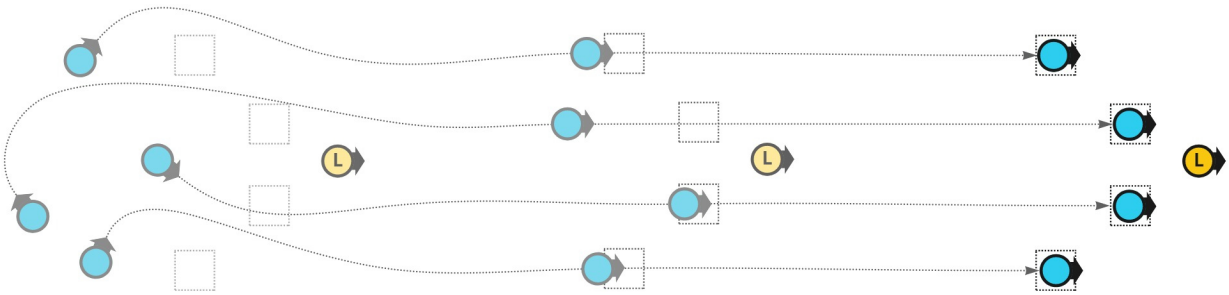
CHALMERS



Formationsbildning med kollisionsundvikning för sfäriska robotar

Kandidatarbete vid avdelningen för System- och reglerteknik på institutionen för Elektroteknik

ALVIN COMBRINK
DANIEL KARLSSON
DANIEL PETTERSSON
CHRISTOFFER SVERNLÖV
SARAH TORSTENSSON
JOHANNA WARNQVIST

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2019

Kandidatarbete 2019: EENX15-19-16

Formationsbildning med kollisionsundvikning för sfäriska robotar

ALVIN COMBRINK
DANIEL KARLSSON
DANIEL PETTERSSON
CHRISTOFFER SVERNLÖV
SARAH TORSTENSSON
JOHANNA WARNQVIST

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Elektroteknik
Avdelningen för System- och reglerteknik
Chalmers Tekniska Högskola
Göteborg, Sverige 2019

Formationsbildning med kollisionsundvikning för sfäriska robotar
ALVIN COMBRINK
DANIEL KARLSSON
DANIEL PETTERSSON
CHRISTOFFER SVERNLÖV
SARAH TORSTENSSON
JOHANNA WARNQVIST

Handledare: Nalin Kumar Sharma, Doktor, Elektroteknik
Examinator: Nikolce Murgovski, Docent, Elektroteknik

Kandidatarbete 2019: EENX15-19-16
Institutionen för Elektroteknik
Avdelningen för System- och reglerteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefonnummer +46 31 772 1000

Framsida: Bild som visar hur en formation bildas kring en ledarrobot. Se figur 4 för vidare förklaring.

Skrivet i LaTeX
Göteborg, Sverige 2019

Förord

Vi vill ägna ett stort tack till vår handledare Dr. Nalin Kumar Sharma för att ha väglett oss genom projektets gång och bidragit med värdefulla råd och insikter.

Forsättningsvis vill vi tacka vår examinator Docent Nikolce Murgovski för snabb återkoppling på planeringsrapporten och uppmuntrande feedback vid våra presentationer.

Vi vill även ägna ett tack till kandidatprojektgruppen EENX15-19-20 för all hjälp med vår rapport i form av respons på struktur och innehåll.

En uppskattning riktas till Ignacio Bona Piedrabuena för hans hjälp i vårt försök att kunna implementera vårt projekt i Sphero BOLT trots brist på officiell JavaScript API.

Ett tack går också till fackspråk för all hjälp med rapporten.

Slutligen vill vi tacka alla våra nära och kära för att de ORCAt lyssna när vi berättat om vårt projekt och kommit med ovärderliga råd som hjälpt oss på vägen.

Abstract

The purpose of this project is to implement a control system for spherical robots to build formations while avoiding collisions. The system was implemented with the ability to build formations around a leader robot which can move in the global coordinate system. Controllers have been designed from a mathematical model of spherical robots and the risk of collision is minimized by the use of reciprocal collision avoidance.

The technology this project addresses could be used in many different areas, where today's machines have difficulty performing tasks in the optimal way or in environments harmful to humans. Examples of such areas are security, autonomous traffic, exploration of planets and agriculture. The spherical robot Sphero BOLT, which is a tool for teaching and playing for children, has been used as a reference object for the project. However, the technology and theory behind the project can also be implemented in other types of spherical robots and unicycle models. With the help of calculations and simulations in MATLAB, results could be presented in a virtual environment. The result can be seen on Youtube at `https://bit.ly/2Z9byEb` and the code for it can be found at `https://git.io/fjWNG`.

Keywords: spherical robot, formation control, collision avoidance

Sammanfattning

Syftet med detta projekt är att skapa en styrning för att få sfäriska robotar att bilda formationer och under bildandet av dessa undvika kollisioner. Detta utfördes även med möjligheten att ha en ledarrobot som formationen ska bildas kring och som kan röra sig i det globala koordinatsystemet. Med en matematisk modell för sfäriska robotar har regulatorer designats för att kunna styra robotarna fart samt riktning och med hjälp av ömsesidig kollisionsundvikning minimeras risken för kollision.

Den teknik som projektet behandlar skulle kunna användas inom många olika områden där det med dagens maskiner är svårt att genomföra uppgiften på optimalt sätt eller där det finns skaderisk för människor. Exempel på sådana områden är säkerhet, autonom trafik, kartläggning av andra planeter och jordbruk. Projektet har som referensobjekt använt den sfäriska roboten Sphero BOLT, vilket är ett redskap för undervisning och lek riktad åt barn. Tekniken och teorin bakom projektet kan även implementeras i andra typer av sfäriska robotar och *unicycle models*. Med hjälp av beräkningar och simuleringar i MATLAB kunde ett resultat visas i virtuell miljö. Resultatet kan ses på Youtube via `https://bit.ly/2Z9byEb` och tillhörande kod för detta på `https://git.io/fjWNG`.

Nyckelord: sfärisk robot, formationsbildning, kollisionsundvikning

Förkortningar

API	Application Programming Interface
BLE	Bluetooth Low Energy
ORCA	Optimal Reciprocal Collision Avoidance (fritt översatt: Optimal ömsesidig kollisionsundvikning)
VO	Velocity Obstacle (fritt översatt: Hastighetshinder)

Begrepp

Globalt koordinatsystem	I rapporten syftar detta på ett koordinatsystem som står stilla i rummet.
Lokalt koordinatsystem	I rapporten syftar detta på ett koordinatsystem med ledaren som origo och ledarens färdriktning som positiva x-axeln.
Ortsvektor	Vektor som utgår ifrån origo.
PID-regulator	Regulator med Proportionell, Integrerande och Deriverande del.
Sphero BOLT	Sfärisk robot från företaget Sphero.

Variabler

Variabler markerade med *, enligt principen x^* där x är en godtycklig variabel, syftar till att förmedla att värdet är *önskat* istället för *nuvarande*.

p	Nuvarande position
x	x-koordinat
y	y-koordinat
θ	Vinkel
ω	Vinkelhastighet
v	Fart
$\boldsymbol{v}$	Hastighet
$\hat{e}$	Enhetsvektor
a	Acceleration
Ω_{max}	Maximala fysikaliska vinkelhastigheten hos roboten
t	Tid
ξ	Maximalt översläng för reglersystemets fart
K_p	Regulatorparameter för den proportionella delen av regulatorn
α_{max}	Maximal vinkelacceleration
s	Sträcka mellan robot och önskad position
K_d	Regulatorparameter för den deriverande delen av regulatorn
τ	Maximala tiden i sekunder framåt som kollisionsundvikning hanteras
$VO^{\tau}_{A\|B}$	Mängd med hastigheter som skulle leda till kollision om $v_A - v_B$ är inom mängden
$\boldsymbol{M}$	Mittpunkt för kollisionsdisk i hastighetsplan
R	Radie för kollisionsdisk i hastighetsplan
r	Radie för robot
s_{max}	Längsta avstånd mellan två robotar som ingår i kollisionssystemet
Δt	Tidssteg mellan beräkning av ny hastighet
$\boldsymbol{u}$	Förändringsvektorn i hastighetsplanet
$\boldsymbol{Q}_{\tau}$	Punkt i hastighetsplanet mellan $\boldsymbol{Q}_{\tau,1}$ och $\boldsymbol{Q}_{\tau,2}$
$\boldsymbol{Q}_{\tau,1}$, $\boldsymbol{Q}_{\tau,2}$	Tangeringspunkter för lilla kollisionsdisken och dess tangentlinjer som går genom origo i hastighetsplanet
$\hat{\boldsymbol{n}}$	Normalvektor ut ur hastighetshindret
d	Minsta avstånd mellan en hastighet till en linje
$\boldsymbol{P}$	Den punkt på en tangentlinje som är närmast den relativa hastigheten
$ORCA^{\tau}_{A\|B}$	Halvplan av tillåtna hastigheter för robot A givet robot B som garanterar att ingen kollision sker inom tiden τ
$\hat{\eta}$	Riktningsvektorn för skärningslinje
f	Detekteringsfrekvens

Innehållsförteckning

1 Introduktion **1**
1.1 Bakgrund . . . 1
1.2 Syfte . . . 1
1.3 Problemformulering . . . 1
1.4 Avgränsningar . . . 2
1.5 Rapportens uppbyggnad . . . 2

2 Formationer **3**
2.1 Det globala och lokala koordinatsystemet . . . 3
2.2 Formationsbildning . . . 3

3 Styrning och dynamik för enskild robot **5**
3.1 Matematisk modell av en robot . . . 5
3.2 Reglering av robotens position . . . 6
3.2.1 Reglering av robotens riktning . . . 7
3.2.2 Reglering av robotens avstånd till en önskad position . . . 8

4 Kollisionshantering **10**
4.1 Bakgrund till kollisionshantering . . . 10
4.2 Val av metod . . . 10
4.3 Hastighetshinder . . . 10
4.4 Begränsning av kollisionssystemets räckvidd . . . 12
4.5 Beräkning av förändringsvektorn $\boldsymbol{u}$ . . . 12
4.6 Hitta bästa tillåtna hastigheten . . . 15
4.7 Kompakta situationer . . . 17
4.8 Val av tidsspann för kollisionsundvikning . . . 19
4.9 Implementering av kollisionsundvikning i styrsystemet . . . 21

5 Simulering **22**
5.1 Simuleringsmetodik . . . 22
5.2 Simuleringsresultat för styrning och dynamik . . . 22
5.3 Simuleringsresultat för kollisionssystem . . . 27
5.3.1 Införande av brus . . . 27
5.3.2 Påverkan av τ på robotens rörelse . . . 30
5.3.3 Kollisionsundvikning för följare och ledaren . . . 31
5.3.4 Flera robotar som undviker varandra . . . 33
5.4 Simuleringsresultat för formationsbildning . . . 34
5.4.1 Stillastående formationsbildning . . . 35
5.4.2 Formationsbildning under rörelse . . . 36
5.4.3 Formationsbyte under rörelse . . . 38
5.5 Kompakta situationer . . . 39
5.6 Robot i låst läge . . . 40

6 Framtida utvecklingsområden **42**
6.1 Fysisk implementation . . . 42

6.2 Styrning och robotens dynamik 42
6.3 Kommunikationssystem 43
6.4 Anpassning till andra förutsättningar 43

7 Slutsats **44**

Referenser **45**

1 Introduktion

Sfäriska robotar har under de senaste åren blivit allt mer uppmärksammade, både inom forskning och genom deras medverkan i olika typer av populärkultur. Ett exempel på detta är filmen Jurassic World [1], där sfäriska fordon används för att förflytta sig i formation i den varierande terrängen utan att kollidera. Utöver dinosaurieparker kan sfäriska robotar och framför allt formationbildning användas i mer verkliga situationer, exempelvis inom den autonoma trafiken där behovet av kollisionsundvikning blir mer väsentligt.

