Designspecifikation

Mats Tjäder

Version 1.0

Status

Granskad		
Godkänd		

PROJEKTIDENTITET

Linköpings tekniska högskola, Instutionen för systemteknik, ISY

Namn	Ansvar	Telefon	E-post
Björn Wedell	kundansvarig (KUN)	070-6655356	bjowe774@student.liu.se
Mats Tjäder	dokumentansvarig (DOK)	070-3545400	mattj452@student.liu.se
Henrik Einarsson	designansvarig bild (DESB)	070-3484676	henei960@student.liu.se
Marcus Olofsson	designansvarig robot (DESR)	070-6713303	marol791@student.liu.se
Johannes Eklånge	testansvarig (TST)	070-7711529	johek016@student.liu.se
Johan Nordin	presentationsansvarig (PRES)	073-1507959	johno660@student.liu.se
Alexander Konradsson	projektledare (PL)	070-2058260	aleko181@student.liu.se

Projekthemsida: http://www.cyd.liu.se/~mattj452/TSRT71
Kund: Henrik Tidefelt, ISY LiTH, 013-281311, tidefelt@isy.liu.se
Beställare: Martin Enqvist, 013-282306, maren@isy.liu.se
Kursansvarig: Anders Hansson, 013-281681, hansson@isy.liu.se
Handledare: Erik Wernholt, 013-281333, erikw@isy.liu.se

Innehåll

Dokumenthistorik

version	datum	utförda förändringar	utförda av	granskad
0.1	2005-02-17	Första versionen	Hela gruppen	Hela gruppen
0.2	2005-02-22	Andra versionen	KUN, DOK, DESB, PRES, PL	Hela gruppen
1.**0**	2005-02-23	Godkänd, inga förändringar från andra versionen.	PL	

1 Inledning

I denna projektkurs gjordes under år 2004 ett projekt där en industrirobot av modell ABB IRB1400 konfigurerades för att spela minigolf på en speciell bana. Den kunde dels styras manuellt genom att man angav utslagsvinkel och hastighet, dels genom att användaren endast angav vinkel och roboten automatiskt slog med rätt styrka för att bollen skulle gå i hål. I samma kurs 2005 kommer projektet att gå ut på att få roboten att prestera bättre bland annat med hjälp av en digitalkamera. Material från föregående år kommer att användas och troligtvis vara till stor nytta. Roboten kommer att kunna utföra samma saker som förra året, men förhoppningsvis bättre. Den kommer dessutom med hjälp av kameran att kunna bestämma mer exakt vad som är fel i ett slag och därmed förbättra sig till nästa slag. Roboten kommer också att kunna hitta en boll som stannat på banan och slå den i hålet därifrån eller alternativt hämta bollen. Banan har även utökats med en green.

1.1 Parter

Kund är Henrik Tidefelt, ISY, LiTH. Projektet utförs av en projektgrupp bestående av sju studenter i årskurs 4 på Y-programmet på kursen TSRT71 i Reglerteknisk projektkurs.

1.2 Mål

Syftet är att vidareutveckla det befintliga demonstrations-styrprogramet till industriroboten ABB IRB1400 vilket framtogs våren 2004 i projektet ”Golfspelande Industrirobot”. Roboten skall programmeras att spela minigolf på en liten men svår minigolfbana. Detta sker bl.a. med en mer avancerad matematisk modell, en kamera som identifierar bollbanan samt förbättrad dynamik.

1.3 Användning

Den minigolfspelande roboten skall visas upp vid olika informations- och reklamevenemang vid Linköpings universitet. Roboten skall utgöra ett exempel på en tillämpning av modellering och reglerteknik.

2 Översikt av systemet

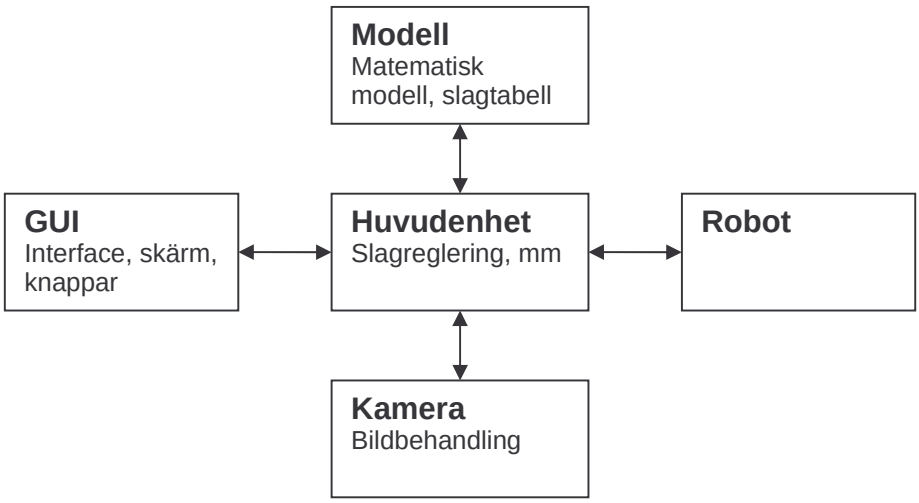
Systemet är uppbyggt av fem moduler: kamera, matematisk modell, grafiskt användargränssnitt, huvudenhet och robot. Det grafiska användargränssnittet och huvudenheten är ett program skrivet i Java där användaren med hjälp av knappar kan styra roboten att spela och ge olika order. Användaren ska kunna säga till roboten att slå bollen med angiven vinkel respektive hastighet, hitta bollen och välja mellan olika moder. Programmet ska också visa en bild av banan där beräknad bana utifrån angiven vinkel och hastighet syns. Efter ett slag (av robot eller människa) ska bollens egentliga bana också visas. Det kan även visas vad som behöver ändras i slaget för att bollen ska gå i hålet.

