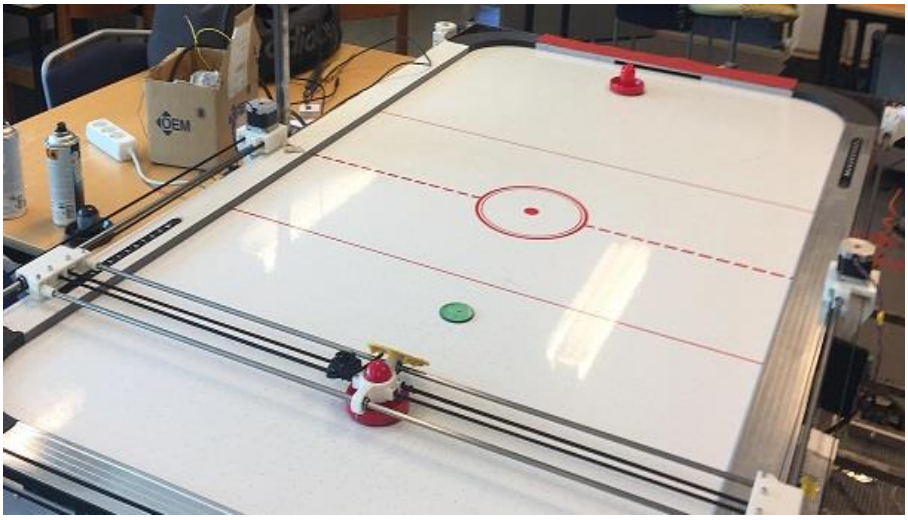
CHALMERS

KANDIDATARBETESRAPPORT



Självspelande Air-hockeyspel
Med maskininlärning i centrum

HAMIDREZA AMININIA, HANNES NILSSON VAN RENSWOUDE & NICKLAS PETTERSSON

Kandidatprojekt
Institutionen för signaler och system
Chalmers tekniska högskola
Göteborg, 2017-06-09 SSYX02-17-04

Sammanfattning

Den här rapporten beskriver en undersökning angående om maskininlärning kan appliceras på en robot för att skapa ett självspelande air-hockeyspel. Roboten ska kunna spela lika bra som den givna indatan och spela mot en mänsklig motståndare, som en jämbördig motståndare.

Den erhållna hårdvaran och en stor del av mjukvaran i projektet är skapad av föregående projektgrupper, vilket leder till att huvudfokus för projektet ligger på att samla indata och konstruktionen av en algoritm. Konstruktion av algoritmen sker i Matlab, som sedan ska kommunicera med Arduino som styr motorerna.

Projektet har resulterat i en robot som kan efterlikna den simulerade utdatan av algoritmen på en grundlig nivå. Roboten kan däremot inte skjuta iväg pucken, eller blockera puckar över 0.5m/s, då kommunikationen mellan Matlab och Arduino är för långsam. Projektet har på så vis inte uppfyllt de givna praktiska kraven, däremot fungerar roboten i teorin.

Abstract

This report describes an investigation as to whether machine learning can be applied to a robot to create an autonomous air-hockey game. The robot should be able to play as well as the given input and play against a human opponent, as an equal opponent.

The used hardware and a large part of the software in the project were created by previous project groups, which means that the main focus of the project is on collecting input data and the construction of an algorithm. Construction of the algorithm takes place in Matlab, which will then communicate with Arduino that controls the motors.

The project has resulted in a robot that can mimic the simulated output of the algorithm on a basic level. However, the robot can not shoot, or block pucks with a velocity above 0.5m/s, since communication between Matlab and Arduino is too slow. The project has thus not fulfilled the given practical requirements, however the robot works in theory.

Förord

Vi skulle vilja tacka följande personer för hjälpen de givit oss under projektets gång.

Jonas Fredriksson, för att ha varit vår handledare och för att ha hjälpt oss med många problem.

Klas Sandell, för att ha tagit sig tiden att förklara det projektet som lägger grunden för vårat.

Lukas Mared, för att ha givit information och insikt hur projektet gick till föregående år.

Instutionen för signaler och system på Chalmers tekniska högskola, för att ha möjliggjort genomförandet av vårt projekt.

Innehåll

Kapitel 1

Introduktion

En teknik som är på frammarsch idag är maskininlärning. Maskiner som är programmerade att utföra speciella uppgifter genom att lära sig utav erfarenhet. Genom att utnyttja maskininlärning går det att analysera extremt stora mängder indata, vilket är värdefullt inom en rad områden. Maskininlärning innebär i korthet att en maskin kan lära sig utföra en handling genom att följa en algoritm bestående av indata istället för att följa en strikt programmeringskod. Inom sjukvården utvecklas tekniken för att förbättra diagnostiken eller för att analysera röntgenbilder som kan avslöja exempelvis cancer. Ett annat exempel som är aktuellt idag är självkörande bilar. Med hjälp av en kamera så kan en dator observera en bilförares rattrörelser vid en viss vinkel och hastighet på vägen. Datorn kan sedan själv avgöra genom erfarenhet från observationer hur mycket ratten ska vridas, och därmed kan en bil köras av sig själv. Trots att maskininlärning har existerat i flera decennier upptäckts ständigt nya användningsområden som tekniken kan appliceras på, dessutom är tekniken en central del inom artificiell intelligens, alltså när datorer förses med något som kan liknas mänsklig intelligens [1].

Det finns olika tekniker inom området, de vanligaste teknikerna som används idag bygger på något som kallas Supervised learning och Unsupervised learning [2]. I Supervised learning tränas datorn med många exemplar på indata och förväntad utdata, men i Unsupervised learning får datorn ingen information om önskat utfall, utan får själv hitta lämpliga sätt att skapa en struktur i informationen den matas med.

Under ett antal år har studenter på Chalmers Tekniska Högskola arbetat med att utveckla en robot som kan ersätta en mänsklig air-hockey-spelare, och i förlängningen, konstruera ett helt autonomt air-hockey-bord. Arbetet har resulterat i en robot som kan försvara målet mot skott och skjuta iväg puckar, dock finns det stora brister. Roboten kan endast blockera puckar som färdas i lägre hastigheter och robotens offensiva spel är nästintill obefintligt. Med andra ord kan inte roboten ersätta en mänsklig air hockey-spelare på ett tillfredsställande sätt [3].

Det här projektet går ut på att undersöka om den ovannämnda roboten kan lära sig spela air-hockey genom maskininlärning. Ett lyckat resultat medför i förlängningen att maskininlärning har potentialen att appliceras på liknande spel och hobbyprodukter, vilket innebär ett helt nytt användningsområde för maskininlärning.