1.1 Bakgrund

Fördelaktiga egenskaper hos sfäriska robotar är att de kan rotera kring sin egen axel samt att de inte kan välta. Detta kan vara en fördel i extrema eller svåråtkomliga miljöer där traditionella fordon eller människor har svårigheter att färdas av olika skäl. Ett exempel på sådana förhållanden är utforskning av planeter och områden där landskapet skiljer sig mycket från traditionella vägar [2].

I trafiken blir det även allt viktigare att fordon samverkar. Ökat samarbete mellan autonoma fordon bidrar till nya lösningar inom många områden, exempelvis hur kollisioner undviks mellan autonoma fordon eller hur miljöeffekter kan minimeras genom mjukare och mer koordinerad körning [3].

Utöver användningsområden i trafiken kan en formation av små luftburna och markbelägna robotar vara intressanta och fördelaktiga för jordbruk. Där skulle detta kunna användas vid skördning eller plöjning, men även vid säker och effektiv skadedjursbekämpning. Noggrannheten i processen skulle sänka både kostnader och hälsorisker för bönder samt även minimera miljöbelastningen i de berörda områdena [4]. Formationen skulle inte nödvändigtvis behöva bestå av en homogen grupp robotar, utan kan vara en mångsidig sådan, som utnyttjar olika styrkor och egenskaper. Varje del av formationen har då en enkel roll men som samlad grupp bildar de tillsammans ett komplext system, med mycket större möjligheter än som enskilda delar [5].

1.2 Syfte

Syftet med detta projekt är att utveckla och skapa ett system för att få sfäriska robotar att följa en ledarrobot medan olika formationer utformas och bibehålls. Formationsbildningen ska även innehålla kollisionsundvikning.

1.3 Problemformulering

Projektet har delats upp i delproblem för att uppnå syftet:

- Modellering av en sfärisk robot
  - En matematisk modell för hur en sfärisk robot rör sig behövs för att kunna designa ett reglersystem som i sin tur ska kunna styra en robot.

- System för formationsbildning med sfäriska robotar
  - För att undvika kollision vid körning av flera robotar behövs ett system för kollisionsundvikning.
  - För att formationen ska kunna förflyttas i det globala koordinatsystemet krävs en ledarrobot som formationen bildas kring och som sedan styr formationen.
- Simulering
  - Genom att simulera resultatet kan slutsatser kring systemet dras. Dels för att säkerställa varje dels funktionalitet men även hur de samverkar för att uppnå syftet. Därför behövs både enskilda och gemensamma simuleringar av formationsbildning, styrning och kollisionsundvikning.

1.4 Avgränsningar

Resultatet kommer presenteras i form av simuleringar i MATLAB och ingen fysisk implementation kommer ske. Systemet kommer vara blint för sin omgivning utöver de andra robotarna och ingen hänsyn tas till andra okända objekt. Robotarna kommer simuleras på plan mark utan yttre krafter som exempelvis luftmotstånd. Ytterligare kommer slirning hos robotarna försummas då perfekt friktion antas. Robotarna antas även inte kunna backa. Ingen hänsyn till inre mekanismer kommer tas vid framställningen av den matematiska modellen för systemet. Genom att bortse från kinetik och endast förhålla sig till kinematiken kommer ett universalt system erhållas för liknande modeller, vilket gör resultatet mer generellt och därmed möjligt att implementera för liknande modeller med annan dynamik men samma kinematik.

1.5 Rapportens uppbyggnad

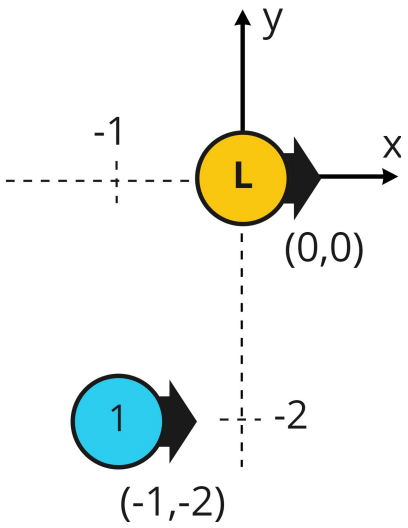
I kapitel 2 introduceras begreppet formationsbildning och hur ett effektivt system skulle kunna implementeras. Det tas upp hur robotarna förhåller sig till varandra samt vad ledaren har för roll i en formation. Sedan följer i kapitel 3 en framtagning av en enskild robots matematiska modell samt styrning för enskilda robotar. Detta för att ta roboten från en position till en önskad position och därmed möjliggöra bildandet av formationer. För att formationsbildningen ska ske så smidigt som möjligt krävs kollisionsundvikning mellan robotarna vilket följer i kapitel 4. Uttömmande och omfattande simuleringar samt diskussion kring resultatet behandlas i kapitel 5. I kapitel 6 diskuteras möjliga framtida utvecklingsområden för projektet och slutligen finns en slutsats i kapitel 7.

2 Formationer

Detta kapitel behandlar formationsbildning och hur formationer skapas. Förhållandet mellan ledarrobot och följarrobot utvecklas och centrala begrepp relaterade till styrning och positionering klarläggs.

2.1 Det globala och lokala koordinatsystemet

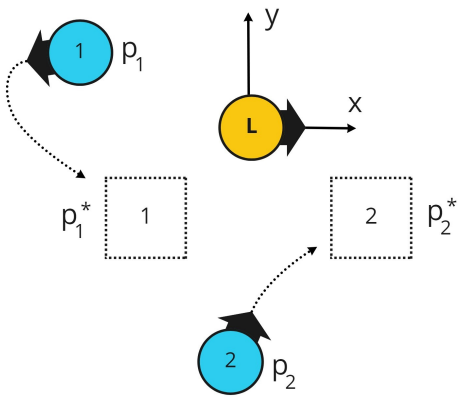
I rapporten syftar det globala koordinatsystemet på ett koordinatsystem som står stilla i rummet. En ledare för formationen implementeras för att ha en punkt att bilda formationen kring och även för att underlätta styrningen av formationen. Ett lokalt koordinatsystem, som utgår från ledarens position och färdriktning, används för att enklare kunna definiera positioner i formationen, se figur 1. Detta gör även att de önskade positionerna inte behöver uppdateras i det lokala koordinatsystemet när formationen rör på sig i det globala koordinatsystemet.



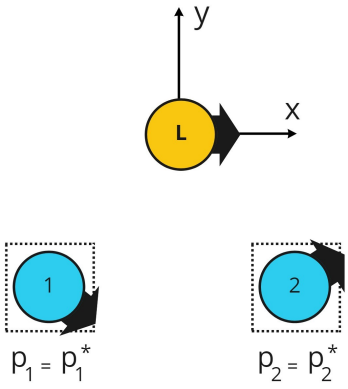
Figur 1: Det lokala koordinatsystemet utgår ifrån ett högerhandsorienterat koordinatsystem. Ledarroboten (markerad med ett L) har alltid koordinaterna $(0, 0)$ och den positiva x-axeln utgår ifrån ledarens färdriktning. Övriga robotar får sin position utifrån avståndet i meter till ledaren i x- och y-riktningen. I figuren har följarroboten (markerad med siffra) avståndet -1 m i x-riktningen och -2 m i y-riktningen, vilket ger den positionen (-1,-2) i det lokala koordinatsystemet.

2.2 Formationsbildning

Varje robot kommer att tilldelas en godtycklig önskad position, p^*, i formationen med villkoret att ingen av robotarnas önskade positioner överlappar. I figur 2 representeras en önskad position i formationen med en streckad ruta där siffrorna visar vilken robot som tilldelas vilken önskad position. Om följarroboten känner till positionen där den önskas vara samt sin riktning och sin nuvarande position p kan dess rörelse styras för att minska dess positionsfel. Figur 3 visar hur samma system ser ut efter att positionsfelet har reglerats till nära noll. Följarrobotarnas nuvarande positioner blir samma som de önskade positionerna och till följd bildas formationen.



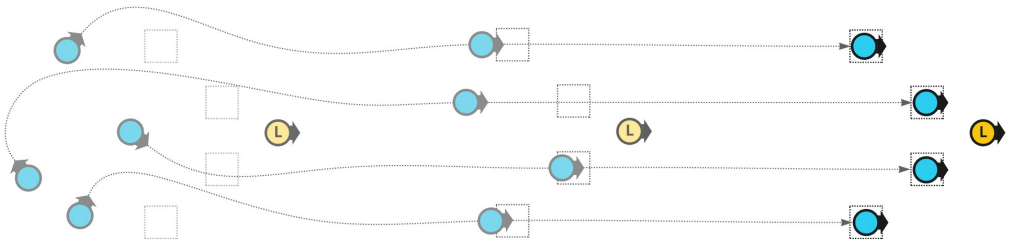
Figur 2: System av robotar vid byte eller uppstart av en formation. Robotarna får här ett fel mellan önskad och nuvarande position och rör sig för att minimera detta.



Figur 3: System av robotar efter att positionsfelet har reglerats. Här har en tid gått sedan figur 2 och robotarna har minimerat sitt positionsfel.

Robotens rörelse när felet minimeras blir likadan oavsett om felet minimeras via det globala eller det lokala koordinatsystemet, vilket gör att det system som passar implementeringen bäst kan användas. Görs till exempel en simulering där alla positioner i rummet är kända används till fördel det globala. Däremot om implementeringen sker i en fysisk robot där avståndet till ledaren är känt, är det lokala koordinatsystemet att föredra.

När ledaren förflyttar sig förflyttas de önskade positionerna med ledarroboten i det globala koordinatsystemet. I det lokala koordinatsystemet är det endast följarrobotarnas position relativt ledaren som förändras. När felet blir större kommer följarrobotarna försöka minska felet och därmed kommer följarrobotarna jaga efter och alltid sträva efter att skapa och därefter hålla formationen kring ledaren, oavsett om ledaren står still eller är i rörelse, se figur 4. Noterbart är att om ledarroboten rör sig nära begränsningen på robotarnas maxfart kommer följarrobotarna ha svårt att komma ikapp sina respektive önskade positioner. Maxfarten måste definieras utifrån underlaget, oavsett vilket koordinatsystem som väljs.