Den matematiska modellen kommer att används i tävlings-, uppvisnings- och manuell mod och kommer att implementeras i Matlab. Den kommer att ta information från huvudenheten och efter

beräkningar skicka tillbaka information om hastighet och eventuellt utslagsvinkel för bollen. I detta projekt skall den matematiska modellen förbättras gentemot 2004 års projekt.

En digitalkamera skall användas för att identifiera bollens bana. Detta för att huvudheten via den matematiska modellen ska kunna räkna ut nya parametrar vid ett misslyckat slag för att förbättra detta. Dessutom skall kameran kunna identifiera var en boll som stannat på banan ligger.

Roboten programmeras i programspråket Rapid 3.0 och utför olika kommandon från huvudenheten.



Figur 1. Blockschema över delmodulerna

2.1 Grov beskrivning av produkten

Det finns fyra olika moder. Tre av moderna inkluderar att roboten ska styras till att slå bollen. I den ena av dessa moder anges både vinkel och hastighet och roboten slår bollen enligt detta utan att tänka själv. I de andra två moderna anges endast vinkel och roboten beräknar med hjälp av modellen en lämplig hastighet för att träffa hålet. I alla dessa moder visas den beräknade bollbanan på bilden. I den fjärde moden används inte roboten utan istället får en användare slå ett slag själv på banan. Bollbanan läses in med hjälp av kameran och visas på bilden. Användaren kan nu få tips på hur slaget ska förbättras för att bollen ska gå i hålet.

2.2 Produktkomponenter

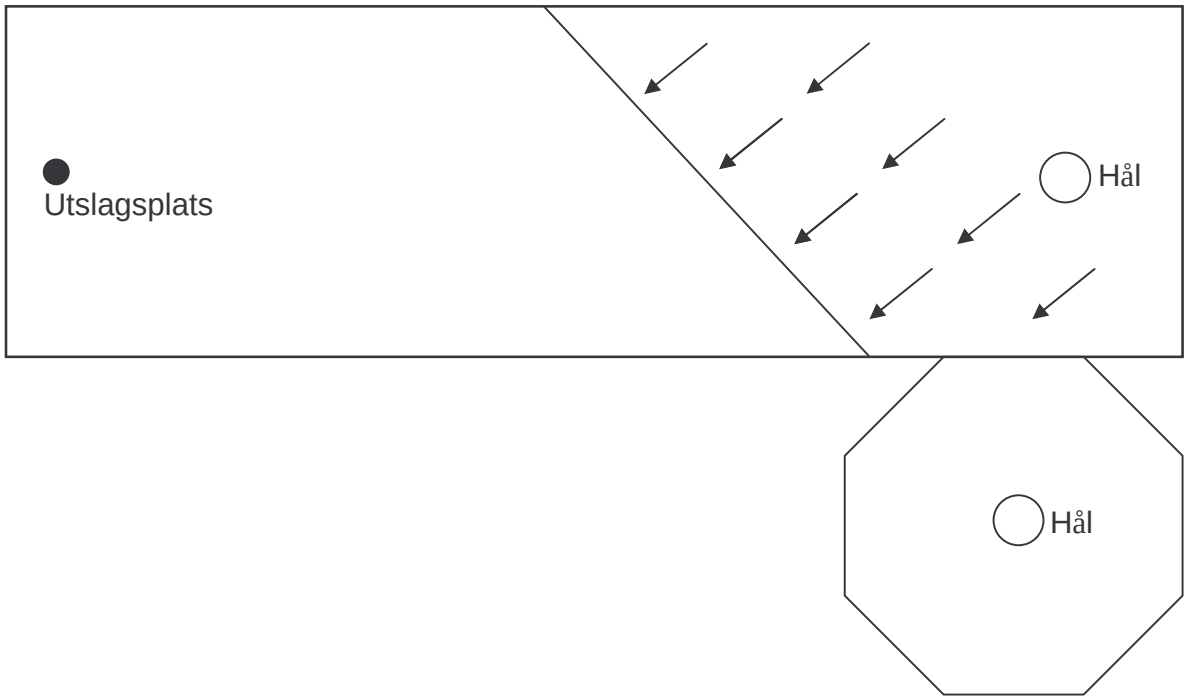
Användargränssnittet och huvudenheten är implementerade i Java och modellen i Matlab. Styrsystemet till roboten är S4C och programmeras i Rapid 3.0. Roboten är av modell ABB IRB1400.

