklighetstrogen fysik simuleras gör också att systemet kommer att reagera som i verkligheten vid interaktion. Interaktionen i sig är möjligt med HoloLens-glasögonen som kan registrera både bärarens position och gester som användaren använder sig av.

Detta sätt att bygga upp en sammansättning av delprogrammen öppnar också upp för att skicka information tillbaka till ett bakomliggande styrsystem. Då kommer hela systemet kunna styras, och därmed reagera, som om det hade funnits i verkligheten.

1.1 Syfte

Projektets huvudsakliga syfte är att undersöka hurvida det är praktiskt att visualisera modeller simulerade med Algoryx fysikmotor AGX Dynamics med Microsoft HoloLens, samt att i högsta möjliga mån genomföra en sådan simulering och visualisering. Vidare ämnar arbetet undersöka vilka hinder som ligger i vägen för att en sådan simulering och visualisering ska vara praktisk, samt ifall och hur dessa faktorer behöver påverkas för att göra en sådan simulering och visualisering praktiskt genomförbar.



Figur 1: Volvos robotcell som den ser ut i Process Simulate. Demonstrerar robotar som i tur och ordning hämtar in en plåtbit från nedre högra hörnet för att svetsas och limmas. Det finns tre robotar i cellen, en blå, en orange och en grön, samt ett vridbord som står i mitten av cellen mellan robotarna.

1.2 Visualisering av produktionslinjer

Att konstruera och revidera befintliga produktionslinjer är dyrt och det finns därför stor anledning till att göra en noggrann planering av var och hur utrustning ska vara monterad och konfigurerad [3]. Delar av denna planering kan underlättas med hjälp av datorsimulering av robotarna i produktionsmiljön. Det kan dock vara svårt att få en intuitiv känsla för hur den färdigbyggda produktionslinjen kommer att se ut och bete sig utifrån en bild på en skärm. AR-tekniken öppnar då upp en möjlighet att gå ut i den avsedda lokalen och projicera tänkta nybyggnationer och

förändringar i det verkliga utrymmet. Detta ger en uppfattning om skala och rum på ett sätt som inte är möjligt på en skärm [4].

Volvo har en produktionscell med tre robotar, ett vridbord samt kringmiljöer som en testcell vilken de nu använder som grund i sitt arbete att utveckla visualisering och simulering, cellen ses i Figur 1. Idag finns cellen fysiskt i deras verkstad i Umeå samt modellerad och animerad i simuleringsverktyget Process Simulate. Målet är att denna cell skall användas i försöken att visualisera och simulera då detta är ett steg på vägen i arbetet med att vidare utveckla tekniker för förbättra upplevelse och kontroll över cellen.

1.3 Simulering

En fysikalisk simulering är ett sätt att beräkna hur föremål hade betett sig i verkligheten med hjälp av matematiska modeller av fysik. Det krävs då först indata i form av tillstånd för de objekt som ska modelleras. Detta kan till exempel vara position och hastighet för en stelkropp. Dessa tillstånd ändras sedan i enlighet med fysikaliska lagar, i detta fall Newtons lagar för dynamik och kinematik i rummet. Det sökta resultatet är då till slut ett verklighetstroget beteende för de givna objekten med avseende på de simulerade faktorerna.

Fysikalisk simulering av produktionslinjer är användbart för att utröna eventuella problem innan produktionslinjen är byggd. Här kommer indatan oftast i form av de operationer som ska utföras i produktionen. Alltså huvudsakligen positioner, vinklar, hastigheter och andra kinematiska storheter. Utdatan som efterfrågas är till stor del dynamiska storheter som krafter, moment, rörelser och spänningar för att till exempel beräkna slitage samt för att kontrollera att operationerna utförs som planerat innan man testar på hårdvaran. Det kan till exempel vara intressant att se hur olika föremål rör sig relativt varandra för att exempelvis förhindra konflikt i operationerna.

En simulering innebär att en rad matematiska modeller för ett objekts fysikaliska beteende under vissa stimuli ställs upp och löses. I praktiken kan detta göras genom att antingen genomföra alla beräkningar i förväg och sedan spela upp visualiseringen i efterhand, eller genom att man gör beräkningar och visualisering i realtid. Dessa har båda för- och nackdelar. Med en “förinspelad” simulering kan mer omfattande beräkningar och större modeller simuleras utan att offra prestan-

da. Detta sker dock på bekostnad av interaktivitet. Vid en realtidssimulering kan i princip processen störas eller påverkas på annat godtyckligt sätt under simuleringens gång, varpå modellen då kan reagera verklighetstroget på detta. Dock begränsas detta i högre grad av den tillgängliga beräkningskraften, vilket leder till att en mer selektiv simulering får genomföras. Till exempel kan de fysikaliska modellerna förenklas eller så får en mindre modell simuleras.

1.4 Visualisering

Att presentera simulerad data på ett bra och lättillgängligt sätt är viktigt då simuleringar ofta genererar mycket information vilken är oförståelig om den inte sätts i en kontext. Ofta handlar det inte bara om att då göra det på ett bra sätt, utan att också använda rätt teknik och hårdvara. För till exempel krafter eller hastigheter kan grafer eller tabeller vara ett bra sett att presentera data, men för att presentera hur en robot rör sig i ett rum en en tabell över vinkeldata hos dess leder vara svårt för användaren att ta till sig på ett givande sätt. Där kan det då till exempel vara bättre att presentera en 3D-modell av roboten och visa dess rörelser grafiskt. För att visualisera de modeller som behandlas i detta arbete används AR-teknik.

1.4.1 Augmented Reality

AR-tänkandet i sig är inget nytt, utan har till exempel länge funnits i flygplan i form av Heads Up-Displayer för att förse piloten med nödvändig information utan att denna ska behöva titta ner på instrumenten [5]. Det blir också vanligare inom industrin där nya lösningar ständigt hittas för att hjälpa användaren. Till exempel börjar det utvecklas manualer med AR-teknik för att underlätta identifiering av komponenter samt ge instruktioner till oerfarna användare [6]. Vi har också sett exempel på att tekniken kan användas av och attrahera den breda massan med det omtalade spelet Pokémon GO som använder AR-teknik för att visa animerade figurer i verkligheten med hjälp av kameran i en smart mobiltelefon [7]. Exempel på dessa tillämpningar ses i Figur 2.

En systerteknik till AR är VR, Virtual Reality, som blivit populär de sista åren med produkter som HTC Vive [8] och Oculus Rift [9]. Konceptet går ut på att användarens syn av verkligheten blockeras och ersätts med digitala displayer som placeras framför ögonen. Detta kombinerat med teknik för att känna av använ-

(a) Pokémon GO använder mobilkameran för att avbilda en Pokémon i verkligheten som användaren sedan kan fånga i spelet.[12]

(b) En HeadsUp-display i cockpiten i ett flygplan. Den används för att visa relevant information till piloten så att denna slipper titta ner på instrumenten. Detta är en av de första tillämpningarna av AR.[13]

Figur 2: Två exempel på tillämpning av AR. Till vänster för att visa figurer i ett mobilspel och till höger som hjälpmedel för piloten i ett flygplan.

dares huvudrörelser och låta bilden stanna på en fast plats i rummet oavsett hur användaren rör på sig ger illusionen av en virtuell verklighet [10]. Denna teknik faller i ena extremänden av det så kallade “Mixed Reality”-spektrat [11]. Mixed reality syftar till blandningen av fysisk och digital verklighet. I ena änden faller VR, som helt ersätter element av den fysiska verkligheten med digitala motsvarigheter. I andra änden faller Augmented Reality, AR, vars syfte är att förstärka och komplettera den befintliga fysiska verkligheten.

AR bygger på att man överlagrar någon form av visuell information på verkligheten. I boken *Understanding Augmented Reality* vill Alan Craig ta det så långt som att AR i sig är ett medium som det finns flera tekniker för att hantera och jämför det med film [4]. För att skapa en film krävs det kunskap om all teknik, men man måste också använda filmen som medie för att lyfta fram sitt buskap. Samma tankegång kan alltså appliceras på AR.

Genom att visualisera Volvos produktionscell med AR är förhoppningen att användaren enklare skall kunna få en känsla för cellen. Att själv kunna röra sig runt ett objekt som placerats i verkligheten tillåter användaren att fokusera på detaljer eller att få en överblick och se helheten.