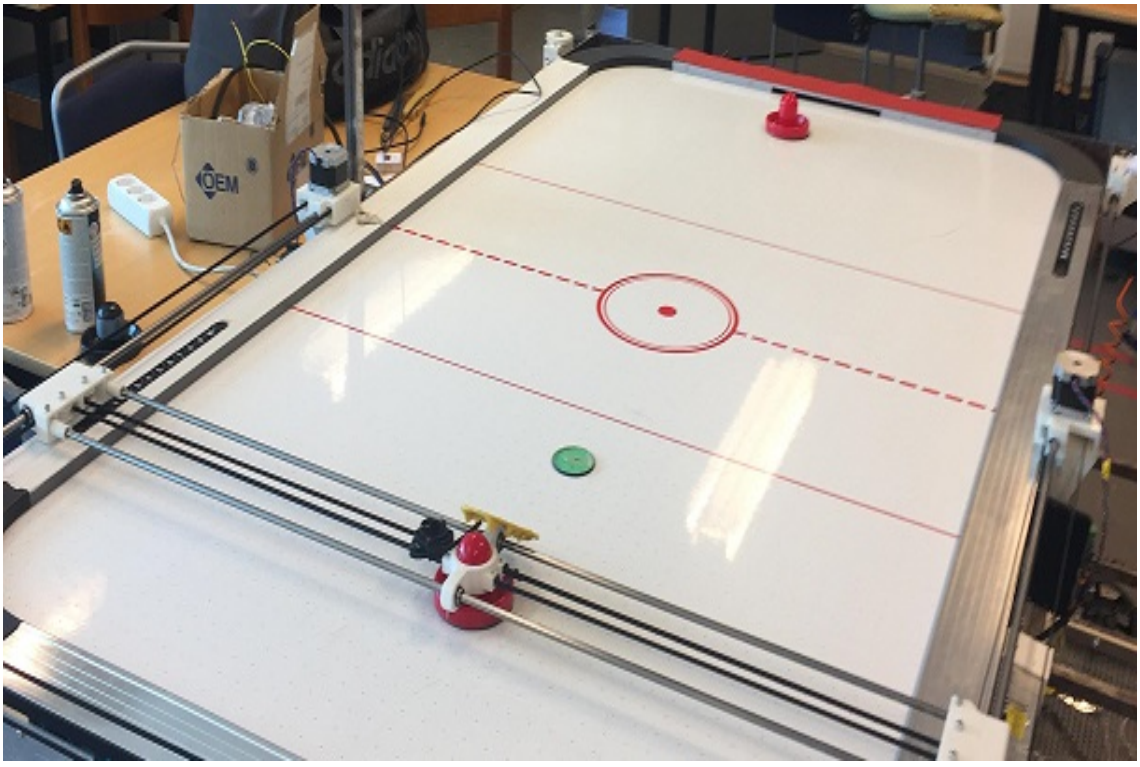
1.1 Syfte

Syftet med det här projektet är att undersöka om en robot kan ersätta en mänsklig air-hockeyspelare genom att applicera maskininlärning. Roboten ska kunna efterlikna den simulerade utdatan från algoritmen och vara autonom.

1.2 Problembeskrivning

Under fjolårets kandidatarbete gjordes ett försök att utveckla ett autonomt air-hockeyspel [3]. Arbetet resulterade i en konstruktion som kunde identifiera pucken, beräkna dess hastighet och förutse dess färdriktning och förflytta klubban därefter för att slå tillbaka pucken. Roboten klarade av att försvara målet mot kommande puckar i hastigheter upp till 7 m/s och kunde skjuta en stillastående puck med en hastighet på 1.7 m/s. Roboten klarade inte av att spela offensivt vid höga hastigheter på egen hand, då det skulle innebära mer komplicerade beräkningar, vilket systemet inte klarade av. Systemet gjorde då felaktiga beräkningar, vilket resulterade i en defensiv robot.

Detta projektets fokus ligger på att undersöka om förra årets arbete kan förbättras genom att använda maskininlärning, vilket innefattar att utveckla en algoritm som implementeras i roboten och att samla indata som implementeras i algoritmen. För att svara på frågan huruvida maskininlärning är en bättre metod för problemet måste projektets resultat jämföras med föregående projekts resultat. Ur ett ingenjörsmässigt perspektiv är detta ett programmeringstungt och utmanande projekt. Se figur 1.1 för övergripande bild.



Figur 1.1: Air-hockeybordet

Systemet måste innehålla en visuell del, i det här fallet en kamera, som har överblick över hela bordet och som kan detektera pucken och de två klubborna. Då air-hockey har ett högt tempo och kräver relativt hög precision så måste frekvensupplösningen och antalet bildrutor per sekund på kameran vara tillräckligt höga. För att kunna samla in all nödvändig data så måste kameran kunna identifiera puckens och klubbornas positioner, det måste även finnas en möjlighet att räkna ut puckens hastighet.

Algoritmen som ska utvecklas måste kunna hantera mycket indata och olika typer av indata, samtidigt måste den arbeta relativt snabbt. Den ska även kunna avgöra vilken indata som är bra och bör användas och sortera ut ogynnsamma indata. Processorn som algoritmen implementeras i ska även den kunna arbeta tillräckligt snabbt. Ett försök att göra roboten användarvänlig ska genomföras, vilket innebär att användarvänligheten även ska beaktas vid val av processor, det är dock inte första prioritet.

1.3 Krav och mål

Nedan listas övergripande krav och mål för utveckling av projektet medans en mer detaljerad kravspecifikation finns i figur A.1.

1.3.1 Krav

Kraven är baserade på utförda tester. Utifrån testerna framgick det att nedanstående krav är rimliga att sätta på projektet för att få ett grundläggande spel.

- (K1) Roboten skall kunna träffa pucken på hela sin planhalva.
- (K2) Klubbans rörelse skall kunna vara snarlik den simulerade utdatan.
- (K3) Systemet skall kunna försvara målet mot en puck som kommer i 6 m/s.
- (K4) Roboten skall kunna slå tillbaka pucken när den kommer med en hastighet av minst 3 m/s.

1.3.2 Mål

Målen är framtagna för att uppnå ett fungerande spel mellan en robot och en mänsklig spelare på nybörjarnivå. Om målen inte uppfylls kommer försöket att implementera ett offensivt spel ha misslyckats.

- (M1) Roboten skall kunna skjuta iväg pucken när hastigheten på pucken är 5 m/s.
- (M2) Roboten skall kunna försvara målet när puckens hastighet är 8 m/s.
- (M3) Roboten skall kunna stå för minst 10% av målen i en match.
- (M4) Systemet skall vara lätt att använda för en utomstående person.

1.4 Avgränsningar

Begränsad tid och resurser leder till vissa begränsningar inom projektet.

- Projektet förutsätter att den mekaniska hårdvaran använd av tidigare projekt är den samma och fungerar [3].
- Maskininlärnings algoritmen förutsätter att föregående projekts mjukvara fungerar.
- Roboten är anpassad för det standard air-hockeybord som används under projektets gång.
- Mänsklig interaktion måste tillämpas i situationer som att lägga upp pucken på bordet.

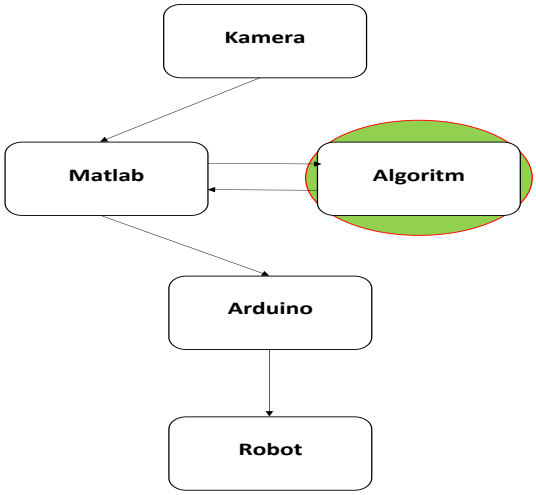
1.5 Upplägg av rapport

Rapporten beskriver systemet i kapitel två. Där beskrivs hur projektet interagerar med det ärvda systemet. Kapitel tre behandlar teorin bakom maskininlärning och ger läsaren en djupare förståelse i det behandlade ämnet. Följande kapitel beskriver hur algoritmen är utvecklad och konstruerad. Det tar även upp hur data insamlas och hur algoritmen kan vidareutvecklas. Validering av krav och mål redovisas i kapitel fem följt av diskussion och slutsatser i kapitel sex respektive sju dragna av resultatvalideringen.

Kapitel 2

Systembeskrivning

Systemet som används i projektet är två tidigare projektgruppers samlade arbete för att skapa ett självspelande air-hockeyspel. Det består av hårdvaru- och mjukvarukomponenter, där de två tidigare grupperna bidragit till båda aspekterna av systemet. Figur 2.1 visar hur systemet är uppbyggt och vad det här kandidatarbete kommer att fokusera på. Projektet kommer fokusera på att implementera en algoritm som interagerar med Matlab, som sedan skickar vidare utdatan av algoritmen till Arduino som styr roboten.