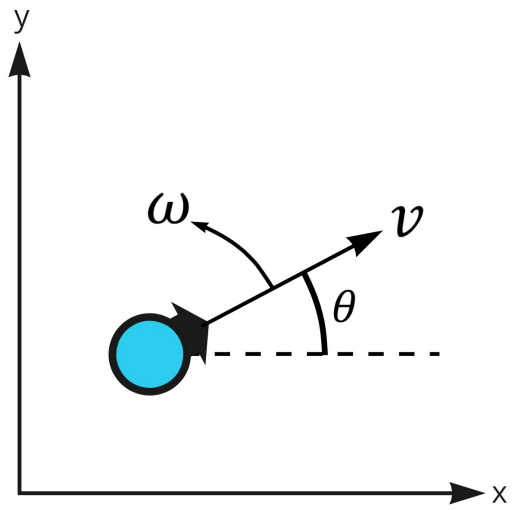
Figur 4: Figuren visar hur det lokala koordinatsystemet rör sig i det globala över tid. De önskade positionerna (fyrkanter) ligger precis på samma avstånd från ledaren vid alla tidpunkter, men flyttar sig i det globala koordinatsystemet. När följarrobotarna hittat sin position kommer dessa att ligga still i det lokala men har i själva verket en hastighet i det globala.

3 Styrning och dynamik för enskild robot

För att få flera robotar att åka i formation behövs först ett sätt att styra robotarna individuellt mot en önskad position. Detta kapitel berör en enskild sfärisk robot, som fritt kan röra sig i det horisontella planet. För denna robot är en förenklad matematisk modell av dess rörelse framtagen och styrning mot en given position implementerad.

3.1 Matematisk modell av en robot

Modelen skapades utifrån en sfärisk robot som fungerar som en *differential wheeled robot* (fritt översatt: differentialdriven robot), det vill säga innanför skalet styrs två hjul separat för att få roboten i rullning. Som förklaras i avgränsningar beaktas inte inre mekanismer som exempelvis masscentrum och motorer i framtagningen av den matematiska modellen. Modellen berör därför en sfärisk robot ur ett yttre perspektiv, där roboten fungerar som en stel kropp, enbart med hänsyn till systemets kinematik. Robotens kinematik kan beskrivas enligt en *unicycle model* (fritt översatt: enhjulingsmodell), det vill säga en modell för robotar utan svängradie, se figur 5.



Figur 5: *Unicycle model* för *differential wheeled robot*, där v är robotens fart, θ är robotens färdriktning relativt positiva x-axeln och ω är hastigheten med vilken θ förändras.

Robotens förflyttning i position samt riktning för ett sådant system beter sig enligt ekvation 1, där positionen i x- och y-led samt den vinkel θ roboten är riktad mot, relativt positiva x-axeln, påverkas av styrsignalerna farten v och vinkelhastigheten ω,

$$\begin{aligned} \dot{x} &= v\cos(\theta), \\ \dot{y} &= v\sin(\theta), \\ \dot{\theta} &= \omega. \end{aligned} \tag{1}$$

Ju högre fart v desto mindre kommer roboten klara av att svänga. Maximala vinkelhastigheten, ω_{max}, begränsas utifrån farten och accelerationen enligt

$$\boldsymbol{v} = v\,\hat{\boldsymbol{e}}_r, \tag{2}$$

$$\boldsymbol{a} = \dot{v}\,\hat{\boldsymbol{e}}_r + v\,\dot{\theta}\,\hat{\boldsymbol{e}}_r, \tag{3}$$

$$\|\boldsymbol{a}\| = \sqrt{\dot{v}^2 + (v\,\dot{\theta})^2} \leq a_{max}, \tag{4}$$

$$\dot{\theta} \leq \sqrt{\frac{a_{max}^2 - \dot{v}^2}{v^2}}. \tag{5}$$

I ekvationerna ovan står $\hat{\boldsymbol{e}}_r$ för enhetsvektorn i robotens färdriktning, $\boldsymbol{v}$ för robotens hastighet och $\boldsymbol{a}$ för robotens acceleration. Maximala teoretiska vinkelhastigheten utgör det maximala värde på $\dot{\theta}$ som uppfyller olikheten i ekvation (5). När farten går mot 0 går $\dot{\theta}$ mot oändligheten. För att ta hänsyn till en robots fysikaliska begränsningar sätts maximala vinkelhastigheten till

$$\omega_{max} = \min\left(\dot{\theta}, \Omega_{max}\right) \tag{6}$$

där Ω_{max} är maximala fysikaliska vinkelhastigheten för en robot. Utifrån ekvation (5) blir det tydligt att förmågan att svänga minskas då en hög acceleration till farten erhålls. För att försäkra att roboten alltid kan svänga begränsas $\dot{v}$ så att $\dot{v} \leq a_{begr} < a_{max}$, där a_{begr} är en vald acceleration beroende på krav för vinkelhastighet, ω. Ett stort a_{begr} innebär att en stor del av accelerationen finns tillgänglig till farten, medan ett litet a_{begr} begränsar accelerationen till farten så att förmågan att svänga bevaras. Risken med att välja en hög a_{begr}, och därmed prioritera robotens fart över förmågan att svänga, är att en situation kan uppstå där roboten passerar den önskade positionen innan den har hunnit rikta sig rätt.

3.2 Reglering av robotens position

Olika typer av lösningar för hur styrningen implementeras har undersökts och slutligen föll valet på ett reglersystem inspirerat av [6] där robotens fart och riktning regleras separat. En skillnad är att här regleras farten genom att reglera avståndet till den önskade positionen. Metoden för framtagningen baseras på kunskaper från [7].

För att en robot som befinner sig i positionen p ska ta sig till en önskad position, p^*, regleras dess riktning samt avstånd till p^*. Enligt ekvation 1 styrs roboten med styrsignaler för farten v och vinkelhastigheten ω, men v styrs genom en acceleration a, därav blir ω och a reglersystemets utsignaler. Robotens fart v bestäms vara större eller lika med 0 då robotarna i avsnitt 1.4 avgränsas till att inte kunna backa.

3.2.1 Reglering av robotens riktning

Låt $\hat{\boldsymbol{e}}_r$ vara robotens enhetsvektor i färdriktning och låt $\hat{\boldsymbol{e}}_g$ vara enhetsvektorn i riktningen från p till p_g, där $\boldsymbol{p}_g$ är ortsvektorn till en godtycklig punkt, p_g, vilket kan uttryckas som

$$\hat{\boldsymbol{e}}_g = \frac{\boldsymbol{p}_g - \boldsymbol{p}}{\|\boldsymbol{p}_g - \boldsymbol{p}\|}. \tag{7}$$

Låt θ_r och θ_g definieras som de positiva vinklarna mellan $\hat{\boldsymbol{e}}_r$ och x-axeln respektive $\hat{\boldsymbol{e}}_g$ och x-axeln. Minsta vinkelskillnaden $\theta_{r,g}$ mellan vektorerna $\hat{\boldsymbol{e}}_r$ och $\hat{\boldsymbol{e}}_g$ för någon position p_g blir således:

$$\theta_{r,g} = \begin{cases} \theta_g - \theta_r & \text{då } |\theta_g - \theta_r| \leq \pi \\ \theta_g - \theta_r - 2\pi \cdot \operatorname{sgn}(\theta_g - \theta_r) & \text{då } |\theta_g - \theta_r| > \pi. \end{cases} \tag{8}$$

Vid skapandet av ett reglersystem för robotens riktning finns det flera tillvägagångsätt. En metod är att direkt reglera robotens riktning mot den önskade positionen p^*. Då roboten är stilla fungerar detta väl, men när p^* förändras kan roboten hamna framför positionen på grund av exempelvis översläng i fartregleringen. Reglersystemet skulle då göra att roboten svänger av åt något håll där den sedan måste komma ikapp p^*, varpå situationen återupprepas. En metod som undviker detta problem är att reglera robotens riktning mot en position p_f^* som är framför p^* i dess hastighetsriktning. Låt t_f vara en tidskonstant och $\boldsymbol{v}_{p^*}$ vara önskade positionens hastighet. Ortsvektorn för positionen p_f^* beräknas då enligt

$$\boldsymbol{p}_f^* = \boldsymbol{p}^* + \boldsymbol{v}_{p^*} \cdot t_f. \tag{9}$$

Tidskonstanten t_f avgör hur långt framför p^* punkten p_f^* är och bestäms så att det maximala överslänget i fartregleringen är kortare än $\max\left(\|\boldsymbol{v}_{p^*}\|\right) \cdot t_f$. Låt maximala överslänget i reglersystemet för farten betecknas med ξ då kan t_f definieras som

$$t_f \geq \frac{\xi}{\max(\|\boldsymbol{v}_{p^*}\|)}. \tag{10}$$

Genom ekvation (7) och (8), med $\boldsymbol{p}_f^*$ istället för $\boldsymbol{p}_g$, fås vinkelskillnaden $\theta_{r,f}$ mellan robotens färdriktning och den önskade riktningen mot punkten p_f^*. Med $\theta_{r,f}$ kan en önskad vinkelhastighet ω^* bestämmas genom en P-regulator. Ett uttryck för ω^* fås enligt

$$\omega^* = \theta_{r,f} \cdot K_{p,\theta}, \tag{11}$$

där $K_{p,\theta}$ är regulatorns proportionella konstant. Som beskrivs i avsnitt 3.1 existerar en maximal vinkelhastighet ω_{max} som styrsignalen ω^* begränsas till. Förändringen av vinkelhastigheten över tid antas begränsas av systemet till en maximal vinkelacceleration α_{max} och därför behöver styrsignalen inte anpassas efter detta. Ekvation (11) kan skrivas om till

$$\omega^* = \min\left(\omega_{max}, \theta_{r,f} \cdot K_{p,\theta}\right). \tag{12}$$

3.2.2 Reglering av robotens avstånd till en önskad position

Då avståndet s från en robot till en önskad position p^* har reglerats till 0 så har roboten kommit fram till den. Avståndet s kan uttryckas som

$$s = \|\boldsymbol{p}^* - \boldsymbol{p}\|, \tag{13}$$

där $\boldsymbol{p}$ är ortsvektorn för nuvarande positionen och $\boldsymbol{p}^*$ är ortsvektorn för den önskade positionen.