1.5 Avgränsningar

Till att börja med avgränsar sig arbetet till att arbeta med färdig rörelsedata i en fil extraherad från Process Simulate, snarare än kontinuerligt ladda in data från källan i realtid. Vidare avgränsas arbetet till att behandla ett förbestämt exempelfall snarare än att bygga en generell modell för modellering, simulering och visualisering. Inte heller andra fysikmotorer, AR-glasögon eller utvecklarmiljöer har undersökts.

1.6 Disponering

I Kapitel 2 beskrivs projektet mer noggrant och komponenterna som använts presenteras. En kravprofil för produkten ställs också upp. Kapitel 3 följer lösningsgången för projektet och detaljer kring hur projektets verktyg har använts presenteras. Lösningen utvärderas med avseende på kravspecifikationen i Kapitel 4 och produkten utvärderas. I Kapitel 5 diskuteras resultatet bredare med fokus på koncepten, begränsningar och vidare utvecklingsmöjligheter.

2 Ansats

Produktionslinjer är viktig del vid massproduktion för att maximera genomflödet på material och produkter. Vid nyinstallation och uppgraderingar av automatiserade verkstadsmiljöer kan det vara svårt att på förhand kunna bekräfta att systemet kommer att fungera som man tänkt. Värdet hos simulering ligger då i att kunna förutsäga systemet, dels så att de fungerar rent designmässigt, dels i att kunna förutsäga hur produktionen kommer att fungera. Det senare är viktigt för att kunna förutspå och planera upp produktionen i sin helhet, vilket kan ha stor inverkan på effektiviteten hos produktionen [14]. Den designmässiga biten kommer att vara viktigare tidigare i processen för att kunna utveckla produktionslinjen så att den passar lokalen och andra komponenter i produktionen.

2.1 Visualisering av simulering med AR

Att kombinera AR-teknik med fysikalisk simulering innebär framförallt en kombination av visualiseringsfördelarna från AR och informationen från simuleringen. Det blir då bli enklare att få ett mer verklighetstroget perspektiv och helhetsintryck än vad som är möjligt på en konventionell datorskärm. Det möjliggör även interaktivitet med simuleringen på en ny nivå genom att man då kan använda sin egen kropp eller miljön runt omkring hologrammet för att påverka simuleringen på olika sätt. Ett exempel är att testa hur robotar beter sig om en människa kommer i närheten av rörliga delar.

Jämfört med att göra samma simulering i VR fås övertaget att användaren hela tiden kan vara medveten om sin faktiska omgivning. Ingen dedikerad miljö krävs för att utföra simuleringen, och befintlig utrustningen behöver inte simuleras, då den fysiska utrustningen kan agera en del av simuleringen.

Simuleringen av robotcellen består främst av att konvertera positioner på robotarnas olika delar till dynamiska storheter som krafter och moment. Dessa positioner kommer i form av vinklar på robotarnas olika leder som har exporterats från Process Simulate.

Visualiseringen består av ett hologram av antingen hela eller en del av cellen. Detta hologram visas upp med hjälp av Microsoft HoloLens, vilken visas i Figur 3, och visar främst upp robotarnas rörelser i en arbetscykel.

Figur 3: Bild av Microsofts HoloLens-glasögon. Glasögonen placeras på användares huvud och ger genom displayer i glasen möjligheten att superponera digitala bilder på den verkliga omgivningen.

Simuleringen och visualiseringen utvecklas separat på grund av kompabilitetsproblem som beskrivs mer ingående i Kapitel 3. Detta innebär att det inte visas något kablage i HoloLensapplikationen men i övrigt påverkas inte det visuella resultatet alls eftersom krafter och moment inte syns visuellt.

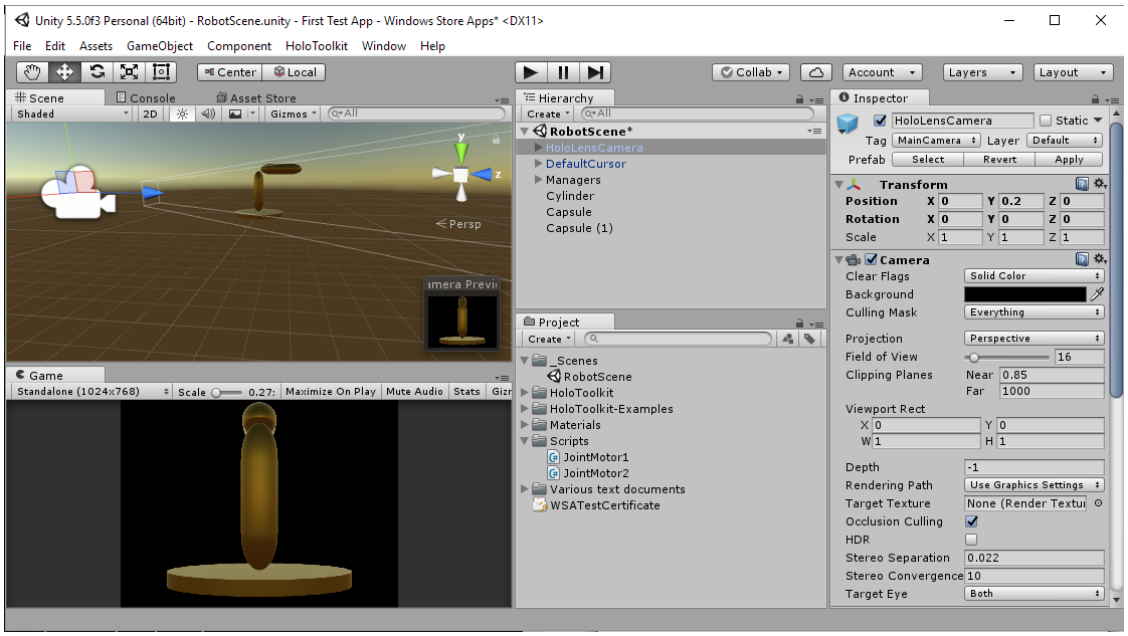
2.2 AR glasögonen HoloLens

HoloLens är ett Augmented Reality-headset från Microsoft. Displayer har monterats som glasen i glasögon, i vilka tillägg till verkligheten kan visas. Detta fungerar genom att en rad sensorer monterade över glasen på headsetet läser av bärarens omgivning och ger återkoppling till programmet som körs. Därmed kan illusionen av att de simulerade objekten interagerar med verkligheten upplevas. Headsetet har en djupseende kamera som gör kartläggningar av omgivningen, så kallad Spatial Mapping, vilken gör det möjligt att till exempel placera ett föremål på ett verkligt bord eller ett internetfönster på en vägg [15]. Det går även att programmera in i sin app att använda Spatial Mapping kontinuerligt, som exempel skulle man i ett spel kunna gömma objekt bakom verkliga saker så att de endast är synbara om man förflyttar sig runt hindret. Slutligen går det även att programmera så bäraren själv kan interagera med objekten i appen [16]. För att utveckla mjukvara till Holo-

Lens har Microsoft integrerat den med några olika mjukvaruplattformar, däribland spelmotorn Unity.