Figur 2.1: Övergripande bild på systemet.

2.1 Hårdvara

På ena sidan av air-hockeybordet kan klubban röra sig längs x och y-axeln med hjälp av stålstavar som hålls ihop av 3D tryckta plastfästen. Tre stegmotorer möjliggör rörelserna i båda leden och är fästa i plastfästena. En stegmotor för att förflytta klubban i y-led och två kraftigare stegmotorer för att förflytta klubban i x-led. Motorerna måste vara kraftfullare i x-led då större vikt ska kunna förflyttas i det ledet. En metallaxel mellan motorerna bidrar till att motorerna är synkroniserade med varandra.

En kamera är placerad på ett stativ vid en exakt höjd ovanför air-hockeybordet. Kameran kopplas in i en dator, där olika bildbehandlingsmetoder sker i Matlab. Med hjälp av kameran och bildbehandlingskoden kan olika uträkningar gällande puck och klubba göras.

En Arduino Mega microcontroller används för att processera all data och två Arduino Uno används för att styra stegmotorerna så att klubban kan förflytta sig till önskad position. Arduino får positionerna från utdatan av maskininlärnings algoritmen. Kommunikationen mellan Matlab och Arduino är en envägskommunikation och sker via en vanlig USB-port.

2.2 Mjukvara

Majoriteten av programmering sker i Matlab. Den skrivna koden som används är framförallt bildhanteringskod. Koden är skriven så att kameran kan upptäcka gul och grön färg, där klubban representeras av gul färg och pucken av grön färg. Kameran är programmerad på ett sätt som gör att det är lättare för den att upptäcka cirkulära former utav en färg, på grund av masscentrumlagen. Detta ger högre precision och gör det enklare för kameran att upptäcka färger i högre hastigheter. Ett koordinatsystem är uppritat i Matlab av det som kameran ser av air-hockeybordet. Detta gör att positionerna för både klubba och puck blir enkel att följa med simpla beräkningar då kameran har 11 bildrutor per sekund.

De gamla projektgrupperna använde främst Arduinokoden till att göra uträkningar och att styra roboten. Detta på grund av att de försökte lösa grundproblemet att skapa ett självspelande air-hockeyspel med reglerteknik. Då maskininlärning är i centrum för detta projekt så används inte all skriven Arduinokod då den är redundant för projektet. Koden som används beskriver främst styrning av motorerna.

Kapitel 3

Maskininlärning

Idag är maskininlärning en populär teknik som används i många program och tjänster. Det handlar om att kunna få en algoritm att analysera en mängd indata för att sedan hitta mönster och strukturer som kan representeras i form av en modell. Detta ska leda till att algoritmen gör förutsägelser eller förslag baserat på indatan. Indatan varierar beroende på vilka områden som maskininlärning används inom. Indata som ligger i fokus i det här projektet är numerisk mätdata som är framtagna av mätningar och tester [2].

En algoritm är en sekvens av instruktioner som transformerar indata till utdata. För en uppgift kan det finnas olika typer av algoritmer som exekverar samma kod och får likadana svar. Det gäller att hitta den algoritmen som är mest effektiv, minst antal instruktioner eller använder minst minne. I maskininlärning är det viktigt att algoritmen approximerar utdatan, sådan att mönster kan upptäckas gentemot indatan. Detta är nischen inom maskininlärning [2].

3.1 Algoritmer

Det finns tre huvudinlärningsmodeller inom maskininlärning. Inlärningsmodell väljs baserat på vilket område eller problem som behandlas. De tre inlärningsmodeller som rapporten beskriver kortfattat är Unsupervised learning och Supervised learning samt Reinforcement learning [2].

En algoritm kan på olika sätt modellera ett problem utifrån miljö eller vilken önskad erfarenhet roboten ska få. Det är därför viktigt att överväga vilken inlärningsmodell algoritmen ska följa. Nedanför förklaras de tre olika inlärningsmodellerna som finns och vad som utmärker dem [2].

3.1.1 Unsupervised learning

Unsupervised learning studerar hur ett system kan lära sig att representera särskilda inmatningsmönster på ett sätt som reflekterar den statistiska strukturen för den totala samlingen av inmatningsmönster. Till skillnad från Supervised- och Reinforcement learning finns det ingen tydlig utdata eller miljömässiga utvärderingar i samband med varje indata. Det handlar om att hitta olika grupperingar eller kluster i indata enligt deras likhet. Kluster är en teknik inom Unsupervised learning som fungerar att gruppera liknande data tillsammans. Algoritmen behöver inte vara tränad för att de ska hamna i samma kluster, detta skiljer sig från Supervised learning där algoritmen måsta vara tränad för att hitta data på rätt

sätt [2].

3.1.2 Supervised learning

Supervised learning innebär att inlärningsalgoritmen använder ett känt dataset, som även kallas träningsdataset, för att göra förutsägelser. Träningsdatasetet innehåller indata och utdata, därefter bygger Supervised learning algoritmen en modell som kan göra förutsägelser för utdatan för ett dataset. Att använda ett större träningsdataset ger ofta modellen en högre prediktiv effekt som kan generaliseras bra för nya dataset.

En Supervised learning algoritm analyserar träningsdata och producerar en avledad funktion som kan användas för att kartlägga nya exempel. Ett optimalt scenario gör att algoritmen korrekt kan bestämma klassetiketterna för osynliga instanser. Detta kräver att inlärningsalgoritmen generaliseras från träningsdata till osynliga situationer på ett rimligt sätt.

Supervised learning innefattar två kategorier av algoritmer, klassificiering- och regression. Klassificiering sorterar utdata i specifika klasser och regression skapar kontinuerlig utdata [2]

Supervised learning används i finansiella applikationer för kreditvärdighet, algoritmisk handel och obligations klassificering. I biologiska tillämpningar används Supervised learning för tumördetektion, läkemedelsupptäckter och i mönsterigenkänningsprogram för tal och bilder [4].

3.1.3 Reinforcement learning

Reinforcement learning innebär att agenten, det vill säga beslutstagaren, genom interaktion med sin omgivning kan lära sig att göra vissa efterfrågade handlingar med hjälp av belöningar eller bestraffningar. På det här sättet kan man träna ett program att prestera bättre och hitta den bästa lösningar för att nå sitt mål.

Skillnaden mellan Reinforcement learning och andra inlärningsmetoder är att agenten kan lära sig de önskade handlingarna genom att minnas vad som ger en belöning eller vad som leder till en bestraffning, det betyder att själva agenten måste upptäcka vilka handlingar som genererar mest belöningar. Ett val behöver inte alltid väljas utefter belöning, ett val kan väljas som leder till en annan situation som därefter leder till belöning. Det här beteendet kallas trial- and-error sökning inom Reinforcement [5].

Enligt Sutton och Barto finns det fyra dolda element som kan ha en avgörande betydelse för inlärningen, utöver agenten och omgivningen [5]. Första är ”policy” som definierar agentens beteende vid en given tidpunkt. Andra är ”belöningsfunktion” som definierar målet i Reinforcement learning, samt kartlägger och ger ett värde till en handling, där agentens mål är att välja den handling som genererar största belöningen. Om belöningen av den valda handling är låg kan policyn ändras för att få en bättre belöning i framtidens handlingar. Tredje dolda elementet är ”värdefunktion” som definierar vad agenten kan förvänta sig för belöning i slutet av en lång handling, det betyder att agenten kommer att hamna i situationer som består av handlingar med omedelbar belöning med låg värde eller ett högt värde i efterföljande handlingar som genererar högre belöning. Det fjärde dolda elementen i Reinforcement learning är ”omgivningsmodell”. Detta är en modell som härmar ett beteende hos omgivningen. Modellen används för planering och kan förutsäga vad en handling kommer att leda till för tillstånd och belöning och överväga möjliga situationer innan de upplevs[2].