Regleringen ska kunna hantera önskade positioner som förflyttar sig, vilket innebär att kvarstående fel i s kan förekomma och av den anledningen behöver reglersystemet en integrerande parameter. s är ett avstånd vilket innebär att den alltid är större eller lika med 0 och att dess integral endast kan växa med tid. Syftet med en integrerande parameter är att den ska kunna anta godtyckliga värden. För att detta ska vara möjligt måste den kunna minskas och därför kan integralen över s inte användas som I-regulator. Istället används en PD-regulator för att reglera avståndet till målposition, där insignalen är s och utsignalen behandlas som den relativa farten v^* i riktning mot den önskade positionen. Ju större avståndet s är desto högre blir v^*, vilket kan beskrivas som

$$v^* = K_{p,s}\, s + K_{d,s} \frac{d}{dt} s \tag{14}$$

där $K_{p,s}$ och $K_{d,s}$ är regulatorns konstanter.

En P-regulator reglerar sedan robotens nuvarande relativa fart $-\dot{s}$ till v^*, där båda är i riktning mot den önskade positionen. Robotens avståndsförändring $\dot{s}$ fås genom att derivera s med avseende på tid. Om s är en känd funktion kan $\dot{s}$ beräknas analytiskt, men i sammanhanget är det mer passande att beräkna den numeriskt. P-regulatorn för att reglera $-\dot{s}$ till v^* kan beskrivas som

$$a^* = (v^* + \dot{s})\, K_{p,v} \tag{15}$$

där $K_{p,v}$ är regulatorns proportionella konstant och styrsignalen kan behandlas som en önskad acceleration a^* av robotens fart. Den önskade accelerationen begränsas och skickas sedan till robotens styrsystem.

Som beskrivet behövs en integrering av felet s så att kvarstående fel motverkas. Då ekvation (15) inkluderar $\dot{s}$ samt att utsignalen a^* integreras två gånger för att påverka s så har termen $\dot{s}$ en integrerande effekt på s då

$$\int \left(\int \dot{s} dt \right) dt = \int s dt. \tag{16}$$

För att kunna hantera att roboten inte alltid är riktad mot den önskade positionen anpassas slutligen a^* för att ta hänsyn till detta. Robotens minsta vinkelskillnad θ_{r,p^*} mellan robotens färdriktning $\hat{\boldsymbol{e}}_r$ och riktningen till p^* fås ur ekvation (7) och (8) med $\boldsymbol{p}^*$ istället för $\boldsymbol{p}_g$. En ny ekvation för a^* kan skrivas som

$$a^* = (v^* + \dot{s})\, K_{p,v} \cos\theta_{r,p^*}, \tag{17}$$

där cosinus-faktorn illustrerar ett vinkelberoende för att sänka farten när vinkeln till den önskade positionen ökar. Som beskrivet i 3.1 begränsas a^* till den maximala accelerationen a_{begr} innan den skickas till roboten.

4 Kollisionshantering

För att robotarna ska kunna hitta till sina önskade positioner utan att krocka med andra robotar krävs en metod för att undvika kollisioner. I detta kapitel behandlas begreppet kollisionsundvikning, nödvändig bakgrund för ämnet samt vilken metod som implementerades i projektet.

4.1 Bakgrund till kollisionshantering

Kollisionsundvikning är ett typ av säkerhetssystem som minimerar risken för kollision. I många situationer där kollisionsrisk förekommer har alla robotar ett mål, men om alla robotar antar den hastighet som ger kortast eller snabbast väg till sina respektive mål kan kollision uppstå vid någon tidpunkt på vägen. Om kollisionssystemet inte tar hänsyn till målet utan endast undviker kollisionen kan det optimala valet av hastighet vara att låta robotarna åka i motsatt riktning från varandra. Detta leder till att robotarna kanske aldrig når fram till sina mål. Genom att ta hänsyn till målet och välja hastigheter som är så nära den önskade hastigheten som möjligt samtidigt som kollision undviks kan ett stabilt system bibehållas där slutmålet alltid kommer att nås såvida målen inte kräver att robotarna är inom varandras kollisionsradie.

4.2 Val av metod

Flera olika metoder och algoritmer för kollisionsundvikning har undersökts och slutligen föll valet på en vektorbaserad metod från [8] som är välbeprövad och välciterad [9], [10]. Metoden tar hänsyn till alla robotars positioner och hastigheter i realtid och säkerställer att kollision undviks τ tid framåt. Metoden är dessutom rättvis på så sätt att alla robotar bidrar lika mycket till kollisionsundvikningen. Ingen kommunikation behöver ske mellan robotarna utöver eventuell kommunikation för att veta nuvarande position och nuvarande hastighet för alla andra robotar. Varje robot behöver också känna till sin egen radie, maxhastighet och önskad hastighet. Om alla robotar har denna information kan de självständigt och parallellt räkna ut vilken hastighet de bör åka i för att undvika kollision och samtidigt sträva mot att närma sig sitt mål.

4.3 Hastighetshinder

Velocity Obstacle (fritt översatt: hastighetshinder) för robot A givet robot B och tiden τ, $VO^{\tau}_{A|B}$, är det område i hastighetsplanet där en kollision skulle ske mellan robot A och robot B inom tiden τ. Detta givet att robot B behåller sin nuvarande hastighet och robot A väljer en hastighet sådan att relativa hastigheten $\boldsymbol{v}_{rel} = \boldsymbol{v}_A - \boldsymbol{v}_B$ hamnar inom $VO^{\tau}_{A|B}$, där $\boldsymbol{v}_A$ och $\boldsymbol{v}_B$ är hastigheterna för robot A respektive B. För att säkerställa att kollision inte sker inom hela tidsintervallet 0 till τ definieras $VO^{\tau}_{A|B}$ matematiskt enligt [8] som

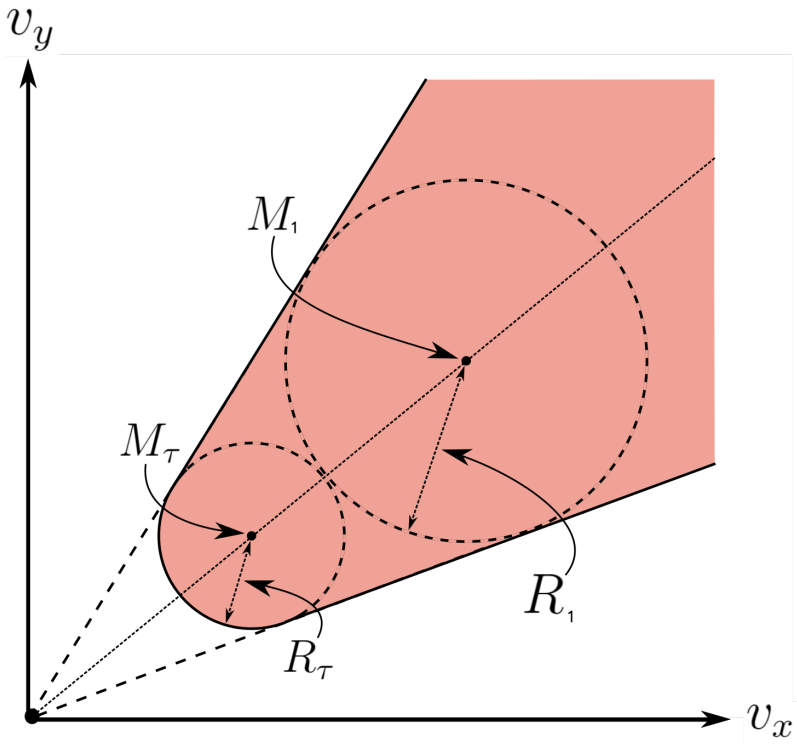
$$VO^{\tau}_{A|B} = \{\boldsymbol{v} \mid \exists t \in [0, \tau] : t\boldsymbol{v} \in D(\boldsymbol{p}_B - \boldsymbol{p}_A, r_A + r_B)\}\,, \tag{18}$$

där $\boldsymbol{p}_A$ och $\boldsymbol{p}_B$ är ortsvektorerna till positionen för robot A respektive B och r_A och r_B är radien för respektive robot.

Geometriskt bildar $VO^{\tau}_{A|B}$ ett område i hastighetsplanet i form av en kon som har sin spets i origo och sidor som tangerar kollisionsdisken betecknad $D(\boldsymbol{M}_1, R_1)$, se figur 6. Kollisionsdiskens mittpunkt $\boldsymbol{M}_1$ motsvarar den hastighet som leder till att robot A efter en sekund befinner sig på exakt samma position som robot B och fås av

$$\boldsymbol{M}_1 = \frac{\boldsymbol{p}_B - \boldsymbol{p}_A}{1}, \tag{19}$$

där 1 är en sekund. Kollisionsdisken har radie $R_1 = \frac{r_A + r_B}{1}$ m/s, där 1 är en sekund. Alla hastigheter inuti kollisionsdisken leder därmed till att robot A och robot B efter en sekund kommer ha ett avstånd till varandra som är mindre än deras gemensamma radie, och därmed har kollision uppstått.



Figur 6: Hastighetshindret VO (färgat område) som utgör de relativa hastigheter som skulle leda till kollision för två givna robotar inom tiden τ. I detta fall är $\tau = 2$ sekunder. $\boldsymbol{M}_1$ är den hastighet som leder till att robotarna befinner sig på exakt samma position efter en sekund. Kollisionsdisken i hastighetsplanet med mittpunkt i $\boldsymbol{M}_1$ och radien R_1 utgör alla hastigheter som orsakar kollision efter en sekund. Den nedre disken representerar de hastigheter som leder till att robotarna befinner sig inom kollisionsradie från varandra efter exakt τ sekunder.

Relativa hastigheter med samma riktning som någon hastighet inom $D(\boldsymbol{M}_1, R_1)$, men med högre fart, kommer alltid att resultera i kollision inom en sekund. Däremot kan en väldigt låg hastighet förhindra kollision inom tidsintervallet ifall farten är tillräckligt låg för att robotarna inte ska hinna fram till varandra. Geometriskt kan

därför konens spets klippas bort vid randen av disken $D(\boldsymbol{M}_\tau, R_\tau)$, där $\boldsymbol{M}_\tau$ och R_τ definieras som

$$\begin{aligned} \boldsymbol{M}_\tau &= \frac{\boldsymbol{p}_B - \boldsymbol{p}_A}{\tau}, \\ R_\tau &= \frac{r_A + r_B}{\tau}. \end{aligned} \tag{20}$$

Disken $D(\boldsymbol{M}_\tau, R_\tau)$ innehåller alla hastigheter som leder till kollision efter exakt τ sekunder. Vid lägre hastigheter rör sig robotarna mot att kollidera, men hinner inte fram till varandra innan τ sekunder och hastigheterna är därmed tillåtna.