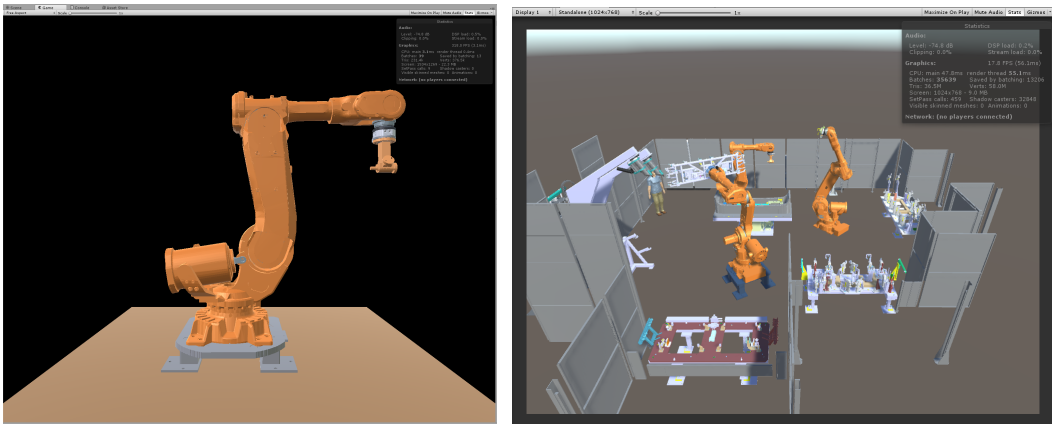
2.3 Spelmotorn Unity, en utvecklingsmiljö för HoloLens

Unity är en spelmotor och utvecklarmiljö som är fritt tillgänglig för privatpersoner och studenter. Den låter användare behandla objekt och många funktioner grafiskt istället för i kod under utvecklingen. För mer avancerat beteende kan man även skriva mindre program, script, i programmeringsspråket `C#`, som kan associeras med objekt i applikationen. I Unitys gränssnitt, avbildat i Figur 4, är skärmen indelad i en rad moduler som kan organiseras efter tycke och smak. Dessa innefattar i allmänhet, men är inte begränsade till, hierarki, projektmapp, kamerabild, en fri bild av "rummet" som spelet designas i, även kallat scenen, samt en inspektör för objekten.



Figur 4: Unitys grafiska gränssnitt. På bilden syns inspektören längst till höger, hierarkin ovanför projektfönstret direkt till vänster om den, samt scenen ovanför kamerafönstret längst till vänster.

Mycket av det huvudsakliga arbetet skedde i Unity. Hit importerades modellerna, avbildat i Figur 5, för att behandlas så att de kunde simuleras fysikaliskt för ledrörelse och kablage. Detta innebar en simulering med relativt enkla fysikberäkningar för den robot som skulle köras på HoloLens jämfört med den som körs med AGX Dynamics insticksmodul till Unity, AGX Unity.



(a) Roboten som den ser ut i kameravyn i Unity.

(b) Hela cellen importerad i Unity.

Figur 5: Skärmdumpar på modellerna som vi jobbat med i Unity. Roboten i den vänstra bilden är den vänstra bakre roboten i cellen.

AGX Dynamics är en fysikmotor fokuserad på simuleringar för VR i realtid [17]. Den är särskilt bra på till exempel simulering av kablage, något som är intressant vid simulering av robotar. En annan faktor som gör den intressant för detta arbete är att de har en insticksmodul till Unity under utveckling, vilket innebär att den i princip skulle kunna användas till HoloLens.

2.4 Krav på applikation

För att konkretisera arbetets syfte arbetstekniskt sammanställdes en rad krav på vad applikationen skulle uppnå. Dessa innefattar huvudsakligen krav på prestanda och funktionalitet och ses i Tabell 1. Målet är alltså att en applikation till HoloLens med fullständig simulering via AGX Dynamics av hela cellen ska konstrueras. Detta innebär att i cellen ska alla rörliga komponenter, 3 robotar och ett vridbord, vara helt simulerade med dynamisk ledrörelse, kollisionsdetektion och kablage. För kablaget ska krafter kunna beräknas och visas beroende på olika faktorer såsom fästpunkter och hastighet på robotens rörelse. Simuleringen ska också kunna styras under pågående körning av applikationen med ett gränssnitt i HoloLens för att till exempel pausa eller ändra hastighet på robotarna. Ifall detta inte är uppnåeligt i sin helhet är minimikravet på den funktionalitet som ska finnas att en robot ska kunna projiceras med förprogramerad rörelse. Användaren ska i någon mån kunna interagera med denna.

Tabell 1: Krav- och målspecifikation som sattes upp för att konkretisera arbetets syfte.

Krav- och målspecifikation			
Krav	Specifikation	Prioritet	Kommentar
Fungerande app med robot i HoloLens	Projicera robot	1	En eller flera robotar ska projiceras
	Animerad robot	2	Robotarna ska kunna röra sig antingen automatiskt eller vid interaktion
	Interaktion med objekt	3	Ska kunna påverka objektet på något sätt med en handgest
Mål	Specifikation	Prioritet	Kommentar
Fullständig modell	Alla robotar samt vridbord	2	Projicera och animera alla tre robotar samt vridbordet
	Omliggande miljö	5	Projicera den fullständiga cellen inklusive väggar, verktyg och dylikt
Kablage	Animerat kablage	1	Inkludera kablage i simuleringen.
	Kraftberäkningar	4	Beräkna hur kablaget påverkas av krafter i syfte att använda för att utvärdera slitage.
Interaktivitet	Styrning av simulation	3	Gränssnitt för att styra simuleringen inifrån med hjälp av menyer eller gester

3 Lösning

Syftet med arbetet var att konstruera en applikation där fördelarna hos både visualisering med AR och simulering med en mer avancerad fysikmotor kunde belysas, samt att demonstrera detta genom simuleringen av en robotcell. På grund av inkompabiliteter mellan de komponenter som valts för att visa detta har det dock visat sig omöjligt att göra detta i en enda applikation. Det visade sig även av prestandaskäl omöjligt att simulera hela cellen, varför simuleringen begränsades till en enda, förenklad, modell av en robot.

Arbetet har därför resulterat i två applikationer. En baserad uteslutande på Unity som kan visas i HoloLens, vilket representerar det första huvudmålet med grundprojektet rörande visualisering. Den andra är byggd med AGX Dynamics som fysikmotor och körs som en 64-bitarsapplikation på en PC. Detta täcker det andra målet i arbetet rörande fysiksimulering.

I den första applikationen kan en kinematiskt simulerad robot visas på HoloLensen, med godtagbar uppdateringsfrekvens och latens. Modellen kan betraktas från godtycklig vinkel och perspektiv och kan av användaren flyttas runt i rummet.

Den andra applikationen tillåter en fullständig dynamisk simulering av roboten. Snarare än kinematiska abstraktioner av ledrörelse simuleras faktiska dynamiska leder som kan konfigureras med olika värden på till exempel dämpning eller tålighet. De reagerar på interaktion med sin omgivning, så om till exempel en arm kolliderar med ett objekt reagerar leden enligt dess fysikaliska modell. Motorn tillåter även simulering av kablage med olika materaldata som påverkar dess beteende. Detta kan i princip användas för att ta ut kraftdata som till exempel spänning eller friktion för kabeln i AGX Dynamics, men detta har ännu inte implementerats i insticksmodulen för Unity.

3.1 Import av modell

Modellen för cellen importerades från Process Simulate i form av en .jt-fil. Denna fil kan inte direkt importeras till Unity eftersom den innehåller så kallade BREPS, *boundary representation surfaces* [18]. Dessa måste konverteras till enklare ytor som trianglar för att ge ett filformat som Unity kan hantera. Detta gjordes med hjälp av CAD-programmet SpaceClaim som utvecklas av ANSYS. Det visade sig då att

modellen var placerad väldigt långt från origo och translaterades därför innan den exporterades till ett annat filformat. Filformatet .obj från Wavefront Technologies valdes då det var relativt enkelt att importera till Unity.

Detta tillvägagångssätt kräver en del handpåläggning eftersom strukturen i filen som exporteras från Process Simulate inte bevaras när den läses in i SpaceClaim. Detta resulterar i att det vid ett senare skede krävs en del arbete för att strukturera upp modellen så att cellen kan delas in i de delar som rör sig oberoende av varandra. Även faktumet att Unity använder ett annat koordinatsystem än Process Simulate resulterar i viss bearbetning av modellen. Unity använder ett vänsterorienterat koordinatsystem där Y-axeln är vertikal till skillnad från det högerorienterade koordinatsystemet där Z-axeln är vertikal vilket används i de flesta andra tillämpningar.

Modellen från Process Simulate är extremt detaljerad och kräver därmed en hel del datorkraft för att bearbeta. Detta resulterade i att hela cellen inte kunde visas i Unity med tillräckligt hög uppdateringsfrekvens. Detta löstes genom att förenkla geometrin. En algoritm i Unity som reducerar antalet trianglar som representerar en given yta gjorde en hel del för att få ner modellstorleken men var långt ifrån tillräckligt. Därför valdes det att ta bort de mest oväsentliga delarna i cellen från simuleringen. Resultatet blev att enbart en robot kan visas upp i HoloLens. Denna robot är dessutom rensad från mindre och interna komponenter som kullager och skruvar.