Kapitel 4

Utveckling av algoritm

Ambitionen med projektet är att lära en robot spela så bra air-hockey som möjligt och målet är att roboten ska behärska spelet till en sådan nivå att den kan fullt ut ersätta en mänsklig spelare. Robotens rörelser, eller snarare robotens hastighet och positionen roboten ska färdas till, vid olika spelsituationer väljs av en algoritm som är baserad på maskininlärning. Idén är att algoritmen ska identiera ett mönster mellan kända indata och kända utdata, så kallad träningsdata, och därefter approximerar en modell som kan förutse utdata baserad på nya indata [2]. Träningsdata fås genom att observera och dokumentera två människor som spelar air-hockey, vilket är logiskt då roboten strävar efter att efterlikna en människas spelstrategi. En av personernas klubbrörelser används som utdata och puckens rörelser används som indata.

För att bli bra på en sport krävs mycket träning, förmågan att ta lärdom från tidigare erfarenheter och en förståelse för spelet och sin motståndare är en allmän sanning som kan appliceras på de flesta sporter, som till exempel air-hockey. De här visdomsorden är främst riktade till mänskliga sportutövare men de bör rimligtvis också gälla artificiell intelligens. För att algoritmen ska kunna producera en bra approximation måste mycket träningsdata matas in i algoritmen, ju mer träningsdata desto bättre modell är en bra riktlinje [2]. Förmågan att ta lärdom från tidigare erfarenheter innebär i det här fallet att algoritmen bör kunna identifiera rörelser som ger ett positivt utfall och rörelser som ger ett negativt utfall, och därefter gynna de positiva utfallen och välja bort de negativa utfallen. Med positiva utfall menas de rörelser som resulterar i att roboten gör mål och med negativa rörelser menas de rörelser som resulterar i att motståndaren gör mål. Identifiering och sortering av positiva respektive negativa utfall görs genom en så kallad reinforcement-algoritm [5].

På grund av tidsbrist har några avgränsningar gjorts gällande algoritmen. Algoritmen begränsas till att endast uppfylla det mest nödvändiga kravet på en fungerande robot, det vill säga approximera klubbans rörelse så att den träffar pucken. Det här innebär att reinforcement-funktionen inte implementeras i algortimen och att motståndarens rörelse ej beaktas vid val av klubbans rörelse. Anledningen till att motståndarklubbans rörelse ej beaktas är att det krävs en betydligt större mängd träningsdata för att uppnå en tillräckligt bra approximation.

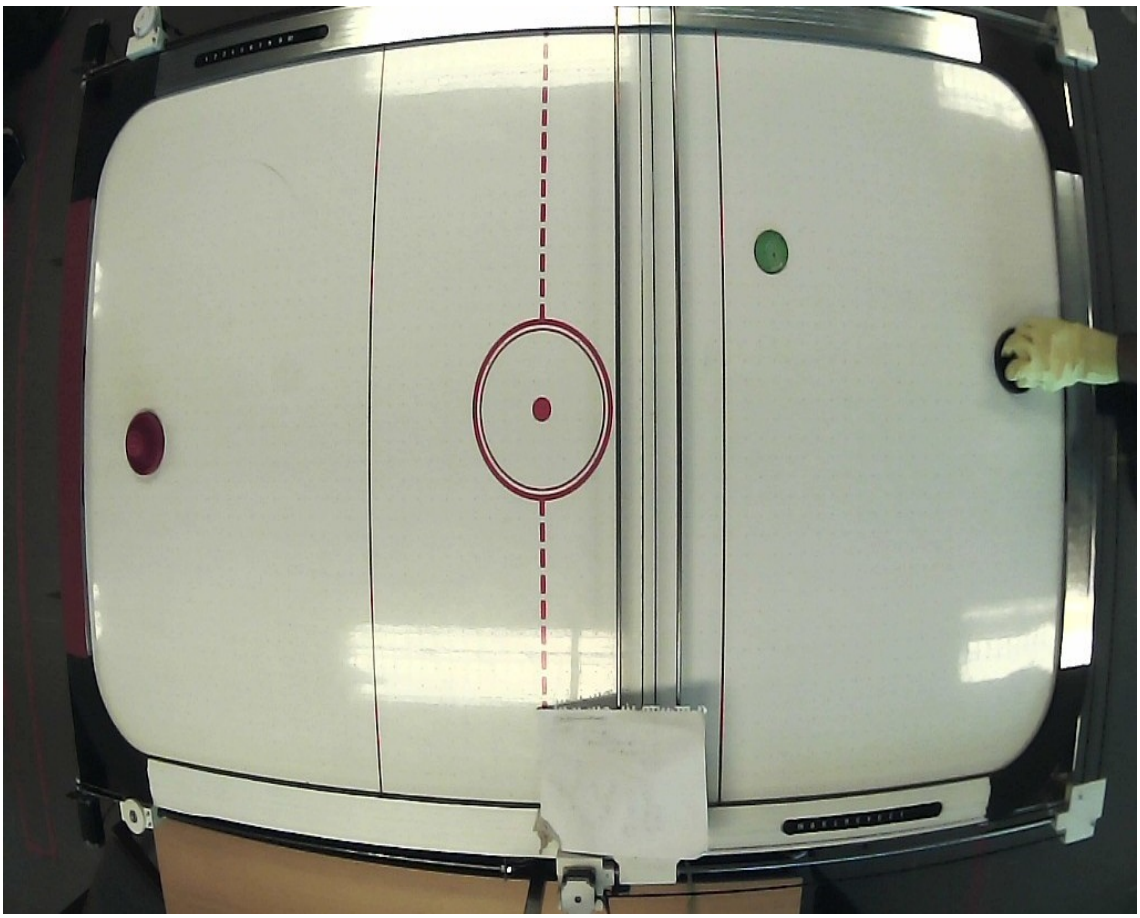
4.1 Insamling av data

Den data som algoritmen tränas med fås som nämnt ovan genom att observera och dokumentera när två människor spelar air-hockey mot varandra. Det här sker med hjälp av ett bildhanteringsprogram

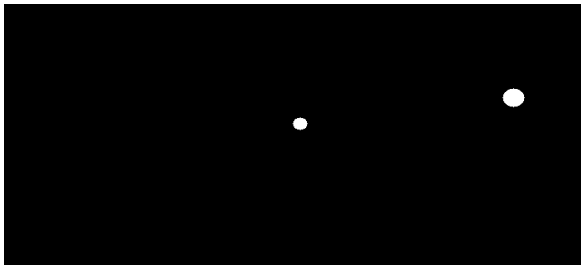
som kan filtrera ut en eller flera väldigt specifika färger ur en bild. I det här fallet är det en bild över air-hockeybordet där puckens och en av spelklubbornas position ska detekteras. Bilderna tas med en kamera som tar ungefär 11 bilder per sekund, vilket innebär att positionerna uppdateras ungefär 11 gånger i sekunden. Kamerasystemet och bildhanteringsprogrammet är skapade av en tidigare projektgrupp, för en mer ingående förklaring hänvisas läsaren till deras projektrapport [6].

Den träningsdata som är intressant i det här projektet är puckens position som används som indata och en av spelklubbornas position som används som utdata. De två objekten skiljs åt och detekteras i bildhanteringsprogrammet med hjälp av färgsystemet HSV, som står för hue, saturation, and lightness, vilket betyder kulörton, mättnad och ljusintensitet på svenska. Alla färger har en unik kombination av HSV-värden, och genom att definera ett objekts färg enligt färgsystemet HSV kan objektet filtreras ut i en bild [7].