4.4 Begränsning av kollisionssystemets räckvidd

För att minimera antalet beräkningar kan ett maximalt avstånd s_{max} väljas så att robotar längre bort än detta inte tas hänsyn till vid kollisionshanteringen [8]. Genom att försäkra att alla robotar med risk för kollision tar hänsyn till varandra kan detta maxavstånd sättas till

$$s_{max} = 2\tau v_{max} + r_A + r_B \tag{21}$$

där v_{max} är maximala farten. Vid beräkning av kollisionsundvikning antar varje robot att övriga robotar i systemet har samma maxfart som den själv, vilket gäller när alla robotar är av samma slag som i denna rapport. Avståndet s_{max} utgör två gånger det maximala avståndet som en robot kan röra sig inom tidsintervallet $(0, \tau)$ plus det avstånd som krävs mellan robotarna i slutändan för att kollision inte ska ske. Detta säkerställer kollisionsundvikning under intervallet, men kan leda till kollision i nästa tidpunkt om inte accelerationen är optimal så att robotarna kan ändra riktning och fart direkt. Detta leder dock inte till några problem eftersom kollisionsundvikning beräknas för varje tidssteg $\Delta t \ll \tau$ och därmed kommer robotarna justera sin riktning och fart i tid för att undvika kollisionen.

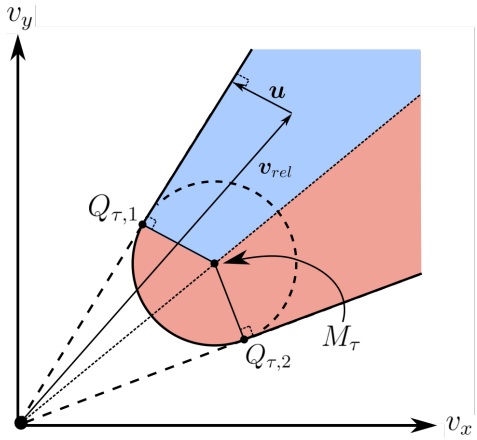
4.5 Beräkning av förändringsvektorn $\boldsymbol{u}$

För att beräkna tillåtna hastigheter för robot A med avseende på robot B beräknas först vektorn från relativa hastigheten $\boldsymbol{v}_{rel}$ till närmaste punkt på randen av $VO^\tau_{A|B}$. Denna förändringsvektor betecknas med $\boldsymbol{u}$,

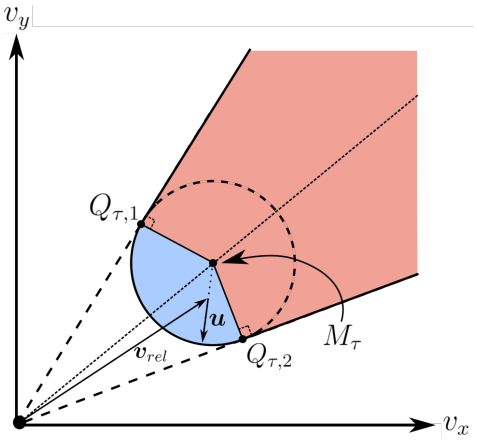
$$\boldsymbol{u} := \underset{\boldsymbol{v} \in \partial VO^\tau_{A|B}}{\operatorname{argmin}} \|\boldsymbol{v} - \boldsymbol{v}_{rel}\| - \boldsymbol{v}_{rel}. \tag{22}$$

Den sökta hastigheten på randen kan befinna sig på antingen någon av de två tangentlinjerna eller på den cirkelbåge som kopplar samman de två linjerna, se figur 7.

Tangentlinjernas ekvationer beräknas genom att använda likformiga trianglar och inledningsvis hitta koordinaterna för tangeringspunkten på cirkeln för respektive linje. Låt $\boldsymbol{Q}_\tau$ vara punkten mitt emellan de två tangeringspunkterna $\boldsymbol{Q}_{\tau,1}$ och $\boldsymbol{Q}_{\tau,2}$, se figur 8. Då fås att



(a) Den punkt på randen som är närmast $\boldsymbol{v}_{rel}$ ligger på första tangentlinjen. Vektor $\boldsymbol{u}$ blir vinkelrät mot tangentlinjen.



(b) Här hamnar $\boldsymbol{v}_{rel}$ i cirkelsektorn närmast origo (se blåmarkerat område). Vektor $\boldsymbol{u}$ blir då i radial riktning.

Figur 7: Vektorn $\boldsymbol{u}$ utgör vektorn från relativa hastigheten $\boldsymbol{v}_{rel}$ till närmsta punkt på randen av $VO^{\tau}_{A|B}$.

$$\begin{aligned} \frac{\|\boldsymbol{M}_\tau - \boldsymbol{Q}_\tau\|}{R_\tau} &= \frac{R_\tau}{\|\boldsymbol{M}_\tau\|} \\ \Rightarrow \|\boldsymbol{M}_\tau - \boldsymbol{Q}_\tau\| &= \frac{R_\tau^2}{\|\boldsymbol{M}_\tau\|} \end{aligned} \tag{23}$$

där $\|\boldsymbol{M}_\tau - \boldsymbol{Q}_\tau\|$ är längden av vektorn från $\boldsymbol{Q}_\tau$ till $\boldsymbol{M}_\tau$. Utifrån detta erhålls

$$\boldsymbol{Q}_\tau = \boldsymbol{M}_\tau + \frac{R_\tau^2}{\|\boldsymbol{M}_\tau\|} \cdot \left(-\frac{\boldsymbol{M}_\tau}{\|\boldsymbol{M}_\tau\|} \right) = \boldsymbol{M}_\tau \left(1 - \frac{R_\tau^2}{\|\boldsymbol{M}_\tau\|^2} \right). \tag{24}$$

Låt $\boldsymbol{Q}_{\tau,v}$ vara vektorn från $\boldsymbol{Q}_\tau$ till $\boldsymbol{Q}_{\tau,1}$. Likformighet ger då

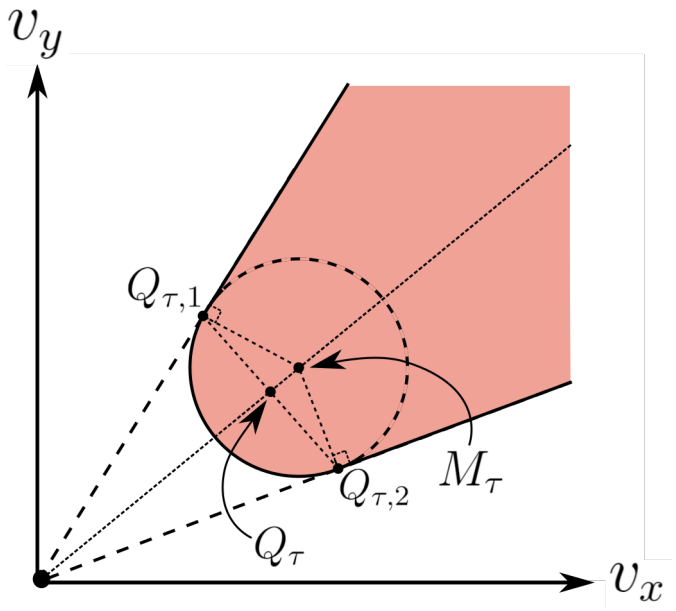
$$\begin{aligned} \|\boldsymbol{Q}_{\tau,v}\| &= \frac{\sqrt{\|\boldsymbol{M}_\tau\|^2 - R_\tau^2}}{\|\boldsymbol{M}_\tau\|} \cdot R_\tau, \\ \boldsymbol{Q}_{\tau,v} &= \|\boldsymbol{Q}_{\tau,v}\| \cdot \frac{\boldsymbol{M}_\tau}{\|\boldsymbol{M}_\tau\|} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \end{aligned} \tag{25}$$

där $\|\boldsymbol{Q}_{\tau,v}\|$ är längden av vektorn och $\boldsymbol{Q}_{\tau,v}$ är vinkelrät mot ortsvektorn $\boldsymbol{M}_\tau$. Detta ger punkterna $\boldsymbol{Q}_{\tau,1}$ och $\boldsymbol{Q}_{\tau,2}$,

$$\boldsymbol{Q}_{\tau,1} = \boldsymbol{Q}_\tau + \boldsymbol{Q}_{\tau,v}, \quad \boldsymbol{Q}_{\tau,2} = \boldsymbol{Q}_\tau - \boldsymbol{Q}_{\tau,v}. \tag{26}$$

Normalvektorn ut ur $VO^{\tau}_{A|B}$ på den del av randen som ligger på tangentlinjen fås av

$$\hat{\boldsymbol{n}}_i = \frac{\boldsymbol{Q}_{\tau,i} - \boldsymbol{M}_\tau}{\|\boldsymbol{Q}_{\tau,i} - \boldsymbol{M}_\tau\|}, \quad i = 1, 2. \tag{27}$$



Figur 8: Lägsta delen av $VO^{\tau}_{A|B}$ (färgat område) där konen klipps av disken med mittpunkt i $\boldsymbol{M}_\tau$. $\boldsymbol{Q}_{\tau,1}$ och $\boldsymbol{Q}_{\tau,2}$ är tangeringspunkter för respektive tangentlinje och $\boldsymbol{Q}_\tau$ utgör punkten mitt emellan tangeringspunkterna.

Vid beräkning av vektorn $\boldsymbol{u}$ beräknas först avstånd till den punkt på respektive tangentlinje som är närmast den relativa hastigheten $\boldsymbol{v}_{rel}$. Dessa punkter finns i normalriktningen från punkten till respektive linje,

$$\begin{aligned} d_i &= \|(\boldsymbol{Q}_{\tau,i} - \boldsymbol{v}_{rel}) \cdot \hat{\boldsymbol{n}}_i\|, \\ \boldsymbol{P}_i &= \boldsymbol{v}_{rel} + d_i \hat{\boldsymbol{n}}_i, \end{aligned} \tag{28}$$

där $\boldsymbol{P}_i$, $i = 1, 2$, är de sökta punkterna och d_i är minsta avståndet från relativa hastigheten till respektive tangentlinje. Betrakta $\boldsymbol{P}_j$ sådant att $d_j \leq d_k$ för $j, k = 1, 2$ och $j \neq k$. Om

$$(\boldsymbol{P}_j - \boldsymbol{Q}_{\tau,j}) \cdot \boldsymbol{Q}_{\tau,j} > 0 \tag{29}$$

är $\boldsymbol{P}_j$ på randen av $VO^{\tau}_{A|B}$ och är därmed den sökta punkten som ligger närmast relativa hastigheten. Detta ger

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{P}_j - \boldsymbol{v}_{rel}, \\ \hat{\boldsymbol{n}} &= \hat{\boldsymbol{n}}_j. \end{aligned} \tag{30}$$

Om olikheten i ekvation (29) inte gäller kontrolleras om olikheten gäller för den andra tangeringspunkten $\boldsymbol{P}_k$, $k = 1, 2$, $k \neq j$. Om den gäller väljs

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{P}_k - \boldsymbol{v}_{rel}, \\ \hat{\boldsymbol{n}} &= \hat{\boldsymbol{n}}_k. \end{aligned} \tag{31}$$