3.2 Gränssnitt mellan simulering och visualiseringsmedium

Vid arbetet med den exporterade cellen i Unity uppdagades att AGX Unity endast går att köra med 64-bitarssystem och att HoloLens operativsystem är 32-bitars. AGX Dynamics däremot går att köra i 32-bitar vilket betyder att det är Algoryx insticksmodul till Unity, AGX Unity, som behöver utvecklas för 32-bitar. I och med detta är det inte möjligt att utveckla en applikation för HoloLens-glasögonen med AGX-fysik i nuläget.

Applikationen med AGX Dynamics kan endast köras på en PC, vilket innebär att användaren inte själv kan gå runt och inspektera roboten ur olika vinklar som var tänkt. Då valdes att ha en fritt rörlig kamera, vilket låter användaren undersöka scenen ur alla tänkbara perspektiv. Vidare har man tillgång till att använda mus

och tangentbord för att uppnå i stort sett godtycklig nivå av kontroll över både simulering och visualisering.

3.3 Simuleringen i Unity

Fysiksimulering i Unity sker automatiskt och styrs av två metoder inbyggda i Unitys *MonoBehaviour* ramverk [19]. Den första är *Update* [20] som körs mellan varje bildruta som visas, detta betyder att tiden mellan dess körningar inte alltid är konstant utan varierar med bildhastigheten. *FixedUpdate* [21] är den andra metoden vilken alltid körs med konstant tid mellan uppdateringarna, vanligtvis 100 körningar per sekund. Den senare är alltså att föredra att använda då det är tidsberoende fysik som simuleras.

För användaren är det enkelt att implementera fysikaliska effekter på objekt. Genom att till exempel applicera en kraft på ett objekt i FixedUpdate-metoden kommer fysikmotorn beräkna effekten på objektet och alla andra objekt i omgivningen som påverkas. Användaren står alltså inte speciellt nära fysikmotorn utan anger endast vilka förutsättningar som råder.

3.4 Abstraktion av ledrörelse

För att sätta cellen i rörelse behöver rörliga komponenter implementeras. I Volvos cell är alla rörliga delar roterande leder som efterliknas genom att sammanlänka objekt och endast lämna en rotationsriktning som frihetsgrad. Det är sedan rotationen kring denna frihetsgrad som utgör grunden för all rörelse av modellen. I arbetet med att efterlikna ett inspelat rörelsemönster för cellen ingick dels hantering av den inspelade datan och dels att sätta lederna i rörelse.

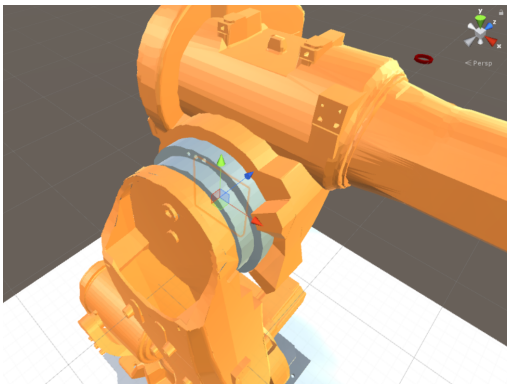
Till de importerade modellerna skapades en skelettstruktur för att kunna hantera de rörliga delarna i cellen. AGX Unity har en rad olika *Constraints*, rörelsebegränsningar, varav en så kallad *Hinge Joint*. Denna fungerar som en gångjärnsled, där alla frihetsgrader utom en rotation runt en axel kan låsas mellan två objekt. Axeln definieras enkelt av användaren i Unitys visuella gränssitt. Unity har liknande motsvarigheter bland sina inbyggda komponenter, dock med mindre flexibilitet gällande hur användaren kan definiera rotationsaxeln. I Unity matades istället rå vinkeldata till objektens rotationsmatris för att simulera en kinematisk rörelse.

Skelettet lades upp så att två fundamentala 3D-objekt från Unity placerades bredvid varandra längs en axel genom den led som skulle rotera, vilket visas i Figur 6. För AGX-simuleringen applicerades en HingeJoint på det ena objektet, som fick representera det "yttre" objektet som skulle rotera kring det inre. Det inre designerades i HingeJoint som ett föräldraobjekt till det yttre, vilket innebär att all rörelse av det yttre objektet sker relativt det inre. HingeJointens rotationsaxel definierades sedan så att den sammanföll med rotationsaxeln för leden.

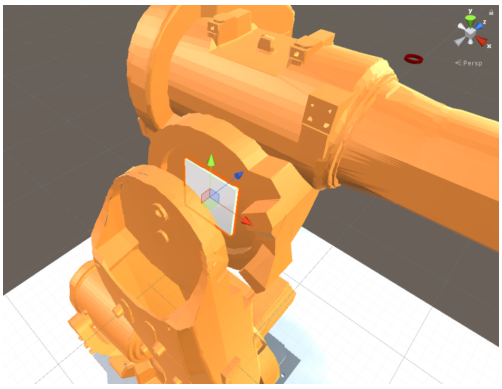
För simuleringen utan AGX fick objekten ordnas i en föräldrahierarki så att basen låg "överst" i hierarkin, och varje påföljande led sedan låg ett steg ner. Sedan applicerades en lokal rotation på den yttre skelettkomponenten. Detta innebar att varje komponent endast roterade relativt den föregående komponenter runt deras gemensamma led.

I Process Simulate var cellen programmerad att utföra en arbetssekvens som motsvarade cellens arbetsuppgift. För att köra samma simulering i Unity spelades rörelsedata in från de rörliga delarna i cellen och sparades i en json-fil. Denna fil läses in i Unity med hjälp av Unitys JsonUtility klass som kan avkoda json-formatet [22]. Vinkeldatan användes för att styra lederna i de två olika applikationerna. I Unity läggs filer som dessa i en mapp som heter StreamingAssets vilken inte kompileras in i programmet utan endast läses, det betyder att man kan byta ut filen i den mappen i det färdiga programmet och därigenom byta beteende på simuleringen utan att bygga ett nytt program. Kraven på filen är att den ska ha samma namn samt vara uppbyggd med den klass-struktur som programmet använder.

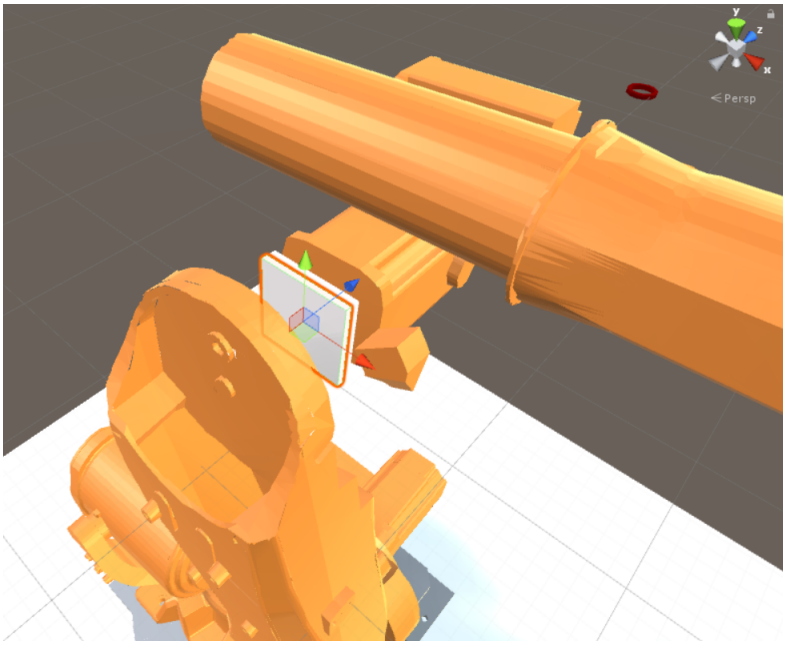
De byggda applikationerna använder en struktur där varje tidpunkt i inspelningen är ett objekt, kallad *Tick*, innehållandes all data för cellen, det vill säga vinklarna för lederna i robotarna och vridbordet i cellen. Alla dessa objekt motsvarandes tidpunkter placeras sedan i en lista, *TickList*, som i sin tur är ett objekt. Strukturen kan ses i Figur 7. I inspelningen är inspelad i 100 ticks per sekund, vilka sedan kan spelas upp i olika hastigheter för få cellen att röra sig i olika hastigheter.