Eftersom klubban kommer täckas över av spelarens hand så är det inte klubban som ska detekteras, utan snarare spelarens hand som ska detekteras. Det här löses genom att den ena spelaren bär en handske med en unik färg, i det här fallet gul. Pucken som används har en nyans av grön. Nedan visas en bild 4.1 på air-hockeybordet från kamerans perspektiv och en binär bild över air-hockey bordet efter filtrering 4.2 .



Figur 4.1: Bild från kameran som används i projektet.



Figur 4.2: Binär bild från kamerans perspektiv med filtrering.

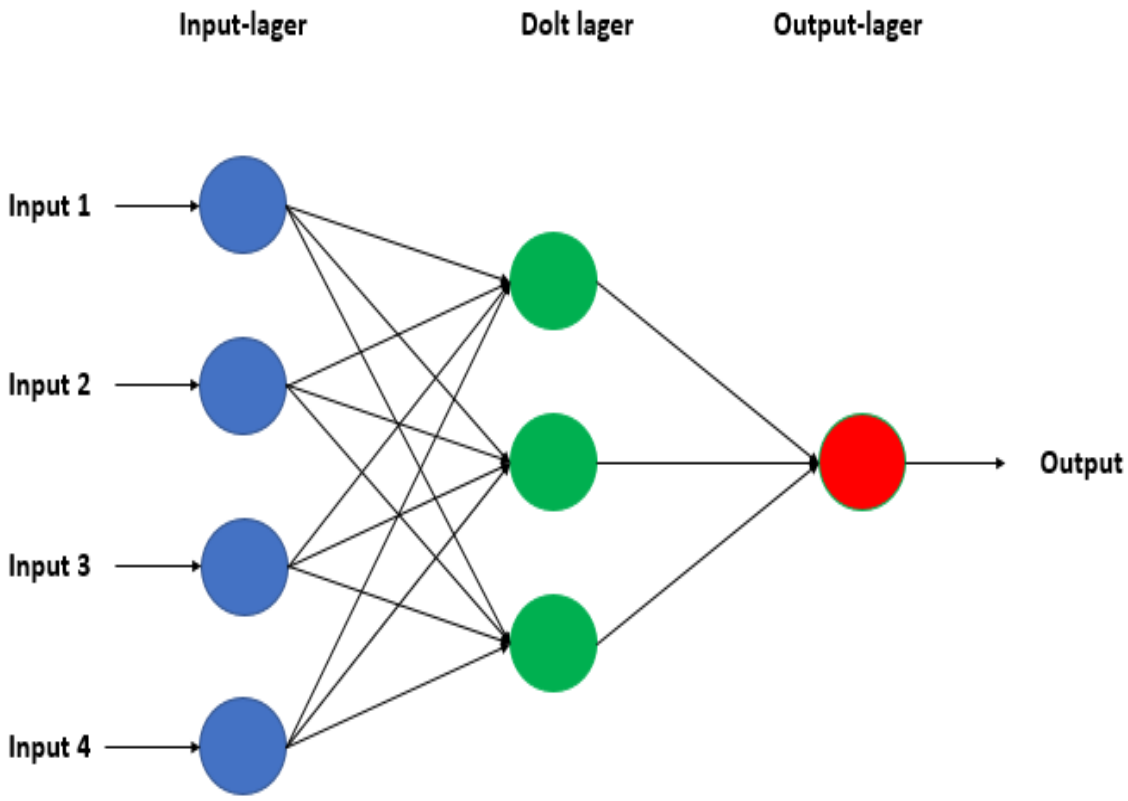
Objektens positioner i x-led och y-led sparas i en textfil för att sedan utnyttjas som träningsdata i algoritmen. Hur mycket träningsdata som krävs för att uppnå ett tillfredställande resultat beror på hur komplex problemet är och hur högt kravet på precision är. Från objektens positioner kan mycket information utvinnas, så som hastighet, färdriktning och avstånd mellan två objekt. Vilken typ av information som är bäst lämpad att använda som data finns ingen vetenskap om, istället testas olika kombinationer för att se vad som ger bäst resultat. Testning av olika typer av parametrar på träningsdata görs genom att implementera ett mindre representativt urval av träningsdata och därefter utvärdera vilka parametrar som ger bäst resultat. Anledningen till att ett mindre urval används är att det tar tid för algoritmen att bearbeta stora mängder data. Därefter sållas all träningsdata med uppenbarliga missvisande värden, som till exempel när klubban eller pucken står still en längre tid eller när kameran inte kan detektera klubban eller pucken. Till sist implementeras den kvarstående träningsdata som gav bäst resultat i algoritmen.

4.2 Konstruktion av algoritm

Det vanligaste sättet att konstruera algoritmer som approximerar en modell baserad på mönsterigenkänning av träningsdata är att bygga upp ett artificiellt neuronnät.Artificiella Neuronnät är ett delområde inom maskininlärning som tagit inspiration från hur den mänskliga hjärnan fungerar [8],[9].

Neuronnätet består oftast av tre lager, ett input-lager, ett output-lager samt ett eller flera dolda lager, som neuroner är uppdelade i. Antalet neuroner i input-lagret respektive output-lagret är lika många som antalet parametrar som används som indata respektive utdata. Antalet neuroner i det dolda lagret är inte fastställt men det är kutym att välja ett värde som ligger mellan antalet neuroner i input-lagret och output-lagret [10].

Bilden nedan 4.3 visar ett exempel på hur ett neuronnäts struktur kan se ut. Fyra stycken indata skickas in i nätet och med hjälp av den modell som skapas från träningsdatan kan ett utdata genereras. Modellen skapas genom att förbindelserna mellan neuronerna i input-lagret och neuronerna i det dolda lagret är viktade. Viktningarna anpassas kontinuerligt under träningsperioden så att kvadraten av differansen mellan ett verklig utdata från träningsdatan och ett utdata genererat av modellen blir så litet som möjligt,det vill säga E=(Känd utdata-approximerat utdata)^2 *skaminimeras*.



Figur 4.3: Struktur för ett neuronnät.

Viktningarna begynnelsevärden slumpas oftast till ett tal mellan 0 och 1. De nya viktningarna fås enligt NEW WEIGHT = WEIGHT + ERROR * INPUT [10]. Om tillräckligt med träningsdata implementeras i neuronnätet så kommer modellen kunna genera utdata med ett väldigt litet fel. Varje dolt lager kan producera sin egen modell, vilket kan vara nödvändigt för komplicerade problem.

Neuronnätet byggs upp i Matlab med hjälp av verktygslådan "neural network toolbox", som är en verktygslåda framtagen specikt för att bygga upp ett neuronnätverk [11]. Nätverkets uppbyggnad beror på vilket problem som ska lösas, och därför måste problemet analyseras. Det finns inget strikt linjärt samband mellan pucken och klubban, vilket innebär att problemet är icke-linjärt. Puckens rörelse kan liknas vid en kontinuerlig funktion i det avseendet att pucken aldrig hoppar från en position till en annan, detta innebär att puckens tidigare positioner är av intresse när klubbans rörelse approximeras. Med samma argument bör även klubbans tidigare positioner beaktas vid approximation av ny position, vilket innebär att neuronnätet bör ha en feedbackwards-struktur. Air-hockey är ett spel i väldigt högt tempo, vilket innebär att klubban kan få bekymmer med att hinna förflyttas till rätt position i tid. En lösning till det här problemet är att konstruera en algoritm med förmågan att förutse framtida utdata baserad på nuvarande indata, eller med andra ord förutse framtida klubbpositioner baserat på nuvarande puckposition. En avvägning som dock måste beaktas är att det försämrar approximationens precision. Puckens och klubbans positioner ändras med tiden, vilket innebär att det är ett tidsserieanalytiskt problem.