Om så inte är fallet beräknas istället $\boldsymbol{u}$ som den vektor som går från relativa hastigheten till den närmaste punkten på randen av disken $D(\boldsymbol{M}_\tau, R_\tau)$,

$$\begin{aligned} \hat{\boldsymbol{n}} &= \frac{\boldsymbol{v}_{rel} - \boldsymbol{M}_\tau}{\|\boldsymbol{v}_{rel} - \boldsymbol{M}_\tau\|}, \\ \boldsymbol{u} &= \boldsymbol{M}_\tau + R_\tau \hat{\boldsymbol{n}} - \boldsymbol{v}_{rel}. \end{aligned} \tag{32}$$

4.6 Hitta bästa tillåtna hastigheten

För att undvika kollision måste relativa hastigheten förändras med vektorn $\boldsymbol{u}$, så att nya relativa hastigheten blir $\boldsymbol{v}_{rel} + \boldsymbol{u}$ och därmed hamnar på randen av $VO^\tau_{A|B}$. $VO^\tau_{A|B}$ är symmetrisk i origo med $VO^\tau_{B|A}$ och får därmed en förändringsvektor för robot B givet robot A som är lika stor och i motsatt riktning mot förändringsvektorn för robot A givet robot B [8]. Genom att anta att andra robotar använder samma kollisionsundvikningsalgoritm kan robotarna addera halva $\boldsymbol{u}$ till sin egen nuvarande hastighet och ändå garantera att risken för kollision minimeras, enligt

$$\boldsymbol{v}_{rel} + \boldsymbol{u} = \boldsymbol{v}_A - \boldsymbol{v}_B + \boldsymbol{u} = \left(\boldsymbol{v}_A + \frac{\boldsymbol{u}}{2}\right) - \left(\boldsymbol{v}_B - \frac{\boldsymbol{u}}{2}\right). \tag{33}$$

Vid fasta objekt ska istället hela $\boldsymbol{u}$ adderas eftersom dessa inte bidrar till kollisionsundvikningen. Samma princip följs när följarrobotarna undviker ledaren, då ledaren inte bidrar till kollisionsundvikningen.

När halva $\boldsymbol{u}$-vektorn adderats till robot As nuvarande hastighet $\boldsymbol{v}_A$ dras i denna punkt en linje vinkelrät mot $\boldsymbol{u}$, se figur 9. Tillåtna hastigheter definieras nu som det halvplan av hastigheter som befinner sig på den sidan av linjen som är i positiv $\hat{\boldsymbol{n}}$-riktning,

$$ORCA^\tau_{A|B} = \left\{\boldsymbol{v} \middle| \left(\boldsymbol{v} - \left(\boldsymbol{v}_A + \frac{1}{2}\boldsymbol{u}\right)\right) \cdot \hat{\boldsymbol{n}} \geq 0\right\}, \tag{34}$$

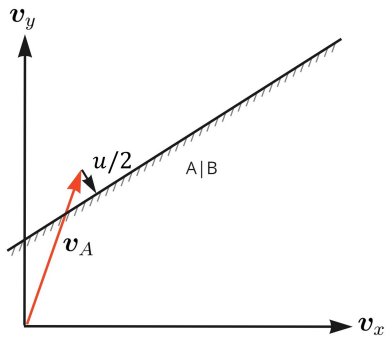
där $ORCA^\tau_{A|B}$ står för *Optimal Reciprocal Collision Avoidance* (fritt översatt: Optimal ömsesidig kollisionsundvikning) för robot A givet robot B inom tiden τ. Det existerar andra val av områden som skulle garantera kollisionsundvikning, men detta val av område är det med flest hastigheter nära nuvarande hastigheten och är även ett rättvist val eftersom alla robotar tvingas göra lika mycket för att undvika kollision [8].

$ORCA^\tau_{A|B}$ beräknas för varje annan robot B i systemet, och alla tillåtna hastigheter kommer då bilda ett konvext område,

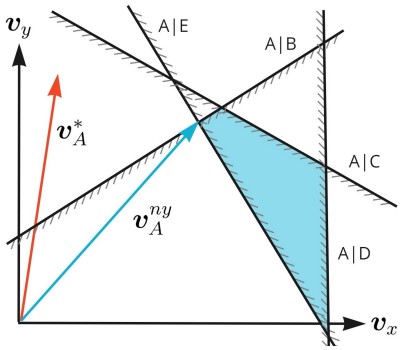
$$ORCA^\tau_A = D(\mathbf{0}, v_A^{max}) \cap \bigcap_{B \neq A} ORCA^\tau_{A|B}, \tag{35}$$

där det alltid kommer att finnas en hastighet i hastighetsmängden som är närmast den önskade hastigheten för robot A [8]. Den tillåtna hastighet som är närmast robot As önskade hastighet väljs till den nya hastigheten, se figur 10.

För att hitta denna hastighet används en optimeringsalgoritm baserad på [11]. Först undersöks om önskade farten är lägre än maxfarten, och annars ändras önskade hastigheten till maxfart med tidigare önskad riktning. Därefter adderas ett $ORCA^\tau_{A|B}$ i



Figur 9: En linje dras vinkelrät mot vektor $\boldsymbol{u}$ genom den punkt som finns vid $\boldsymbol{v}_A + \frac{\boldsymbol{u}}{2}$. Den streckade sidan ligger i riktning $\hat{n}$ från linjen och visar det halvplan med hastigheter som är tillåtna för robot A att anta givet robot B.



Figur 10: Det blå området visar den mängd med hastigheter som är tillåtna för robot A givet alla andra robotar i systemet. Den tillåtna hastigheten närmast den önskade hastigheten $\boldsymbol{v}_A^*$ väljs till den nya hastigheten, $\boldsymbol{v}_A^{ny}$.

taget, med slumpvis ordning på B för att förenkla beräkningarna [11], och den bästa tillåtna hastigheten givet de halvplan som hittills blivit tillagda beräknas. Även maxfarten tas hänsyn till i varje steg genom att kontrollera att farten aldrig överstiger maxfarten. Ett nytt B slumpas och nästa $ORCA^{\tau}_{A|B}$ läggs till. När alla halvplan har lagts till är den bästa tillåtna hastigheten hittad med avseende på hela $ORCA^{\tau}_{A}$.

När ett nytt halvplan adderas finns tre möjligheter. Den tillåtna mängden kan bestå, minska eller försvinna helt. Först kontrolleras därför om den tidigare bästa hastigheten ligger i det nya halvplanet. Om så är fallet behålls det gamla valet av hastighet.

Om den tidigare bästa hastigheten inte är tillåten måste den gamla bästa hastigheten ha blivit bortskuren av det nya halvplanet och den nya bästa hastigheten måste ligga på randen till det nya halvplanet [11]. Därför beräknas först den punkt på randen av det nya halvplanet som är närmast den önskade hastigheten. Om denna punkt ligger i alla tidigare tillagda halvplan så är denna den nya bästa tillåtna hastigheten. I annat fall beräknas alla skärningspunkter mellan randen av nya $ORCA$-halvplanet och randen av alla tidigare halvplan. Varje skärningspunkt undersöks för att hitta det tillåtna intervallet på randen av det nya halvplanet, inramat av två skärningspunkter. Den ändpunkt på intervallet som befinner sig närmast den önskade hastigheten väljs till den nya bästa tillåtna hastigheten.

I fall där den nya bästa hastigheten har högre fart än maxfarten måste den justeras. Om det finns någon hastighet inom det tillåtna intervallet på randen av det nya halvplanet som har lägre fart än maxfart väljs den av de hastigheterna som ligger närmast den hittade bästa hastigheten, vilket kommer vara en hastighet med maxfart.

4.7 Kompakta situationer

Det kan uppstå fall då ingen tillåten hastighet finns, antingen för att halvplanen inte sammanfaller eller för att de enda hastigheterna som garanterar kollisionsfri körning har en fart över maxfart. I dessa fall väljs den säkraste möjliga hastigheten genom att minimera maximala avståndet till randen på alla halvplan och på så vis förhindra de värsta krockarna.

Låt $d_{A|B}(\boldsymbol{v})$ vara avståndet från hastigheten $\boldsymbol{v}$ till närmsta punkt på randen av $ORCA^{\tau}_{A|B}$, sådant att $d_{A|B}(\boldsymbol{v})$ är negativt om $\boldsymbol{v}$ är inuti området och annars positivt. Den säkraste hastigheten $\boldsymbol{v}^{ny}_A$ är den hastighet som minimerar det största avståndet till randen av $ORCA_{A|B}\tau$ för alla B,vilket skrivs som

$$\boldsymbol{v}^{ny}_A = \underset{\boldsymbol{v}\in D\left(\mathbf{0},v^{max}_A\right)}{\operatorname{argmin}} \left(\max_{B\neq A} d_{A|B}(\boldsymbol{v})\right). \tag{36}$$

Avståndet $d_{A|B}(\boldsymbol{v})$ kan ses som höjden av ett plan som skär hastighetsplanet i den linje som utgör randen till $ORCA^{\tau}_{A|B}$. För att minimera det maximala avståndet till randen av alla $ORCA$-halvplan kan detta översättas till att hitta den punkt i tre dimensioner som minimerar höjden och samtidigt ligger över alla plan. Den nya hastigheten blir den som fås av att projicera ner den punkt som hittats på hastighetsplanet [8]. För att hitta denna punkt används en metod inspirerad av den metod som användes för att hitta den bästa tillåtna hastigheten i 4.6, men anpassad till att hitta punkten längst bort i en viss riktning snarare än närmast en annan specificerad punkt [11].

Ett plan läggs till i taget i slumpvis ordning och den säkraste hastigheten beräknas iterativt för de hittills tillagda planen. Första planet som läggs till genererar en hastighet som är den punkt på randen av motsvarande $ORCA$-halvplan som har lägst fart. Om denna fart är över maxfart väljs istället den hastighet som är i samma riktning men har maxfart istället.

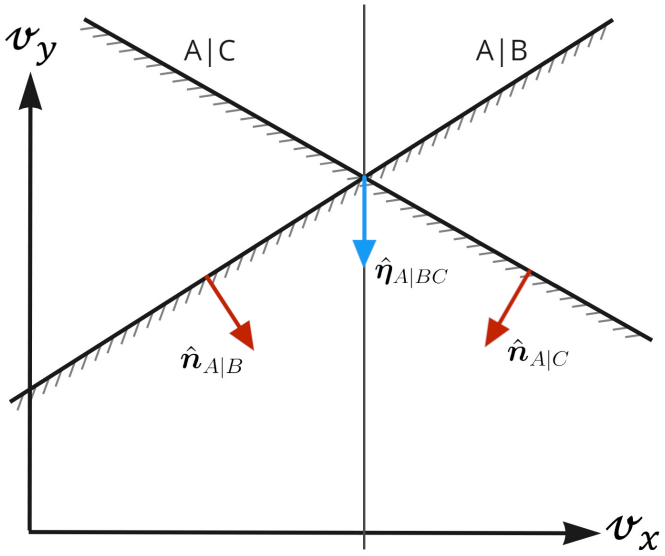
När ett nytt plan läggs till kan höjden i v_z-led endast höjas över områden, inte sänkas, eftersom alla tidigare plan finns kvar. Om höjden för att befinna sig över alla plan inte har höjts vid den tidigare lägsta punkten behålls valet av säkrast möjliga hastighet som innan. Annars behöver en ny beräkning av den säkraste punkten utföras.