(a) Skelettet är dolt bakom de grafiska komponenterna under simuleringen. Skelettet består av rätblock, var konturer här syns bakom robotens skal.



(b) Varje led representeras av två skelett-delar, en för varje anslutande komponent. Här är en av ledkomponenterna dolda för att visa den ena skelettkomponenten.

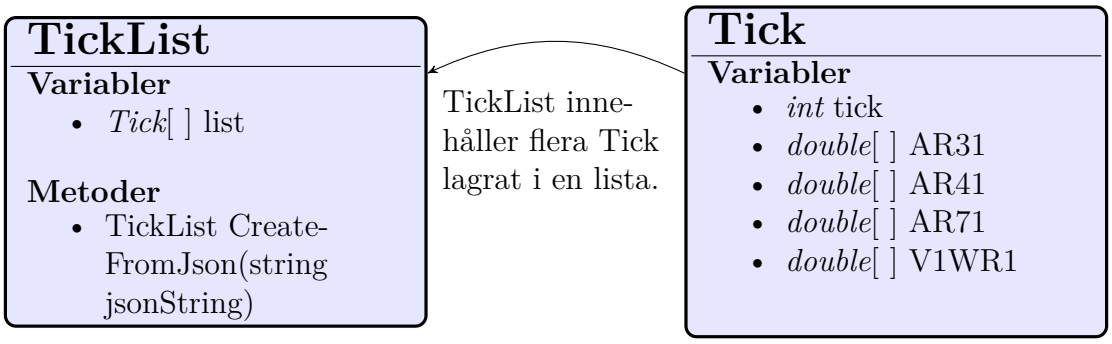


(c) De två skelettkomponenterna ligger axisymmetriskt runt ledens rotationsaxel, här representerad av den blå pilen längs z-axeln.

Figur 6: Det skelett som rörelserna för roboten är uppbyggda på ses på bilderna ovan. Modellen har sedan associerats till detta skelett för att modellen skall följa ledernas rörelser då simuleringen styr skelettets rörelser.

3.5 Visualisering med Augmented Reality

Som tidigare nämnt är AR, Augmented Reality, ett alternativ till mer traditionell visualisering som möjliggör en bättre upplevelse för användaren. I detta projekt är målet bland annat att visa upp möjligheterna för att visualisera en produktionscell i sin tänkta miljö.



Figur 7: Klass-struktur för programmen samt json-filerna som används för lagring av simuleringsdata. *ClassList* innehåller en lista med flera *Tick* vila i sin tur innehåller datan för cellen under en diskret tidpunkt.

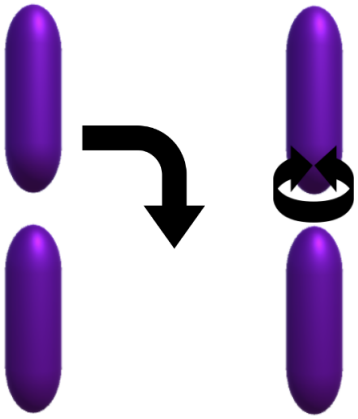
När det kommer till valet av AR-glasögon så finns det idag inte så mycket att välja på som har den prestanda som behövs till detta projekt. HoloLens och Meta 2 är de AR-glasögon som någorlunda matchar prestandakraven. Det som skiljer dem mest åt och som avgör vilka som är mest passande är projektionsytan och portabiliteten.

	HoloLens	Meta 2
Projektionsyta	35° [23]	Hela glasögonens yta (90°) [23]
Portabel	Ja	Nej

För att i högsta möjliga mån utnyttja fördelarna med AR som visualiseringsmedium implementerades en rad standardfunktioner för HoloLensen. *Gaze* används för att interagera med objekt i användares blickfång, *Gestures* som låter HoloLensens sensorer tolka användares handrörelser för att interagera med simuleringen, samt *Spatial Mapping* för att låta simuleringen interagera med omgivningen. Med hjälp av dessa designades gränssnittet så att användaren med en "klickning" med pekfingret kan markera roboten, och sedan placera den på valfri plats i glasögonens blickfång, till exempel på ett bord eller på golvet.

3.6 Fysikalisk Simulering

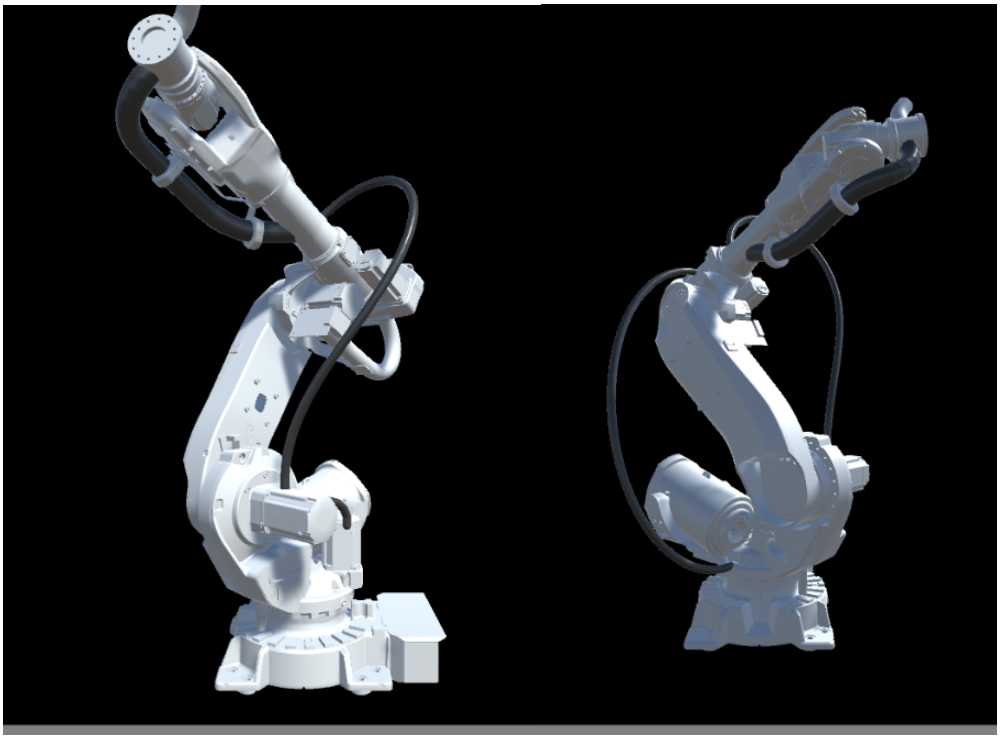
För att även uppnå målet om att utföra en mer utförlig fysikalisk simulering konstruerades även en applikation med AGX-komponenter. Då AGX Unity visat sig vara inkompatibel med materialdata från Process Simulate fick en tidigare version utan materialdata användas för denna applikation, som ses i Figur 9. Här simulerades även kablage. AGX har stöd för att utvinna kraftdata ur dessa kablagesimuleringar, men detta har ännu inte implementerats i AGX Unity, varför det



Figur 8: Grafisk representation av hur Unity (vänster) hanterar hinges och hur skelettstrukturen är uppbyggd baserat på AGX-hinges (höger). Då AGX har ett mer flexibelt system för att definiera rotationsaxel för dess leder ledde detta till en struktur som verkade intuitiv med AGX leder men som visade sig svårarbetad med Unitys.

uteblev ur det färdiga applikationen.

Lederna i AGX Unity flera kontrollmekanismer för att styra och begränsa rörelse. Exempelvis finns *Speed Controller* som ger leden en bestämd hastighet, eller en *Electric Motor Controller*, som simulerar en elmotor. Detta är fördelaktigt till exempel vid manuell styrning av en led. För simuleringens syfte visade sig den bäst lämpade vara *Lock Controller*. Den fungerar genom att den låser leden till en i radianer given vinkel. Hur leden rör sig från en vinkel till nästa kan styras med hjälp av så kallad *Compliance*, *Damping* och *Force Range*. De första två styr helt enkelt hur benägen leden är att röra sig mot sitt mål, samt hur stort motstånd den stöter på. Den sista bestämmer kraften som måste appliceras på leden för att denna ska brytas.