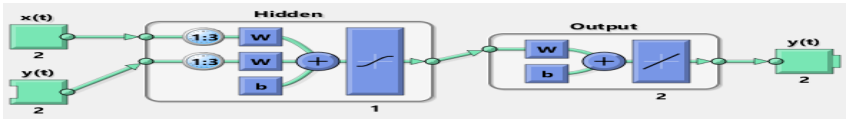
Med problemanalysen ovan beaktad byggs ett så kallat time series dynamic neural network upp som löser icke-linjära tidsberoende problem. Dynamiska neuronnät används när framtida värden ska förutses baserat på tidigare värden, vilket är tillämpbart i många olika branscher så som finansbranschen och som i det här fallet då ett system ska styras. De framtida värdena fås av att lösa ekvationen nedan.

y(t)=f(x(t-1)...,x(d),y(t-1)...,y(d))

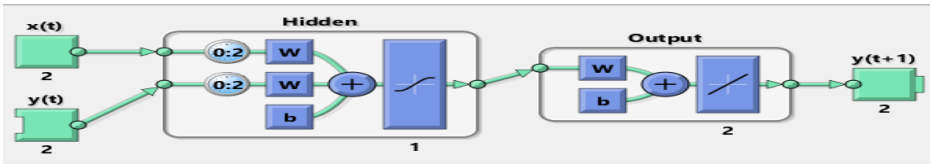
Där y(t) är den förutsedda utdatan, x(t) är indatan, d står för delay och är antalet tidigare värden som ska beaktas och t är den akutella bildrutan. Funktionen f fås genom att träna neuronnätet med den insamlade träningsdatan, vilket innebär att neuronnätet försöker identierar ett mönster mellan de insamlade indata och den tillhörande utdata. Antalet indata – och utdata-delays som används testas fram för att se vad som ger bäst resultat. Det samma gäller för antalet neuroner i det dolda lagret.

4.2.1 Val av neuronnät

För att skapa det bästa tänkbara neuronnätet för det aktuella problemet måste olika kombinationer på in-och utdata testas tillsammans med olika inställningar på neuronnätet. De typer av data som anses vara av intresse att testa är puckens och klubbans positioner, hastighet, färdriktning samt avståndet mellan pucken och klubban. De olika inställningarna som justeras på nätverket är antalet indata- och utdata-delays, antalet neuroner i det dolda nätverket och träningsmetod. Efter ett flertal tester kunde det bästa neuronnätet väljas. Det valda neuronnätets struktur visas i figur 4.4. Den andra strukturen som beaktades var ett så kallat step a head-neuronnät, se figur 4.5 Ett step a head-nät kan förutse framtida utdata baserad på nuvarande indata, vilket innebär att klubban får information var den ska färdas ett steg, eller i det här fallet en bildruta, i förväg. Det här är givetvis en väldigt bra fördel, men det visade sig att approximationen av utdata blev betydligt sämre och därför valdes den strukturen bort.



Figur 4.4: Det valda neuronnätet.

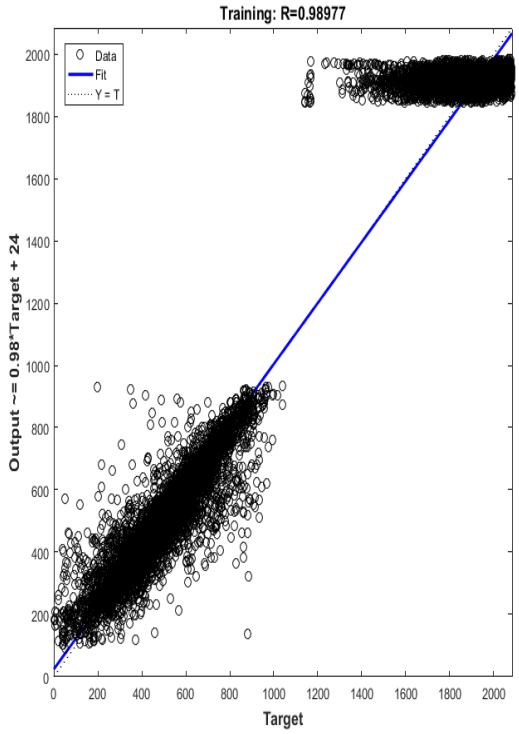


Figur 4.5: Step ahead neuronnätet.

Det valda neuronnätet använder puckens positioner som indata och klubbans positioner som utdata. Neuronätet använder sig av träningsmetoden Bayesian Regularization för att hitta en modell som approximerar utdata. Bayesian Regularization är en träningsmetod som lämpar sig bra för relativt

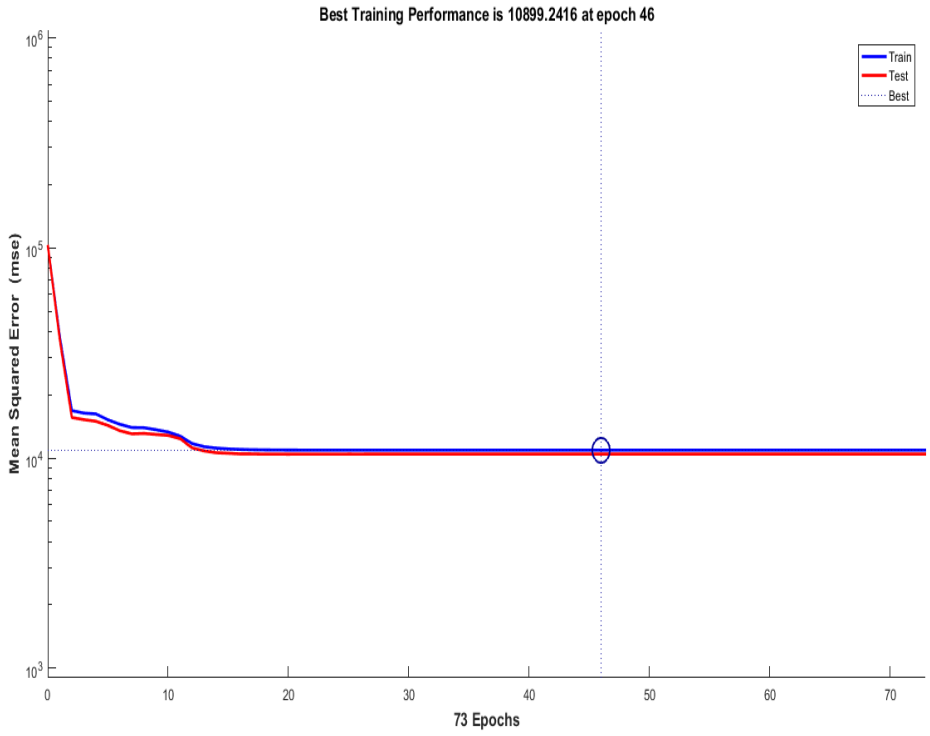
komplexa problem, dock så tar träningen lite längre tid men det är inget problem då den bara behöver tränas en gång. Antalet input-delays är tre stycken och antalet output-delays är också tre stycken, vilket innebär att nätet använder sig av tre tidigare in-och utdata när nätet skapar modellen.

I figur 4.6 visas hur den av neuronnätet approximerade utdatan korrelerar mot den verkliga utdatan som användes som träningsdata. Den approximerade utdatan stämmer överrens med den verkliga utdatan till över 98.8 procent, vilket innebär att approximationen är väldigt bra.