2.3 Ingående delsystem

Systemet är uppbyggt av fem moduler: kamera, matematisk modell, grafiskt användargränssnitt, huvudenhet och robot.

2.4 Banspecifikation

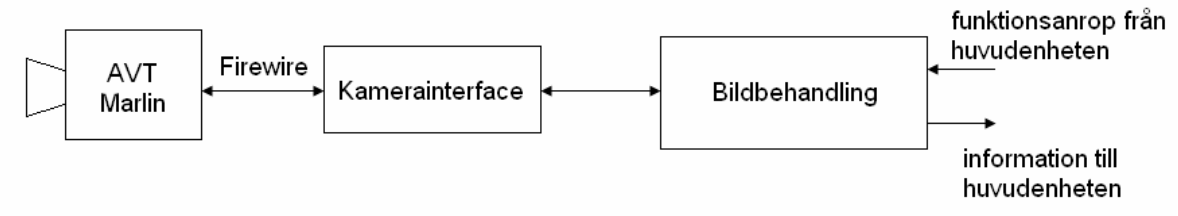
Banan är en minigolfbana som är byggd av ISY vid Linköpings universitet. Banan har en backe som sluttar åt ena kanten upp mot ett hål som leder ned till en green där hålet som bollen ska träffa finns.



Figur 2. Skiss över banan

3 Kamera

Systemet är utrustat med en digitalkamera av typen AVT Marlin F-145C2. Denna enhet ska ge information om slaget som huvudenheten ska kunna använda för att förbättra slag/modell. Den skall också identifiera var bollen är på banan så att roboten kan fortsätta att slå tills bollen gått i hålet och/eller flytta på bollen om det behövs. Kameran skall också i så stor utsträckning som möjligt användas för kalibrering av systemet.



Figur 3. Översikt, bildbehandling och kamera

3.1 Kamerainterface

Kameran är ansluten till den externa PC:n via Firewire. Ett kamerainterface implementeras i C++ med hjälp av AVT FirePackage. Kamerainterfacet skall kunna anropas via Matlab, vilket gör att det skrivs som en MEX-fil och returnerar bilder från kameran. Både enstaka stillbilder och bildsekvenser skall kunna returneras till bildbehandlingsalgoritmerna. Dessa bilder ska sedan behandlas i Matlab.

3.2 Bildbehandling

Bildbehandlingsenheten får förfrågningar från huvudenheten via funktionsanrop att returnera bollens position eller att leverera en uppsättning koordinater som beskriver bollens bana i x- och y-led i det senaste slaget. Förfrågningar kan också komma där bildbehandlingsenheten skall tala om i fall kamera och/eller banan har flyttats sedan den senaste kalibreringen.

3.3 Kalibrering av kamera

Kalibrering av kameran skall ske enligt metoden beskriven i [5]. Kameran skall också identifiera ett antal fixa referenspunkter i rummet och på banan för att kunna avgöra om avstånd och/eller vinklar mellan dessa har förändrats sedan senaste kalibreringen, vilket innebär att en ny kalibrering kan behövas.

3.4 Gränssnitt mellan huvudenhet och kamerainterface

Kameraintefacet kommer att anropas genom att följande matlabkommandon skrivs i Matlabs kommandofönster: *CameraInteface('kommando',argument)*.

CAM:startAcq
Förklaring: Startar inspelning av bilder.
Insignaler: -
Utsignaler: -

CAM:startBus
Förklaring: Startar Firewirebussen.
Insignaler: Bussnummer
Utsignaler: -

CAM:getNext
Förklaring: Slänger bort den nuvarande bilden och dess tidsstämpel och tar fram nästa ur kön.
Insignaler: -
Utsignaler: -

CAM:getImage
Förklaring: Hämtar in den nuvarande bilden till Matlab
Insignaler: -
Utsignaler: Bilden i YUV-422 - format.

CAM:stopAcq
Förklaring: Stoppar inspelning av bilder.
Insignaler: -
Utsignaler: -

CAM:getTimeStamp
Förklaring: Hämtar in den nuvarande bildens tidsstämpel till Matlab.
Insignaler: -
Utsignaler: Tidsstämpeln.

IM:processImage
Förklaring: Identifierar bollpositionen i bilden.
Insignaler: Bild
Utsignaler: Bollens position i bilden och en flagga att den syns.

IM:postProcess
Förklaring: Beräknar bollens absoluta position, hastighet och om studs har skett för alla bilder från slaget.
Insignaler: Sekvens av signalerna från processImage
Utsignaler: Bollposition, hastighet, studs, boll-i-hål för alla bilder tagna under slaget.