Skärningslinjer mellan det nya planet och alla andra plan beräknas. Dessa projiceras ner på hastighetsplanet och kan där parametriseras som en punkt och en riktningsvektor. Punkten utgörs av skärningspunkten mellan randen på det nya $ORCA$-halvplanet och randen av det tidigare tillagda $ORCA$-halvplanet, och riktningsvektorn definieras som

$$\hat{\boldsymbol{\eta}}_{A|BC} = \frac{\hat{\boldsymbol{n}}_{A|B} + \hat{\boldsymbol{n}}_{A|C}}{\|\hat{\boldsymbol{n}}_{A|B} + \hat{\boldsymbol{n}}_{A|C}\|} \tag{37}$$

där $\hat{\boldsymbol{n}}_{A|B}$ och $\hat{\boldsymbol{n}}_{A|C}$ utgör vektorn in i det nya halvplanet $ORCA^{\tau}_{A|B}$ samt det tidigare

tillagda halvplanet $ORCA^{\tau}_{A|C}$ för alla $C \neq B$, vinkelräta mot randen av respektive område, se figur 11.



Figur 11: Skärningslinjen mellan de plan som går genom randen av $ORCA^{\tau}_{A|B}$ och $ORCA^{\tau}_{A|C}$, projicerat ner på hastighetsplanet. $\hat{\boldsymbol{\eta}}_{A|BC}$ utgör riktningsvektorn för skärningslinjen.

När två eller fler plan har lagts till finns nästan alltid endast en minpunkt. Enda undantaget är när randen av två eller fler plans motsvarande $ORCA$-halvplan är parallella och en del av deras skärningslinje istället utgör minimala höjden. Eftersom alla plan sträcker sig mot oändligheten kan endast en sänka skapas där minpunkten finns i botten av sänkan. Om den tidigare minpunkten skärs bort betyder det att botten av sänkan har höjts och den nya minpunkten kommer befinna sig på den nya botten, som kommer utgöras av det nya planet begränsat av dess skärningslinjer med tidigare tillagda plan samt eventuellt begränsningen av maxfart. Den punkten som har lägst höjd på det nya planet av alla de punkter som ingår i den nya botten är den nya minpunkten.

Den tidigare minpunktens projicering på hastighetsplanet kommer fortfarande befinna sig inuti det tillåtna området när det projiceras på hastighetsplanet, men inte längre utgöra den säkraste punkten. Normalriktningen för skärningslinjen definieras som vinkelrät mot riktningsvektorn $\hat{\boldsymbol{\eta}}_{A|BC}$ och riktad mot den sida som tidigare valda säkraste hastigheten finns på och därmed alltid in mot området som utgör sänkans botten.

Nu kan algoritmen från 4.6 översättas till att behandla dessa skärningslinjer istället för randen av halvplanen som tidigare. Istället för att eftersöka punkten närmast en önskad punkt, eftersöks istället en punkt som har lägst höjd. Denna kommer finnas i negativa gradientriktningen för det nya planet och begränsas av maxfarten och skärningslinjerna. För att säkerställa att maxhastigheten beaktas väljs maxfart i negativa gradientriktningen som första hastighet. Därefter adderas en skärningslinje

i taget. Om den tidigare valda nya hastigheten blir ogiltig väljs en ny på den nya linjen enligt algoritmen i 4.6. Om det bästa valet egentligen har en fart högre än maxfart väljs istället den punkt på nya skärningslinjen som är i maxfart och ligger närmast den som ville väljas.

4.8 Val av tidsspann för kollisionsundvikning

Variabeln τ spelar en central roll i metoden för kollisionsundvikning. Den avgör hur långt in i framtiden kollisioner bör undvikas och därav hur försiktigt robotarna kör. En metod för att välja ett rimligt värde på τ, utifrån vissa antaganden, presenteras i det här avsnittet.

Den beskrivna metoden för att undvika kollisioner beräknar en hastighet för varje robot som garanterar att inga kollisioner sker inom tiden τ, däremot är det sällan utanför en simulerad miljö som hastigheten kan styras direkt. Oftast erhålls direkt kontroll över en robots acceleration som i ett senare skede påverkar hastigheten. På grund av detta måste värdet på τ väljas tillräckligt högt för att robotarna ska kunna förutse om en ändring i hastigheten måste göras och hinna utföra den ändringen innan kollisionen sker. Det som förhindrar τ från att väljas till ett godtyckligt högt tal är faktumet att den beskrivna metoden beräknar tillåtna hastigheter utifrån att robotarna bibehåller sina nuvarande hastigheter tiden τ framåt. Om τ väljs för högt kommer systemet agera för att undvika kollisioner tidigare än nödvändigt vilket felklassificerar kollisionsfria och tidseffektiva hastigheter som otillåtna.

Eftersom undvikande av kollisioner prioriteras ska ett minimivärde τ_{min} beräknas. Ett lämpligt värde på τ blir ett som är både större än τ_{min} och lågt nog att störningen på robotarnas beteende är acceptabelt. Av den anledningen är valet av värdet på τ lika mycket ett designproblem som ett matematiskt sådant.

Den lägsta tillåtna τ är den som gör att en robot som måste utföra den högsta möjliga hastighetsändringen hinner göra det innan den kolliderar. Situationen som kräver den högsta möjliga hastighetsändringen är då två robotar A och B åker i högsta möjliga fart mot varandra, där robot B inte kan sakta ner och varken robot A eller robot B kan väja för den andra eftersom det skulle orsaka kollisioner med andra robotar i systemet. Då måste robot A hinna vända och köra i maxfart i motsatt riktning innan den kolliderar med robot B. Robot A kan först detektera robot B då avståndet till den är lika eller mindre än s_{max}, se ekvation (21). Låt robot B börja i origo med riktning i positivt x-led och farten v_{max}. Dess position respektive fart längs x-axeln vid tiden 0 kan uttryckas som $p_B(0) = 0$ och $v_B(0) = v_{max}$. Låt samtidigt robot A börja i positionen s_{max} längs x-axel med farten v_{max} och riktningen i negativ x-led, då kan dess position och fart längs x-axeln vid tiden 0 beskrivas som $p_A(0) = s_{max}$ och $v_A(0) = -v_{max}$. Det vill säga, robotarna åker i maxfart rakt mot varandra.

Antag att robot A tillämpar en konstant acceleration $a_A(t) = a_{max}$ i positivt x-led,

då kan dess fart beskrivas som

$$v_A(t) = v_A(0) + \int_0^t a_A(T)dT = -v_{max} + a_{max}t. \tag{38}$$

Antag att robot B åker i konstant hastighet. Robotarnas positioner i x-led över tid kan då uttryckas som

$$p_A(t) = p_A(0) + \int_0^t v_A(T)dT = 2\tau v_{max} + r_A + r_B - v_{max}t + a_{max}\frac{t^2}{2}, \tag{39}$$

$$p_B(t) = p_B(0) + \int_0^t v_B(T)dT = v_{max}t. \tag{40}$$

För att en kollision inte ska ske inom önskad tid τ måste skillnaden i robotarnas positioner alltid vara minst $r_A + r_B$. Eftersom båda robotar befinner sig på x-axeln fås

$$p_A(t) - p_B(t) = 2\tau v_{max} + r_A + r_B + a_{max}\frac{t^2}{2} - 2v_{max}t \geq r_A + r_B \quad \forall t \in [0, \tau]. \tag{41}$$

Detta kan skrivas om för att ge en undre gräns för τ,

$$\tau \geq t - \frac{a_{max}t^2}{4v_{max}}, \quad \forall t \geq 0. \tag{42}$$

Då τ är en konstant behöver den sättas till att vara oberoende av tiden t. Därmed erhålls

$$\tau_{min} = \max_{t\in[0,\tau]} \left(t - \frac{a_{max}t^2}{4v_{max}} \right). \tag{43}$$

Maximeringsproblemet löses genom att hitta en tid $\bar{t}$ då $\frac{d}{dt}\left(t - \frac{a_{max}t^2}{4v_{max}}\right) = 0$,

$$\frac{d}{dt}\left(t - \frac{a_{max}t^2}{4v_{max}} \right)\bigg|_{t=\bar{t}} = 1 - \frac{a_{max}\bar{t}}{2v_{max}} = 0 \Rightarrow \bar{t} = 2\frac{v_{max}}{a_{max}}, \tag{44}$$

$$\Rightarrow \tau_{min} = 2\frac{v_{max}}{a_{max}} - \frac{v_{max}}{a_{max}} = \frac{v_{max}}{a_{max}}. \tag{45}$$

För detta τ_{min} har endast accelerationen tagits hänsyn till. Denna undre gräns gäller då roboten har förmåga att backa. Om roboten endast kan ha positiv hastighet måste den vända ett halvt varv, π radianer, för att kunna köra åt motsatt håll. Därmed måste även tiden det tar att vända räknas med. Det är viktigt att vändningen blir exakt π radianer och går så snabbt som möjligt. Enligt ekvation (5) är det teoretiska ω_{max} oändligt när hastigheten är 0. Därmed begränsas ω endast av dess maximala fysikaliska vinkelhastighet Ω_{max} och vändtiden blir

$$t_\pi = \frac{\pi}{\Omega_{max}}. \tag{46}$$

Den maximala vinkelaccelerationen, α_{max}, antas vara såpass hög att dess påverkan är försumbar. Det nya värdet på τ_{min} blir därmed

$$\tau_{min} = \frac{v_{max}}{a_{max}} + t_\pi. \tag{47}$$

Ett problem som uppstår då detektering av andra robotar sker med en frekvens f, som till exempel en sensors utsignalsfrekvens, är att avståndet från en robot till en annan då de detekterar varandra kan vara mindre än s_{max}. Om det händer hinner inte robot A vända innan den kolliderar med B. Låt f_{min} vara den lägsta detekteringsfrekvensen, vilket ger att den högsta tiden mellan två detekteringar blir $t_{d,max} = 1/f_{min}$. Robotarna A och B hinner maximalt färdas avståndet $t_{d,max}v_{max} = \frac{v_{max}}{f_{min}}$ på tiden $t_{d,max}$. Därmed fås ett nytt s_{max} som tar hänsyn till detekteringsfrekvensen genom

$$s_{max} = 2\tau v_{max} + r_A + r_B - 2\frac{v_{max}}{f_{min}}. \tag{48}$$

Det nya s_{max} insatt i robot As startposition $p_{A,0}$ ger ett nytt uttryck för τ_{min} som tar hänsyn till en detekteringsfrekvens,

$$\tau_{min} = \frac{v_{max}}{a_{max}} + t_\pi + \frac{1}{f_{min}}. \tag{49}$$

4.9 Implementering av kollisionsundvikning i styrsystemet

Hittills i kapitel 4 har en algoritm för undvikning av kollisioner utvecklats. För att algoritmen ska komma till användning så måste den integreras med styrsystemet som har utvecklats i kapitel 3.