Figur 9: Tidig version av en robot utan materialdata, med AGX-fysik och kablage, sett från två vinklar.

4 Utvärdering

Arbetet utvärderas i termer av modellering, simulering och visualisering utifrån de krav som ställdes upp i Avsnitt 2.4. Utvärderingen syftar till ifall mål och krav har uppnåtts, till vilken grad, vilka åtgärder som vidtagits istället om dessa varit ouppnåeliga samt varför de inte kunde uppnås.

Sammanfattningvis har samtliga av de grundläggande kraven uppnåtts, där en lyckad projicering av en rörlig robot har genomförts med vinkeldata från Process Simulate. Denna var även interaktiv i det att roboten gick att förflytta i rummet med en handgest.

4.1 Modellering och simulering

Den importerade robotcellens modell innehöll från början cirka sju miljoner hörn och Unitys rekommendation är att hålla sig under 200 000 för PC-applikationer och 100 000 för mobilapplikationer [24]. Unity har inga specifika rekommendationer när det kommer till hur stora modeller som kan användas till HoloLens men de rekommenderar att man håller sig till modeller som kan visas med en uppdateringsfrekvens på 60 bilder per sekund [25]. Dessutom finns det en övre gräns på hur mycket minne en app får ta upp, 900 MB [25]. Vid användande av hela cellen överskreds detta mer än fem gånger om, vilket gjorde applikationen omöjlig att visa i HoloLens. Även cellens givna materialdata som importerades tillsammans med modellen har viss inverkan på prestandan då det finns en stor mängd olika material i modellen. Delar av samma material som inte rör sig oberoende av varandra kan med hjälp av så kallad “batching” renderas tillsammans, vilket reducerar antalet anrop som CPU:n behöver skicka vidare till GPU:n, och därmed minskar renderingstiden [26].

För att uppnå rimlig prestanda för simuleringen i termer av uppdateringsfrekvens, latens och responstid till HoloLensen behövde modellerna skalas ner avsevärt. Många detaljer i CAD-modellerna som är relevanta ur en teknisk synvinkel, som till exempel muttrar, kullager och andra interna eller små komponenter, bidrar inte till en kinematisk simulering av robotar. Dessa togs därför bort. Vidare kunde endast en robot åt gången köras, då att inkludera fler än en robot i scenen återigen ledde till för många komplexa objekt. För att få med fler bakgrundsföremål i visualiseringen skulle så kallad “static batching” kunna användas för att rendera

Figur 10: Roboten presenterad i HoloLens, placerad på ett objekt i verkligheten.

alla statiska föremål. Detta fungerar genom att de statiska modellerna kombineras till ett enda stort mesh som kan renderas betydligt snabbare [26]. Nackdelen med detta är att det krävs stora mängder minne vilket skulle innebära att applikationen fort går över minnesgränsen för HoloLens [26].

Gällande versionen med AGX kunde fler detaljer tas med i simuleringen, då det finns tillgång till kraftfullare hårdvara. Den mer avancerade fysikmotorn tillät även en mer djupgående simulering av dynamiska ledrörelser istället för kinematiska, samt att kablage kunde simuleras. I AGX Dynamics finns funktioner för att utläsa krafter ut kablaget som simuleras, men då AGX Unity befinner sig i ett tidigt utvecklingsstadium så har detta inte implementerats än. Därmed är kablaget i huvudsak demonstrativt.

4.2 Visualisering

Till följd av inkompabiliteten mellan AGX Unity och HoloLens visualiserades simuleringen på två olika sätt. För versionen till HoloLens blev interaktivitetsnivån relativt begränsad. Ett visst gränssnitt för att styra simuleringen kan utformas, men visualiseringen fick i huvudsak styras genom att användaren fysiskt förflyttar sig i rummet, eller flyttar på objektet som visualiseras. Visualiseringen i HoloLensen blev också lidande av det begränsade synfältet. Modellen kunde endast observeras styckvis, trots att den skalats ner till att vara ungefär en halvmeter hög, om den inte observerades på långt håll. Detta begränsade övertagen med tekniken, då det var svårt att få en känsla av skala när man endast såg segment av den simulerade modellen.

AGX-applikationen visualiserades på en vanlig datorskärm. Detta saknade många av övertagen som AR-visualisering erbjuder, men till följd av att programmet konstruerades från grunden kunde interaktiviteten skräddarsys i högre mån. I det här fallet valdes att ha en fritt rörlig kamera, vilket låter användaren undersöka scenen ur alla tänkbara perspektiv. Vidare har man tillgång till att använda mus och tangentbord för att uppnå i stort sett godtycklig nivå av kontroll över både simulering och visualisering.

4.3 Uppfyllnad av krav

Sett till kraven som specificerades i Avsnitt 2.4 så ser man att flera eftergifter har fått göras. Materialdata kunde bevaras för HoloLens-applikationen, men inte till AGX. I båda fallen fick cellen reduceras till en robot, men i AGX-applikationen var detta en fullskalig robot och inte en komprimerad och förenklad version. En fysikalisk simulering kunde genomföras i förhållandevis stor omfattning, men då endast en robot simulerades kunde vissa funktioner så som kollisionsdetektering inte demonstreras.

Tabell 2: Utvärdering av hur väl kraven och målen som specificerades i Tabell 1 uppfylldes. Under Resultat-kolonnen kan man se vad som uppnåtts inom de olika kategorierna.

Utvärdering av krav och mål		
Krav	Specifikation	Resultat
Fungerande app med robot i HoloLens	Projicera robot	Lyckad projicering av robot i HoloLens
	Animerad robot	Lyckad animering med vinkeldata från Process Simulate
	Interaktion med objekt	Lyckad interaktion, roboten kan flyttas runt och placeras på olika ytor i det verkliga rummet
Mål	Specifikation	Resultat
Fullständig modell	Alla robotar samt vridbord	Inte möjligt då Unity inte klarar av att hantera så stora modeller
	Omliggande miljö	Inte möjligt av samma anledning som ovan
Kablage	Animerat kablage	Kablage animerat i AGX-varianten av simuleringen, inte möjligt i HoloLens då AGX är inkompatibel med HoloLens
	Kraftberäkningar	Inte möjligt på grund av begränsningar hos AGX
Interaktivitet	Styrning av simulation	Lyckad styrning med hjälp av knappar som startar, stannar eller återställer roboten.

5 Diskussion

System som detta där flera avancerade och tunga komponenter används tillsammans för att förbättra upplevelsen och förståelsen är under utveckling. Man måste också konstatera att dessa system inte är en uppfinning utan en evolutionär process som kontinuerligt drivs framåt med framsteg inom mindre områden. Här kommer det att diskuteras hur resultaten i detta projekt platsar in i den kontinuerliga utvecklingen. Diskussionen kommer även behandla vilka krav som finns på utveckling för att alla mål i projektet skall gå att uppfylla.

5.1 Fysikalisk simulering

Den fysikaliska simuleringen har fått en begränsad roll i arbetet till följd av att insticksmodulen AGX Unity befinner sig i ett tidigt utvecklingsskede och därmed saknar en stor mängd funktionalitet. De bitar som fungerar ter sig dock lovande för framtida utveckling. Att använda AGX HingeJoints var till exempel mycket enklare under arbetets gång än att använda någon av de metoder som testats med enbart Unitys inbyggda funktionalitet. Möjligheten att simulera kablage och att kunna utvärdera kraftdata från dessa är också något som skulle kunna ha stora praktiska tillämpningar, så som tillexempel planerat underhåll, när detta väl implementerats.