4 Huvudenhet och grafiskt användargränssnitt

I det grafiska användargränssnittet ger användaren kommandon som leder till modbyte eller till att roboten utför kommandon. Kommandon till roboten finns angivet under Robotavsnittet.

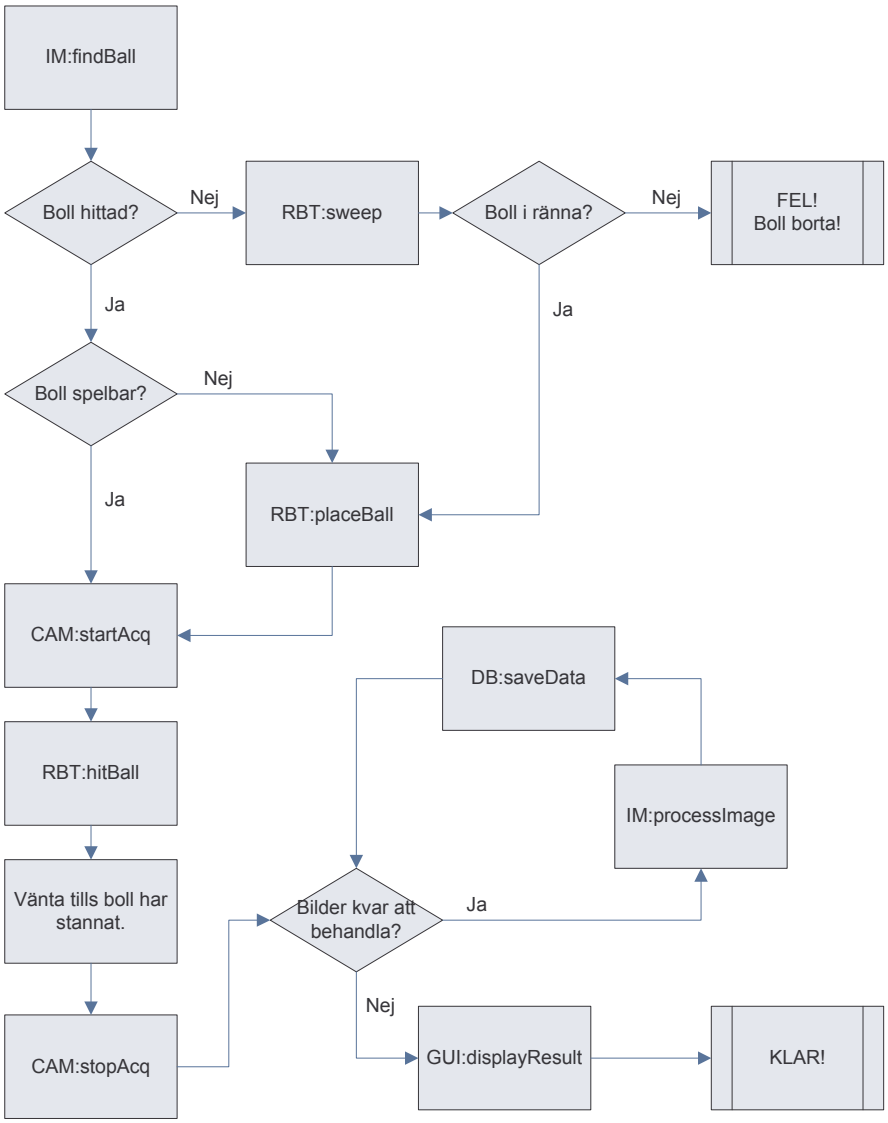
På skärmen som användaren ser kommer hela tiden en avbildning av minigolfbanan att synas. Till denna figur kommer ett koordinatsystem att knytas. Detta koordinatsystem är gemensamt för modellen och bildbehandlingsprogrammet. Med hjälp av detta ska man på ett relativt enkelt sätt kunna visualisera dels beräknade bollbanor från modell samt av kameran observerade bollbanor.

På skärmen kommer det även finnas knappar för att välja moder. Moderna som kan väljas är manuell mod (default), uppvisningsmod och träningsmod. Dessutom kommer det att finnas en knapp för att starta tävling som startar den fjärde moden, tävlingsmoden.

Användargränssnittet och huvudenheten uppträder lite olika beroende på vilken mod man befinner sig i.

4.1 Manuell mod

Om användaren väljer manuell mod kommer möjlighet ges att ange utslagsvinkel och hastighet. När dessa ändras finns en pil från utslagspunkten som ändrar riktning och skalas i förhållande till angiven vinkel och hastighet. På skärmen finns i denna mod även en knapp för att simulera slag. Denna knapp anropar kommandot MDL:simulate från modellen som ger bollens tänkta position i alla tidpunkter beräknat utifrån utslagsvinkel och hastighet. Detta ritas nu ut på bilden av minigolfbanan. I detta läge finns även en knapp för att få roboten att utföra slaget. Om bollen ligger i rännan utförs slaget enligt flödessekvensen i figur 4.