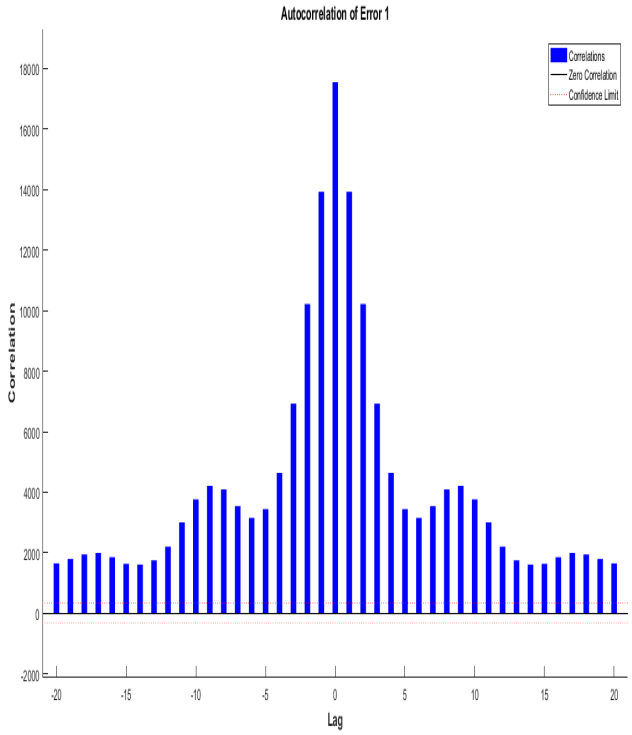
Figur 4.6: Grafen visar hur den approximerade utdatan korrelerar mot den verkliga utdatan.

I figur 4.7 visas hur kvadraten på differansen mellan den verkliga utdatan och den approximerade utdatan förändras med antalet gånger neuronnätet går igenom träningsdatasetet. Diagrammet visar att felet är som minst när neuronnätet går igenom tränningsdatasetet 41 gånger. Det är det bästa resultatet som neuronnätet använder.



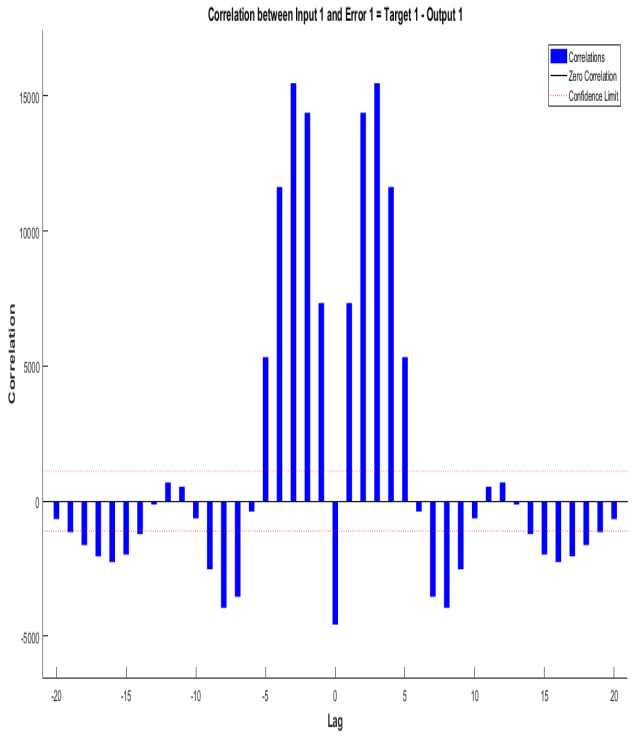
Figur 4.7: Root mean square error.

För att uppnå ett perfekt neuronnät så är det viktigt att utdatan endast har ett enda värde som autokorreleras och detta ska ske utan tidsfördröjning. Om det skulle visa värden som autokorreleras med tidsfördröjning så kan neuronnätet förbättras. Eftersom neuronnätet som används i projektet inte har perfekt korrelation utan tidsfördröjning kan det förbättras, se figur 4.6. Bilden 4.8 nedan visar ett neuronnät som kan förbättras eftersom flera värden autokorreleras med tidsfördröjning.



Figur 4.8: Autokorrelation av utdata i förhållande till tid.

Till skillnad från bilden ovan som visar hur utdatan korrelerar i förhllande till tid, så visar bilden 4.9 nedan hur indatan korrelerar med tiden. För ett perfekt neuronnät så ska alla värden vara noll.



Figur 4.9: Autokorrelation av indata i förhållande till tid.

Kapitel 5

Verifiering

Det här kapitlet kommer att ägna sig åt att redovisa resultatet av de krav och mål som är satta för projektet. Det har gjorts utförliga tester på varje krav och mål för att fastställa huruvida de uppfylls eller inte. Läsare hänvisas till figur A.1 för det uppställda krav och mål för projektet.

För att testa om (K1) och (K2) är uppfyllda så läggs pucken på olika koordinater nära mittlinjen och kanter. Koordinaterna är noga valda för det syftet att testa om roboten kan förflytta sig till olika koordinater runt sin planhalva. Testet undersöker även om roboten kan följa den angivna utdatan, eftersom den angivna koordinaten ska vara utdata från algoritmen. Roboten ska klara av två av tre tester för att (K1) ska anses godkänd, med en felmarginal mindre än 10% för att (K2) ska vara godkänd. Tre tester gjordes på varje koordinat för att med säkerhet fastställa att testet har lyckats. Klubban åker från sin initialposition med koordinaterna (1900,500), som befinner sig precis framför målet. Resultatet visar att klubban klarar testet med att träffa pucken även om det uppstår en liten felmarginal för det verkliga värdet klubban åker till. I tabellen 5.1 redovisas resultatet. Resultatet visar att kraven (K1) och (K2) är godkända.

Puckposition(X,Y)	Test1	Test2	Test3	IG/G
(1200,200)	(1240,218)	(1249,231)	(1260,228)	G
(1200,500)	(1255,512)	(1260,508)	(1266,514)	G
(1200,800)	(1266,823)	(1249,814)	(1249,816)	G
(1900.300)	(1931,315)	(1937,312)	(1927,308)	G
(1900,800)	(1888,868)	(1888,880)	(1892,948)	G

Tabell 5.1: Resultat för (K1) och (K2)

För att testa (K3) och (M2) så placeras roboten på en koordinat framför målet. Pucken skjuts sedan med en varierad hastighet mot målet för att undersöka om klubban kan blockera skottet. Detta resulterade i att klubban endast lyckades blockera pucken om dess hastighet var 0.5 m/s, då tidsskillnaden är för stor när klubban ska förflytta sig till pucken. Detta leder till att (K3) och (M2) inte är uppnådda. Resultaten redovisas i tabellen 5.2 nedan.

Puckens hastighet m/s	Test1	Test2	Test3
(0,5)	(G)	(G)	(G)
(1)	(IG)	(IG)	(IG)
(1,5)	(IG)	(IG)	(IG)
(2)	(IG)	(IG)	(IG)
(2,5)	(IG)	(IG)	(IG)

Tabell 5.2: Resultat för (K3) och (M2)

(K4) och (M1) testades genom att låta klubban försöka träffa inkommande puckar. Resultatet visar att klubban har svårt att träffa de inkommande puckarna på grund av tidskillnaden och när klubban väl träffar pucken är hastigheten på klubba och puck för liten för att överstiga kravet på 3 m/s. (K4) och (M1) är inte uppnått.

För (M3) antas att de föregående kraven är uppnådda för att skapa ett grundläggande spel. Detta leder till att klubban inte kan stå för 10% av målen gjorda i en match och (M3) är inte uppfyllt.

Ingen fokus har lagts på att försöka uppnå (M4), då fokus lagts på att försöka uppnå de andra mer kritiska kraven. (M4) är inte uppfyllt.

Kapitel 6

Diskussion

I det här kapitlet kommer robotens resultat och framtida förbättringar diskuteras. Alternativa lösningar till olika problem som uppkom kommer också diskuteras. Hänvisningar till kravspecifikationen A.1 kommer att göras för att bestämma och diskutera angående projektets slutresultat.