5.2 AR som visualiseringsmedium

AR har stor potential som visualiseringsmedel. Förmågan att som användare kunna röra sig i rummet ger enastående möjligheter att ta till sig ett helhetsintryck av modellen. Tyvärr lider tekniken av en rad barnsjukdomar, både inom hårdvara och mjukvara. Först och främst är synfältet oerhört begränsat, vilket i hög grad negerar den största fördelen med tekniken. Detta är så klart något som kommer förbättras med tiden när paneltillverkningsmetoderna har mognat och större displayer med tillräcklig pixeltäthet kan tillverkas till ett rimligt pris. Som resultat kräver detta i sin tur att beräkningskraften i glasögonen ökar markant, då denna redan idag är en väldigt begränsande faktor. I dagsläget går detta att ta sig runt med hjälp av tekniken Holographic Remoting. Denna teknik innebär att programmet exekveras på en kraftfullare dator, som sedan strömmar bilden över en WiFi-uppkoppling till glasögonen. Detta begränsar dock glasögonens mobilitet, som i övrigt är små och

enkla att ta med sig [27].

Gällande mer detaljerad information från simuleringar, som kräver representation i form av siffror, grafer och text, har HoloLensen också en rad begränsningar. Det faktum att AR av sin natur är additiv till omgivningen kan den till exempel inte representera äkta svärtor, vilket kan leda till svaga kontraster i ljusa rum [28]. Detta kan göra det ansträngande för användaren att läsa text och siffror. Det kan även vara svårt ur ett designfilosofiskt perspektiv att hitta en plats där det är effektivt att placera texter och dylikt för att de ska vara enkla för användaren att ta till sig, särskilt med det begränsade synfält som glasögonen idag lider av.

5.3 Prestandabegränsningar i modell

Vid bearbetningen av modellen upplevdes i flera steg stora prestandabegränsningar. Att öppna den exporterade .jt-filen i ett för detta avsett program skedde förhållandevis smärtfritt, men efter export från detta till mer konventionella 3D-format så uppstod snart stora problem i bearbetning av modellen, då den blev oerhört trögkörd. Även efter stor kompression vid import till Unity var det stor latens vid grafisk bearbetning av cellen. En möjlig anledning till detta skulle potentiellt kunna vara att Unity inte är designat för att hantera modeller med så stor detaljrikedom som förekommer i ingenjörsmässiga CAD-modeller, utan snarare är byggt för enklare mesh-strukturer med högupplösta texturer på för att ge illusionen av detaljrikedom.

5.4 Eliminering av “handpåläggning”

För att genomföra simuleringen i dagsläget krävs stora mängder manuellt arbete. Modeller behövde exporteras från PS i flera steg, dessa modeller behövde behandlas för att applicera leder och övrig fysikalisk interaktivitet, och slutligen måste script skrivas för att översätta rörelsedata från PS till ett format som kan användas i Unity. En mer detaljerad beskrivning av processen för att kunna få en ny cell att fungera i de nu designade programmen finns i Appendix A.

Den tänkta lösningen på detta är ett standardiserat gränssnitt som innehåller all denna information från början, alltså geometrier, leddata, rörelsedata samt eventuella övriga fysikaliska komponenter. Hur detta implementeras är inte klart i nuläget

men antingen måste det stödjas direkt av Unity eller så måste ett program designas för att översätta denna modell till en prefab för Unity. Utifrån de program som tagits fram i detta projektet kan det konstateras att detta gränssnitt måste åtminstone innehålla:

- Geometrier för modellen.
- Specifikationer för de leder som finns i modellen.
- Materialdata.
- Indata för styrning av modellerna, exempelvis från ett styrsystem.
- Utdata från censorer i modellen, exempelvis till ett styrsystem.
- Övrig data som skall visualiseras.

Detta kan jämföras med FMI, *Functional Mock-up Interface*, som är ett gränssnitt som har utvecklats sedan 2010 som strävar efter att binda samman olika simuleringsprogramvaror [29]. Om alla komponenter i ett traditionellt system ska kommunicera med varandra så ökar antalet gränssnitt exponentiellt med antalet komponenter. FMI tar sig runt detta genom att alla komponenter istället kommunicerar genom en central hub, och därmed bara behöver ett gränssnitt.

6 Slutsatser

Sammanfattningsvis kan det konstateras att rörande modellen så kommer den behöva omarbetas så att den blir enklare för programmen att hantera. Som det ser ut nu tynger mängden delar och detaljer ner prestandan till en nivå där det blir svårt att jobba med modellen även med bra hårdvara. Detta bör vara möjligt då många delar i cellen inte fyller något visuellt syfte då de befinner sig inne i modellen. Detaljnivån på modellerna skulle också kunna sänkas, men då med viss estetisk förlust.

Att simulera cellen med AGX Dynamics som fysikmotor fungerar bra och öppnar upp för flera möjligheter. Framförallt kommer det kunna bli ett kraftigt verktyg då det är mer välutvecklat och fler funktioner är implementerade i AGX Unity. Projektet har visat att det i nuläget finns vissa direkta problem med metoden, men att den samtidigt redan erbjuder flera bra funktioner som bra kontroll över leder och enkel implementation av slangar.

Att visualisera produktionscellen med HoloLens har visat sig vara ett kraftigt verktyg. Det öppnar upp möjligheter att hantera cellen på ett sätt som tidigare inte varit möjligt då den nu kan visualiseras i vilken miljö som helst. Samma program kan användas för att testa cellen inför produktion i naturlig skala i en fabrik, som används på ett skrivbord för att visa en kund hur produktionen går till. Tillämpningarna kommer troligtvis vara många om man ger anställda möjlighet att jobba med denna utrustning då nyttoområderna kommer träda fram för de insatta användarna. HoloLensen i sig erbjuder både många möjligheter utöver den rena AR-upplevelsen, som gester och röststyrning vilka möjliggör utveckling av smidiga verktyg för användaren. Dock måste man konstatera att HoloLens är ett steg i utvecklingen mot en mer komplett AR-upplevelse och har vissa ordentliga begränsningar i hur väl de fungerar.

Slutsatsen som kan dras kring systemet som helhet är att det idag inte går att bygga ett system för simulering och visualisering med dessa komponenter. Delarna var för sig är välutvecklande men ett visst arbete måste ske för att systemet skall kunna sättas samman. Det är framförallt AGX Unity som är den felande länken som hittades i detta projekt. Vidare slutsatser om kritiska brister kan inte dras mer än att vissa prestandaproblem finns med stora modeller.

Resultatet visar också att detta är ett område som det är värt att arbeta vidare på,

dels genom att titta på andra möjligheter att uppnå samma sak, men också med samma metod då komponenterna utvecklats vidare. Möjligheterna som öppnas med dessa verktyg är stora och nyfikenheten för denna och liknande tekniker är stora inom många områden.

Referenser

[1] Siemens, "Process simulate," https://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-simulation/assembly/process-simulate.shtml, hämtad: 2017-05-12.

[2] Algoryx, "About Algoryx," https://www.algoryx.se/company/about-algoryx/, hämtad: 2017-03-31.

[3] C. V. Breakfield and R. E. Burkey, "Introduction," in *Managing Systems Migrations and Upgrades: Demystifying the Technology Puzzle*. Digital Press [Imprint], 2002, tillgänglig som ebok: http://dx.doi.org.proxy.lib.chalmers.se/10.1016/B978-155558256-2/50001-6.

[4] A. B. Craig, *Understanding Augmented Reality: Concepts and Applications*. Morgan Kaufmann, april 2013.

[5] SAAB, "Aviguide head up display," http://saab.com/air/avionics-systems/avionics-equipment/aviguide-aircraft/, hämtad 2017-01-27.

[6] W. Wilhelmsson, "Hyundai tar fram augmented reality-manual till sina bilar," http://feber.se/bil/art/338458/hyundai_tar_fram_augmented_rea/, november 2015, hämtad: 2017-01-27.

[7] N. Reiff, "How pokemon go uses augmented reality (ntdoy)," http://www.investopedia.com/articles/investing/073016/how-pokemon-go-uses-augmented-reality-ntdoy.asp, juli 2016, hämtad: 2017-01-27.

[8] L. Prasuethsut, "Htc vive: Everything you need to know about the steam vr headset," https://www.wareable.com/vr/htc-vive-vr-headset-release-date-price-specs-7929, hämtad: 2017-05-12.

[9] S. L. S. . M. Andronico, "What is occulus rift?" http://www.tomsguide.com/us/what-is-oculus-rift,news-18026.html, hämtad: 2017-05-12.

[10] J. vince, *Introduction to Virtual Reality*. Gray Publishing, Mars 2004.