Figur 4. Flödessekvens vid slag

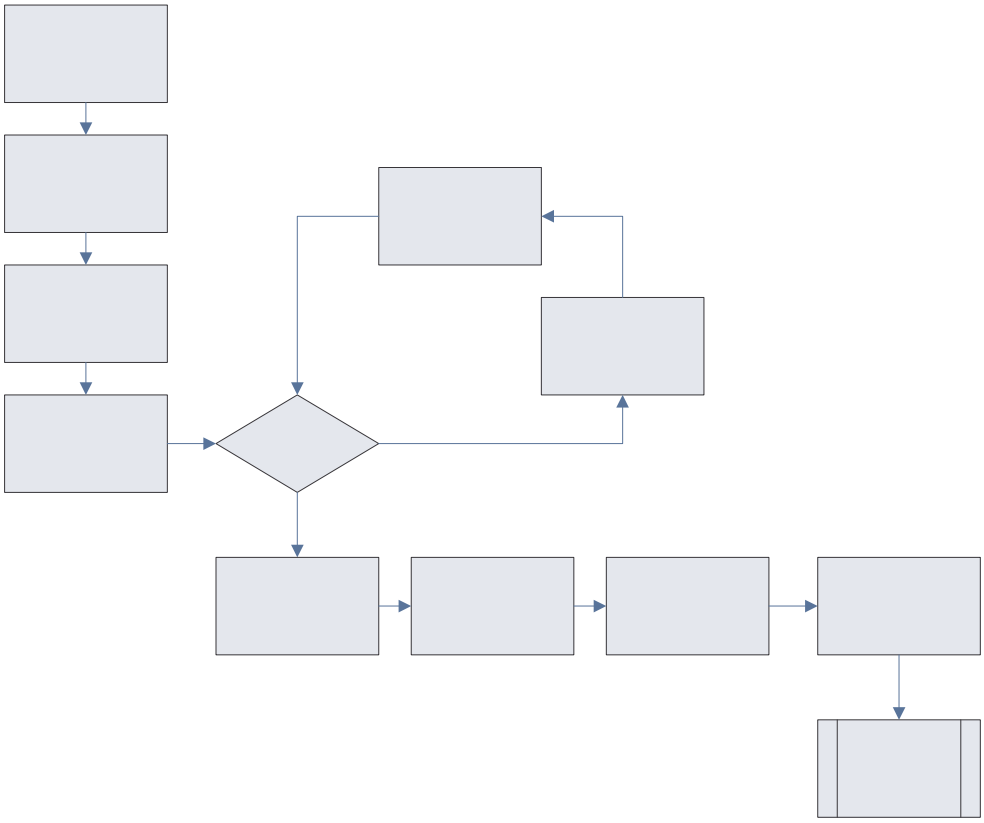
4.2 Uppvisningsmod

Tanken med uppvisningsmoden är att visa upp hur roboten kan lära sig slå bättre utifrån ett tidigare slag. För att detta ska ske måste modellen arbeta utifrån ganska grovt gissade parametrar och sedan förbättra sig med hjälp av justeringsparametrar. Modellens arbetssätt ändras med funktionen MDL:setMdlType i modellen. I uppvisningsmoden finns endast möjlighet att ange ett vinkelintervall i GUI:t. I jämförelse med den manuella moden finns nu en knapp för bestämning av lämpliga slagdata. Nu beräknas lämplig hastighet och exakt vinkel för att bollen ska gå så nära hålet som möjligt. Knappen för att beordra slag finns på exakt samma sätt som i den manuella

moden. Även här utgår vi från att bollen ligger i rännan (eller inte på green) och utför slaget enligt flödessekvensen i figur 4. När detta gjorts första gången träffar bollen antagligen inte hålet och funktionen MDL:update körs för att förbättra modellen utifrån det utförda slaget. Väljer man nu att beräkna slagdata på nytt borde det beräknade slaget stämma bättre överens med verkligheten och man borde se hur systemet utvecklas genom att observera sina egna handlingar.

4.3 Träningsmod

I träningsmoden medverkar inte roboten överhuvudtaget. Här får en användare slå ett slag och bollbanan analyseras med hjälp av kameran. Utifrån detta får användaren tips på hur slaget kan förbättras om bollen inte gick i hålet. I GUI:t finns nu endast en knapp där användaren kan ange att han eller hon är redo att göra ett slag. I och med detta får roboten order om att förflytta sig så att den inte står i vägen för användaren när denne ska slå ett slag. När roboten har förflyttat sig ska en instruktion om att trycka in nödstoppet visas i GUI:t. Efter detta startar sekvensen som visas i figur 5.