6.1 Utvärdering

Även om roboten inte fungerar som tänkt i praktiken, fungerar den i teorin. Syftet med det här projektet var att undersöka om maskininlärning var en bra lösning för att skapa en självspelande robot. Även om de önskade resultaten inte kunde uppnås i praktiken, så är algoritmen lyckad. Flera puck positioner testades för att kunna avgöra om (K1) och (K2) är uppfyllda. Testerna visade att klubban åker till positionerna där pucken befinner sig, dessutom träffade den pucken med en felmarginal mindre än 10% vilket tyder på att (K1) och (K2) är uppfyllda.

(K3) är inte uppnått. Kommunikationen mellan Matlab och Arduino är för långsam på grund av en implementerad delay i loopen som läser kommunikation från Matlab till Arduino. Detta gör det omöjligt för klubban att blockera en puck med en hastighet över 0.5m/s. Delayen som är implementerad är tyvärr nödvändig för att Arduino ska undvika att få in konstiga värden, vilket leder till felaktiga rörelser för roboten. Detta leder även till att (M2) blir ett orealistiskt mål.

Kravet (K4) och målet (M1) som beskriver hur klubban ska kunna skjuta tillbaka en puck som har en hastighet på 3 m/s respektive5 m/s, misslyckades. Eftersom Arduino får in värden för långsamt ifrån Matlab så är chansen väldigt liten att den skulle träffa pucken i dessa hastigheter. Klubban blir dessutom mindre träffsäker vid höga avstånd mellan klubba och puck. Detta beror på att vridmomentet minskar vid högre hastigheter och gör klubban mindre träffsäker.

(M3) förutsätter att kraven är uppnådda, vilket leder till att roboten omöjligt kan göra 10% av målen i en match. Då mycket tid har lagts på att försöka uppnå de satta kraven så har prioritet inte lagts på att försöka göra systemet användarvänligt (M4).

Projektet har varit utmanande på många sätt. Ett problem som uppstod tidigt var att det var betydligt svårare än förväntat att förstå de tidigare projektens kod. Alldeles för mycket tid lades på att försöka förstå kod än att utveckla algoritmen. På så vis hade möjligtvis algoritmen kunna bli ännu bättre och på så vis bättre resultat uppnås.

6.2 Framtida förbättringar

Något som framkom i slutet av projektets gång var att mer indata hade behövts för att ge roboten mer erfarenhet i vissa moment av spelet. Detta på grund av sena observationer som visar att den förflyttar sig bättre på ena sidan av sin planhalva.

Den största förbättringen som kan göras gällande algoritmen är att implementera Reinforcement learning. På så vis kan roboten själv lära sig vilka slag som är att föredra i vissa positioner av spelet. Att föredra så implementeras det i algoritmen att ge belöningar för slag som är målfarliga och bestraffningar om klubban skulle missa pucken eller skjuta den löst över till andra planhalvan.

Att lägga till så att roboten tar hänsyn till motståndarens rörelser är också en stor förbättring. Något som inte skulle ta allt för lång tid att göra, då det egentligen bara kräver att implementera en till färg som kameran kan upptäcka och att den nya klubbans positioner skickas in som indata till algoritmen.

Kameran som används i projektet kan ersättas med en kamera som har högre upplösning och fler bilder per sekund. Tyvärr är detta begränsat av budget och inte den viktigaste förbättringen.

Kapitel 7

Slutsats

Syftet med här projektet var att undersöka om maskininlärning kunde appliceras för att skapa ett självspelande air-hockeyspel.

Resultatet som uppnåtts är en robot som kan åka till hela sin spelplan och efterlikna algoritmens simulerade rörelse. Bristande kommunikation mellan Matlab och Arduino, samt brist på indata begränsar resultaten angående robotens syfte, vilket är att spela air-hockey. Troligtvis kommer roboten fungera avsevärt bättre om kommunikationen mellan Matlab och Arduino fungerade bättre.

När den simulerade utdatan jämförs med de faktiska värden roboten får utav algoritmen syns det tydligt att roboten förflyttar sig väldigt likt utdatan ifrån algoritmen. Vilket ger att det som återstår är att överföra det till praktiken. Detta ger att projektet är lyckat ur ett teoretiskt perspektiv

Då endast hälften av kraven och inga mål uppnåddes, så kan inte kraven för en godkänd självspelande robot mötas. Däremot kan den implementerade algoritmen anses vara lyckad, trots möjliga förbättringar.

Referenser

[1] Advania.se. (2015). Rådmark, H. Maskininlärning gör it smartare. [online] Available at: https://www.advania.se/Aktuellt/Nyheter/advania-news-nr-2-2015/maskininlarning-gor-it-smartare-an-du/. [Accessed 2017-02-02]

[2] Alpaydin, E. Introduction to machine learning,(2009) [book] Available at: http://cs.du.edu/ mitchell/mario_books/Introduction_to_Machine_Learning_-_2e_-_Ethem_Alpaydin.pdf.[Accessed 2017-04-02]

[3] Ansgar, M., Berggren, H., Borg, P., Damberg, R., Gudjonsson, M and Mared, L. Autonomous Air Hockey, kandidatarbete, Chalmers Tekniska Högskola. [online] Available at: http://publications.lib.chalmers.se/records/fulltext/240634/240634.pdf. [Accessed 2017-02-02]

[4] Brownlee, J. Master Machine Learning Algoritms. (2014) [Book] Available at: https://machinelearningmastery.com/master-machine-learning-algorithms/ [Accessed 2017-04-12]

[5] Sutton, R and Barto, A. Reinforcement learning,(2012) [book] Available at: http://people.inf.elte.hu/lorincz/Files/RL_2006/SuttonBook.pdf.[Accessed 2017-04-08]

[6] Berntsson, C., Brown, L., Fitz, Simon., Hallberg, A., Sondell, D and Svensson, K. Autonomous Air Hockey, kandidatarbete, Chalmers Tekniska Högskola. [online] Available at: http://publications.lib.chalmers.se/records/fulltext/242068/242068.pdf. [Accessed 2017-02-02]

[7] Shipman, J. Introduction to color theory, (2012) [online] Available at: http://infohost.nmt.edu/tcc/help/pubs/colortheory/web/hsv.html [Accessed 2017-04-30]

[8] Stergiou, C. and Siganos, D. Neural networks, [online] Available at: https://www.doc.ic.ac.uk/ nd/surprise_96/journal/vol4/cs11/report.html [Accessed 2017-04-30]

[9] Kriesel, D. A brief introduction to neural networks, (2005) [online] Available at:http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-1col-dkrieselcom.pdf [Accessed 2017-04-30]

[10] Hagsten, A. Maskininlärning med ett neuralt nätverk i C#-Del1-Teorin, [online] Available at: http://infozone.se/maskininlarning-med-ett-neuralt-natverk-i-c-del-1-teorin/ [Accessed 2017-04-25]

[11] Neural Network Time-Series Prediction and Modeling. The MathWorks, (2017) Inc. [Online] Available at: https://se.mathworks.com/help/nnet/gs/neural-network-time-series-prediction-and-modeling.html [Accessed 2017-03-20]

Bilaga A

Kravspecifikation

Kravspecifikation "Självspelande airhockey-spel"				
Huvudfunktioner	Krav och målnummer	Målvärde	Vikt	Verifering
Träffa pucken	K.1	Kunna träffa pucken på hela halvaplan	5	Fysisk provning
Klubbans rörelse	K.2	Skall kunna vara snarlik den simulerande utdata	5	Fysisk provning
Försvara målet	K.3	När puckens hastighet är 6 m/s	4	Fysisk provning
Slå tillbaka pucken	K.4	När pucken kommer i hastighet 3 m/s	5	Fysisk provning
Skjuta iväg pucken	M1	När puckens hastighet är 5 m/s	5	Fysisk provning
Försvara målet	M2	När pucken kommer i hastighet 8 m/s	4	Fysisk provning
Stå för 10% av målen	M3	Göra mål	3	Fysisk provning
Användarvänlighet	M4	Lätt att använda	1	Fysisk provning

Figur A.1: Kravspecifikation