[11] Microsoft, "Mixed reality," https://developer.microsoft.com/en-us/windows/mixed-reality/mixed_reality, hämtad: 2017-05-07.

[12] T. P. G. D. Team, https://www.nianticlabs.com/blog/pokemon-go-first-look/, Mars 2016, hämtad: 2017-05-11.

[13] T. Lappin, https://www.flickr.com/photos/telstar/4136242/, 2004, hämtad: 2017-05-11.

[14] S. Shaaban, "Techniques available to manage your production line," http://www.manmonthly.com.au/news/techniques-available-to-manage-your-production-line/, april 2008, hämtad: 2017-01-27.

[15] Microsoft, "Mixed reality - spatial mapping," https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_mapping, hämtad: 2017-05-7.

[16] ——, "Hololens hardware details," https://developer.microsoft.com/en-us/windows/holographic/hololens_hardware_details, hämtad: 2017-02-06.

[17] Algoryx, "Agx dynamics," https://www.algoryx.se/products/agx-dynamics/, hämtad: 2017-02-04.

[18] Siemens, "Jt file format reference," http://www.plm.automation.siemens.com/en_us/Images/JT-v10-file-format-reference-rev-B_tcm1023-233786.pdf.

[19] Unity, "Unity scripting api: Monobehaviour," https://docs.unity3d.com/ScriptReference/MonoBehaviour.html, hämtad: 2017-05-11.

[20] ——, "Unity scripting api: Monobehaviour.update()," https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html, hämtad: 2017-05-11.

[21] ——, "Unity scripting api: Monobehaviour.fixedupdate()," https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html, hämtad: 2017-05-11.

[22] U. Technologies, "Jsonutility," https://docs.unity3d.com/ScriptReference/JsonUtility.html, hämtad: 2017-04-28.

[23] P. Renner and T. Pfeiffer, "Attention guiding techniques using peripheral vision and eye tracking for feedback in augmented-reality-based assistance systems," in *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 2017, pp. 186–194.

[24] Unity, “Unity - manual,” https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html, hämtad: 2017-05-10.

[25] Microsoft, “Performance recommendations,” https://developer.microsoft.com/en-us/windows/holographic/performance_recommendations, hämtad: 2017-01-31.

[26] Unity, “Unity - manual,” https://docs.unity3d.com/Manual/DrawCallBatching.html, hämtad: 2017-05-10.

[27] Microsoft, “Add holographic remoting,” https://developer.microsoft.com/en-us/windows/holographic/ad_holographic_remoting, hämtad: 2017-01-29.

[28] ——, “Color design,” https://developer.microsoft.com/en-us/windows/mixed-reality/color_design, hämtad: 2017-05-07.

[29] “Fmi, functional mock-up interface,” https://www.fmi-standard.org/, hämtad: 2017-05-07.

A Steg för att sätta upp en ny cell

Detta är de steg som behöver genomgås för att kunna få en ny cell att fungera. Det är i dagsläget mycket handpåläggning som krävs och alla detaljer kan inte beskrivas här, med grundförståelse för programvaran borde det dock inte vara några problem att göra detta.

1. Typkonvertera modellen till ett format som stödjs av Unity, i detta projektet användes .obj.

2. Importera modellen till Unity. Här kan det bli nödvändigt att sortera hierarkin i cellen om den inte följer med modellfilen.

3. Skapa ett skelett för alla rörliga delar. Detta realiseras med två rätblock på vardera sida om varje led.

4. Lägg json-fil med vinkeldata, byggd enligt strukturen i Sektion 3.4, med namnet *simulation-asList.json* i StreamingAssets-mappen i Assets.

5. För att få rörelserna realiseras detta på två olika sätt för AGX-komponenter och Unity-komponenter.
   AGX:

   (a) Associera hinges mellan varje par av skelettbitar som ska rotera relativt varandra.

   (b) Lägg till det AutomaticHingeController script som är anpassat för AGX på varje robot/vridbord och associera varje hinge med en plats i den hinge-array som finns i scriptet. Välj vilken av enheterna som skall simuleras med en siffra 1 till 4 i fältet *Choose Device* i scripthanteraren.

   (c) *Valfri:* Lägg till det prefab innehållandes knappar och associera knapparna med metoderna *RunSimulation, StopSimulation* och *ResetSimulation* i varje AutomaticHingeController-script. Görs inte detta måste man se till att scriptet är skrivet så att det startar automatiskt och inte vid knapptryck.

   (d) Ersätt modellens kablage med AGX-kablage.

   (e) Bygg ett program till 64-bitars PC.

Unity:

(a) Strukturera om hierarkin så att allt som ska röra sig med en skelettbit ligger under denna skelettbit.

(b) Applicera det AutomaticHingeController-script som är anpassat för Unity-komponenter på varje robot/vridbord i cellen samt associera varje skelettkomponent med motsvarande komponent på andra sidan leden.

(c) Bygg till Visual Studio

(d) Distribuera till HoloLens.

B Guide för att göra en Unity-scen redo för HoloLens

För att kunna köra en applikation från Unity i HoloLens måste dessa steg följas.

1. Hämta en mapp med färdiga Assets (ännu ej sammanställd) och lägg den i Assets-mappen under ditt projekt.

2. Ta bort *Main Camera* och *Directional Lights* som finns som default i hirearkin.

3. Dra in objekten *Main Camera*, *Lights*, *Cursor*, *Spatial Mapping* och *Collection* från mappen prefabs under *Resources*.

4. **Cursor:**

   (a) Dra in scriptet *WorldCursor* till moderobjektet *Cursor*

   (b) Dra in materialet *CursorMaterial* till dotterobjektet *CursorMesh*

   (c) Dra in meshet *Cursor* till dotterobjektet *CursorMesh*

5. **Spatial Mapping:**

   (a) Dra in scriptet *SpatialMapping* till objektet *Spatial Mapping* i hirearkin.

   (b) Klicka på den lilla cirkeln efter rutan vid *Draw Material* i inspektorn,

sök där upp *SpatialMappingWireframe* och välj detta

(c) Klicka i *Draw Visual Meshes* i inspektorn.

6. **Collection:**

   (a) Dra in scripten *GazeGestureManager* och *SpeechManager* till moderobjektet *Collection*

   (b) Dra in scriptet *TapToPlaceParent* till dotterobjektet *Scene*

   (c) Lägg samtliga objekt tillhörande projektet som dotterobjekt till *Scene*.

Övriga inställningar:

1. Välj *Edit* > *Project Settings* > *Quality*, klicka sedan på default-pilen under Windows-storeloggan och välj *fastest* så att den grönmarkeras i listan.

2. Välj *File* > *Build Settings*

   (a) Välj *Windows Store* i platformlistan.

   (b) Sätt SDK till *Universal 10*

   (c) Sätt target device till *HoloLens*

   (d) Sätt Build Type till *D3D*

   (e) Klicka i *Unity* `C#` *Projects* under debugging

3. Öppna Player Settings

   (a) Välj inställningar för Windows Store (Klicka på loggan)

   (b) Klicka på Other Settings

   (c) Bocka i *Virtual Reality Supported* och se till att *Windows Holographic* är vald som Reality SDKs

Bygga Appen:

1. Välj *File* > *Build Settings*

2. Klicka på *Add open scenes*

3. Klicka på *Build* och skapa en ny mapp som heter *App* (om det är första gången appen byggs, annars se till att rätt mapp är vald så den bara uppdaterar den befintliga appen.

Debugga/releasa appen på HoloLens:

1. Öppna appen i Visual Studio från mappen som heter App

2. Välj *Debug* om du bara vill testa appen och Release om du vill spara appen till HoloLens-enheten

3. Välj *x86* (istället för arm) i dropdownmenyn i vektygsfältet

4. Välj *Device* i nästa dropdownmeny om du använder sladden och *Remote Machine* om du använder WiFi

5. Om datorn inte tidigare använts med HoloLens-enheten kommer de behöva paras ihop

   (a) Med sladden behöver du ta fram en pinkod under *Settings > Update & Security > For Developers > Pair*

   (b) med WiFi behöver du ta fram HoloLens-enhetens *IPv4-adress* under *Settings > Network & Internet > Advanced Options*

6. Slutligen väljs *Start without debugging* under *Debug*