Figur 5. Flödessekvens vid träning

4.4 Tävlingsmod

Tävlingsmoden är till stor del en kombination av uppvisningsmoden och träningsmoden. När användaren väljer att starta en tävling så frågas det först efter namn på alla spelare och vilken typ av spelare det är. Med detta menas om det är en människa eller någon av de olika typerna av robotpersonligheterna. När detta är bestämt och användaren anger att den är klar sparas detta i någon temporär tabell och en poängtavla dyker upp. Förövrigt ser GUI:t ut ungefär som i träningsmoden om det är en människa som ska slå. Innan den mänskliga spelaren ska slå trycker man på knappen för att ange att spelaren är klar. I och med detta får roboten order om att förflytta sig så att den inte står i vägen för användaren när denne ska slå ett slag. När roboten har förflyttat sig ska en instruktion om att trycka in nödstoppet visas i GUI:t. Efter detta utförs sekvensen i figur 5. I den temporära tabellen ökas nu fältet för antal slag för denna spelare med ett. Detta repeteras tills bollen går i hålet och rullar ned till green. När bollen ligger på green utförs inte hela sekvensen utan antalet slag räknas endast tills bollen hamnar i hålet eller antalet slag överstiger 6. Om spelaren är roboten ser GUI:t ut ungefär som i uppvisningsmoden. Här finns dock inte möjligheten att ange vinkel och hastighet. En knapp för att få bollen att slå finns och här utförs sekvensen i figur 4. Antalet slag i tabellen för denna spelare räknas upp och proceduren upprepas tills bollen ligger i hålet eller antalet slag överstiger 6. När hela tabellen av spelare behandlats utses en eller flera vinnare.

4.5 Funktioner hörande till GUI:t eller huvudenheten

GUI:displayResult

Förklaring:	Funktionen plottar en bollbana på bilden av minigolfbanan.
Insignaler:	Matris av position vid olika tidpunkter. Denna kan komma från både IM (verklig) och MDL (beräknad).
Utsignaler:	-

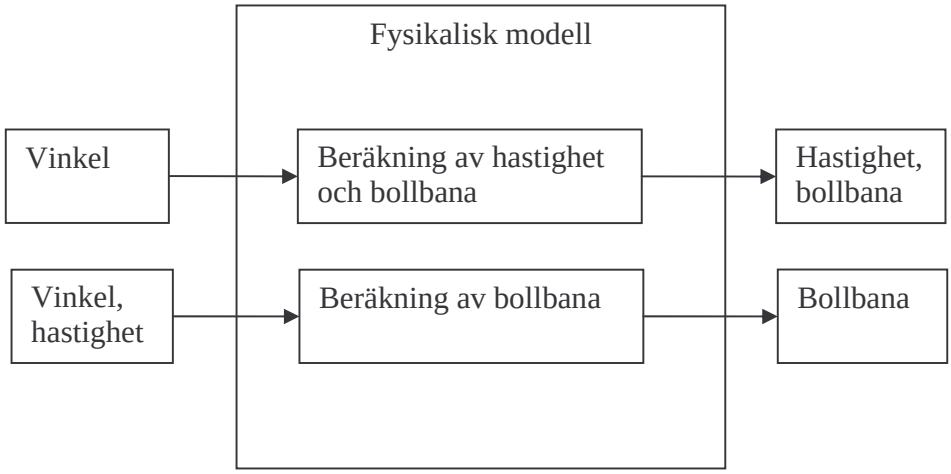
DB:saveData

Förklaring:	Sparar data om bollens uppförande i databas. Eventuellt beräknas de svårare variablerna (hastighet, studs?, energi, ihål?) också och sparas.
Insignaler:	x-position, y-position och tid från IM.
Utsignaler:	-

DB:findClosest

Förklaring:	Jämför en bollbana med alla sparade bollbanor och hittar den närmaste.
Insignaler:	En matris av x-position, y-position och tid från t.ex MDL:simulate.
Utsignaler:	-

5 Matematisk modell



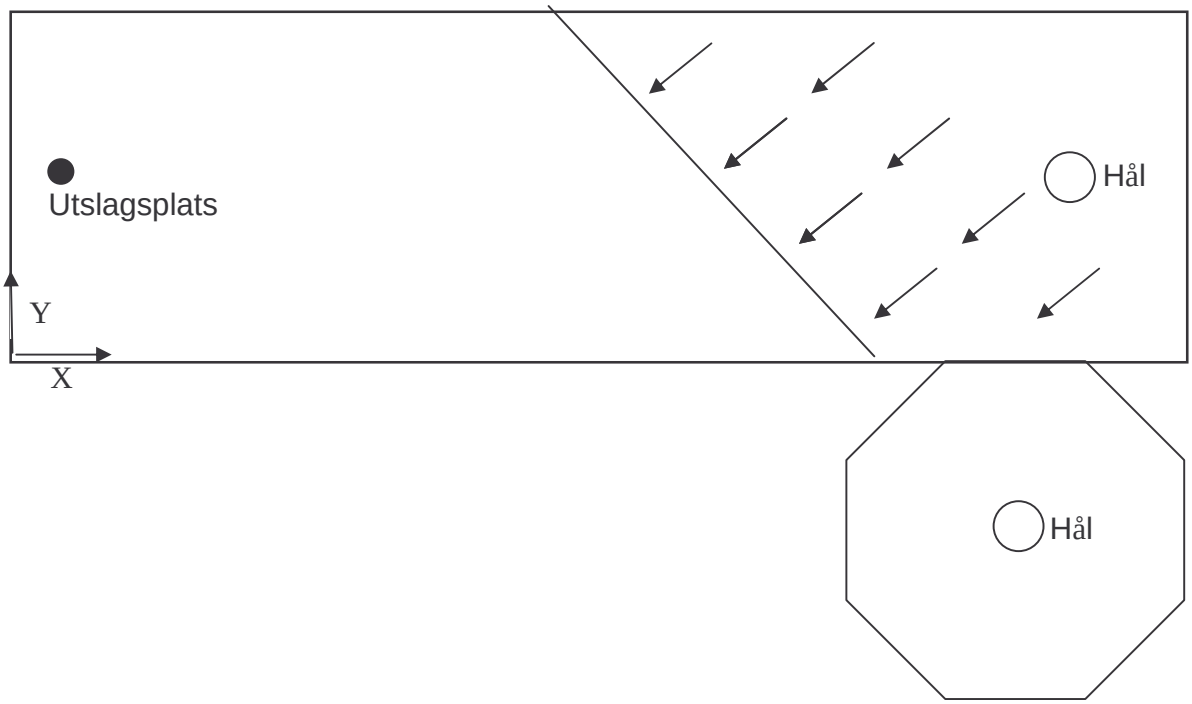
Figur 6. Matematisk modell

Den matematiska modellen ska utifrån en vinkel föreslå en hastighet som gör att bollen träffar eller kommer nära hålet. Dessutom ska simulerad bollbana returneras. Om både vinkel och hastighet anges ska bollbanan returneras. Kommunikationen mellan huvudenheten och modellen kommer att ske genom funktionsanrop i Matlab. Huvudenheten kommer att anropa modellen med dessa funktioner:

```
bollbana = simulate(vinkel, hastighet)

[hastighet, bollbana] = simulate(vinkel)
```

Hastigheten som avses är bollens utgångshastighet. Bollbanan kommer att anges i en matris med dess x- och y-koordinater. Koordinatsystemet är definierat enligt nedanstående bild.



Figur 7. Skiss över hur koordinatsystemet är definierat

Golfbanans utseende i rummet bestäms med mätningar från roboten. Banans normaler beräknas med hjälp av spline-toolboxen i Matlab. Bollens rörelseekvationer härleds ur 3-dimensionell kinematik och kinetik. Approximationsgraden i modellen bestäms efter hand, där approximationerna t.ex. kan vara av formen ”ingen glidning”.

Systemidentifieringen görs via optimering av en lämplig förlustfunktion, där simuleringsdata jämförs med data från slagdatabasen.

En statistisk modell över slag som går i hål byggs upp från slagdatabasen.

6 Robot

6.1 Inledande beskrivning av roboten

Roboten är en industrirobot av modell ABB IRB1400. Den har sex axlar och därmed sex frihetsgrader. Roboten styrs av styrsystemet S4C och programmeras med hjälp av ABB:s programspråk RAPID 3.0. Verktygen till roboten är utbytbara men kommer under detta projekt att bestå av ett verktyg bestyckat med minigolfklubba samt en sugkopp vända 180 grader mot varandra.

Roboten får parametrar från huvudenheten och använder dessa i olika subrutiner i robotens interna program. De subrutiner som finns är följande:

- ”hitOnTee”: utför utslag av boll från angiven startposition på tee med angiven vinkel och hastighet
- ”hitOnGreen”: utför putt av boll från angiven startposition på green
- ”placeBall”: utför hämtning av boll m.h.a. sugkoppen från angiven startposition och därefter placering av boll på angiven slutposition
- ”sweep”: utför fösning av boll på den del av banan där kameran inte kan ”se”
- ”moveAway”: ställer roboten i en position sådan att den inte skymmer kameran och inte står i vägen för en motspelare

Efter varje slutförd subrutin läser roboten av sensorn vid bollrännan som indikerar om en boll ligger där eller ej och skickar vidare denna information till huvudenheten.

6.2 Gränssnitt mellan robot och huvudenhet

Det finns två alternativ till gränssnitt mellan robot och huvudenhet. Båda dessa bygger på att kommunikationen mellan dessa enheter sker m.h.a. en textfil med namnet ”robot_huvudenhet”.

6.2.1 Alternativ 1

Alternativ 1 förutsätter att programspråket RAPID 3.0 klarar av att detektera om en textfil är låst eller ej. Om programspråket inte klarar av detta kommer alternativ 2 att användas. Vid alternativ 1 sköts all kommunikation mellan robot och huvudenhet via textfilen ”robot_huvudenhet”. Textfilen består av ett antal rader med en variabel på respektive rad. Dessa variabler kan läsas respektive skrivas över av robotens styrprogram och av huvudenheten. De ingående variablerna är följande:

which unit:	0 = huvudenheten får läsa från / skriva till filen 1 = robotens styrprogram får läsa från / skriva till filen
sensor:	0 = ingen boll i rännan 1 = boll i rännan
command:	någon av dessa textsträngar: ” hitOnTee”, ” hitOnGreen”, ”placeBall”, ”sweep”, ”moveAway”
angle:	[grader] (0 grader rakt fram)
speed:	[mm/s]
startposition x:	x-koordinaten för aktuellt kommandos startposition
startposition y:	y-koordinaten för aktuellt kommandos startposition

startposition z: z-koordinaten för aktuellt kommandos startposition

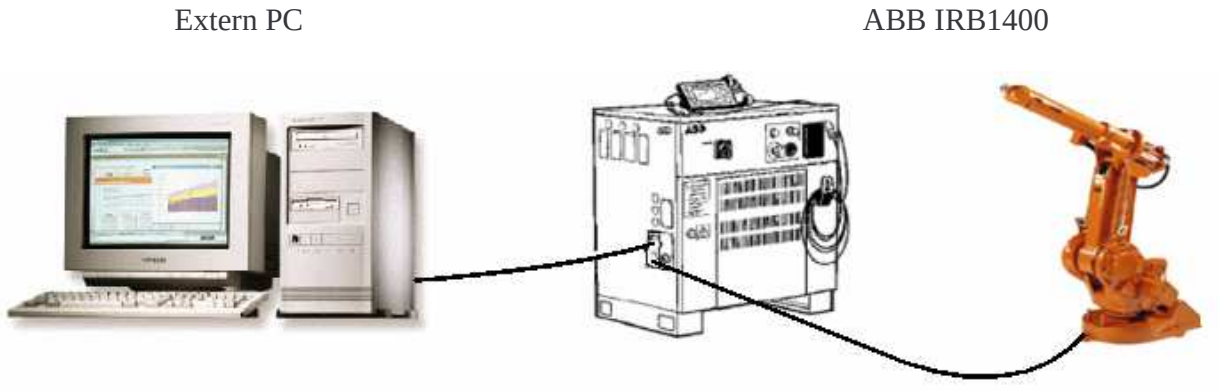
stopposition x: x-koordinaten för aktuellt kommandos slutposition

stopposition y: y-koordinaten för aktuellt kommandos slutposition

stopposition z: z-koordinaten för aktuellt kommandos slutposition

6.2.2 Alternativ 2

Alternativ 2 ser likadant ut som alternativ 1 med tillägget av de två tomma textfilerna ”robot” och ”huvudenhet”. Då roboten ska läsa från eller skriva till textfilen ”robot_huvudenhet” kollar den först om textfilen ”huvudenhet” existerar eller ej. Om den inte existerar skapar roboten textfilen ”robot” och sedan är den tillåten att använda textfilen ”robot_huvudenhet”. Om textfilen ”huvudenhet” existerar använder huvudenheten textfilen ”robot_huvudenhet” och roboten får då inte använda denna.



Figur 8. Robot med styrsystem

6.3 Funktioner hörande till robotens styrprogram

RBT:hitOnTee
Förklaring: Funktionen ger order om utslag från tee.
Insignaler: Position, vinkel och hastighet.
Utsignaler: -

RBT:hitOnGreen
Förklaring: Funktionen ger order om putt på green. Roboten räknar utifrån angiven position ut vilken vinkel roboten ska slå i.
Insignaler: Position, hastighet.
Utsignaler: -

RBT:placeBall
Förklaring: Funktionen ger order om hämtning av boll och sedan placering av boll.
Insignaler: Två positioner. Den första för vart bollen ska hämtas och den andra för vart den skall placeras.
Utsignaler: -

RBT:sweep
Förklaring: Funktionen ger order om fösning på den plats där kameran inte kan se.
Insignaler: -
Utsignaler: Flagga för om bollen hamnat i rännan eller ej.

RBT:moveAway
Förklaring: Funktionen ger roboten order om att flytta sig till angiven position.
Insignaler: Position.
Utsignaler: -

Referenser

[1] Svensson, Tomas & Krysander, Christian (2002), *LIPS – nivå 1*. Bokakademin, version 1.0.

[2] Enqvist, Martin (2005), *Projektdirektiv – Golfspelande industrirobot med kamera*.

[3] Tjäder, Mats (2005), *Kravspecifikation – Golfspelande industrirobot med kamera*.

[4] Tjäder, Mats (2005), *Systemskiss – Golfspelande industrirobot med kamera*.

[5] Zhengyou Zhang: *A Flexible New Technique for Camera Calibration*, Technical Report, MSR-TR-98-71, 1998, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, http://research.microsoft.com/~zhang