I styrsystemet beräknas en acceleration och vinkelhastighet som behövs för att styra roboten till en önskad position. Eftersom kollisionsundvikningen bygger på att evaluera en önskad hastighetsvektor och redigera denna utefter vad som krävs för att undvika kollisioner så måste accelerationen och vinkelhastigheten ur styrsystemet integreras och adderas till robotens nuvarande fart och riktning för att få den nya önskade farten och riktningen. Detta konverteras till x- och y-komponenter och sedan skickas till kollisionsundvikningsalgoritmen, var efter fås den redigerade hastighetsvektorn som undviker kollisioner. Denna vektor konverteras tillbaka till en fart och riktning, deriveras för att få den önskade accelerationen och vinkelhastigheten och skickas ut i systemet för att förverkligas.

5 Simulering

I detta kapitel redovisas och diskuteras resultatet från varje del av projektet i form av simuleringar. Dessa utförs utifrån ett system av robotar som är baserade på Sphero BOLT [12], som är en kommersiell sfärisk robot.

5.1 Simuleringsmetodik

Simuleringsprogrammet är skrivet i MATLAB R2018b med en Akademisk licens. Filmer på simulationer finns tillgängliga på `https://bit.ly/2Z9byEb`. Kod för simuleringsprogrammet innehållande styrning, kollisionsundvikning samt formationsbildning kan hämtas på `https://git.io/fjWNG`.

Alla robotar anses ha samma radie, 0.0365 m, vilket är enhetligt med Sphero BOLT [13], och att alla robotar känner till alla andra robotars position och hastighet. Robotarnas maximala fart är 1.5 m/s, vilket väljs för att ha en marginal från den maximala farten på 2 m/s för en Sphero BOLT. I de flesta verkliga fall varierar maximala momentet en motor kan leverera beroende på bland annat varvtalet. I robotarnas fall översätts detta till att maximala accelerationen varierar beroende på farten. Robotarna i simuleringen har däremot en konstant maximal acceleration oavsett vilken fart de har, till följd av förenkling och avsaknad av relevant data. Denna acceleration väljs till 0.7 $\mathrm{m/s^2}$. Utöver detta väljs den maximala accelerationen i fartregleringen, a_{begr} som beskrivs i avsnitt 3.1, till 0.35 $\mathrm{m/s^2}$. Anledningen till detta är att en robot som är långt ifrån sin önskade position, och därav vill öka farten med den maximala tillgängliga accelerationen a_{begr}, alltid ska ha ett ω_{max} som är större än noll så att den kan svänga.

För att underlätta olika test har ledaren i simuleringen en fristående rörelse som satts till en fördefinierad hastighetsfunktion, opåverkad av styrnings- och kollisionsundvikningsalgoritmerna.

I en verklig situation beror antalet beräkningar som varje enskild robot måste utföra på den unika situationen roboten befinner sig i. Till följd av detta är det högst osannolikt att alla robotar skulle utföra sina beräkningar synkront med varandra. Till skillnad från verkliga fall underlättas simuleringen genom att alla beräkningar utförs synkront då det inte anses påverka resultatet noterbart. Simuleringsberäkningarna görs med en frekvens på 100 Hz.

5.2 Simuleringsresultat för styrning och dynamik

Som beskrivet i kapitel 3 använder roboten två reglersystem för att styras, en till farten och en till riktningen. Dessa regulatorer har konstanter som påverkar robotens beteende och har därför valts utifrån reglersystemens stegsvar.

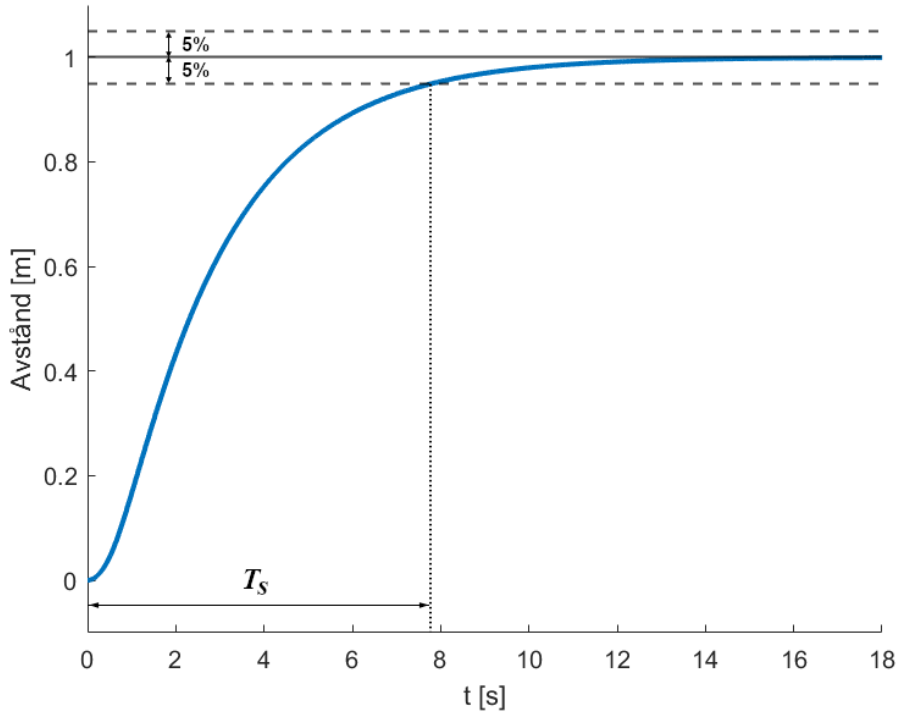
Tabell 1: Simuleringens parametrar

Variabel	Värde	Enhet	Förklaring
a_{max}	0.7	m/s^2	Maximal acceleration
a_{begr}	0.35	m/s^2	Begränsning av acceleration
Δt	0.01	s	Tidssteg mellan beräkning av ny hastighet
$K_{d,s}$	0.4	-	Regulatorparameter för deriverande delen av avståndsregulatorn
$K_{p,s}$	0.5	-	Regulatorparameter för proportionella delen av avståndsregulatorn
$K_{p,v}$	2	-	Regulatorparameter för proportionella delen av fartregulatorn
$K_{p,\theta}$	4	-	Regulatorparameter för proportionella delen av vinkelregulatorn
r	0.0365	m	Radie för robot
t_f	0.4	s	Tidskonstant
v_{max}	1.5	m/s	Maximal fart
α_{max}	90	rad/s^2	Maximal vinkelacceleration
τ	2.48	s	Maximala tid framåt som kollisionsundvikning hanteras
ω_{max}	10	rad/s	Maximal vinkelhastighet

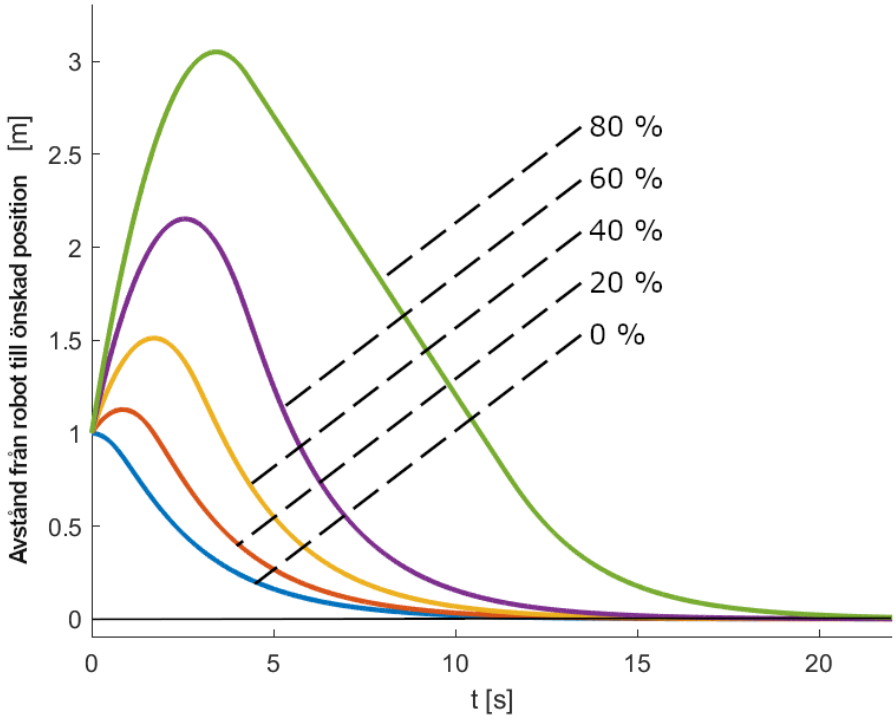
Vid valet av konstanterna till systemet som reglerar robotens avstånd till en önskad position, vilket beskrivs i avsnitt 3.2.2, efterfrågas en försiktig reglering som undviker översläng. Anledningen till detta är för att roboten måste vända och närma sig den önskade positionen på nytt då översläng sker eftersom den inte kan backa. Utöver detta hade en avståndsreglering som är snabbare än en riktningsreglering lett till att roboten skjuter förbi den önskade positionen då den inte hinner rikta sig mot den innan den ökar farten vilket som följd begränsar förmågan att svänga, som beskriven i avsnitt 3.1.

För att undvika dessa problem måste robotens riktningsreglering vara snabbare än avståndsregleringen. Ett mått för snabbheten kan vara stegsvarets insvängningstid T_s som är tiden då nuvarande värdet håller sig inom 5 % av det önskade värdet. Avståndsregleringens stegsvar med regulatorkonstanterna som återfinns i tabell 1, visas i figur 12, där de horisontella streckade linjerna motsvarar 5 % av det önskade värdet på 1 m. Med de valda konstanterna är avståndsregleringens insvängningstid cirka $T_s = 7.8$ s vilket är långsamt relativt vad systemet klarar av, men enhetligt med det som önskas av den.

Trots robotens relativt höga insvängningstid elimineras det kvarstående felet inom en rimlig tidsram. I figur 13 visas robotens avstånd till en önskad position som börjar 1 m framför roboten och ändras med en konstant fart på en procentsats av robotens maximala fart i riktningen ifrån roboten. Kurvan som motsvarar att den önskade positionen ändras med 80 % av robotens maximala fart antar en linjär form mellan cirka 3-9 sekunder eftersom roboten når sin maximala fart vilket gör att ändringen i avståndet blir konstant. Efter cirka 20 s har roboten kommit ikapp den önskade positionen. Då den önskade positionens ändring närmar sig 100 % av robotens maxfart går tiden det tar för roboten att nå den önskade positionen mot oändligheten, eftersom roboten inte kan öka sin fart tillräckligt för att minska felet.

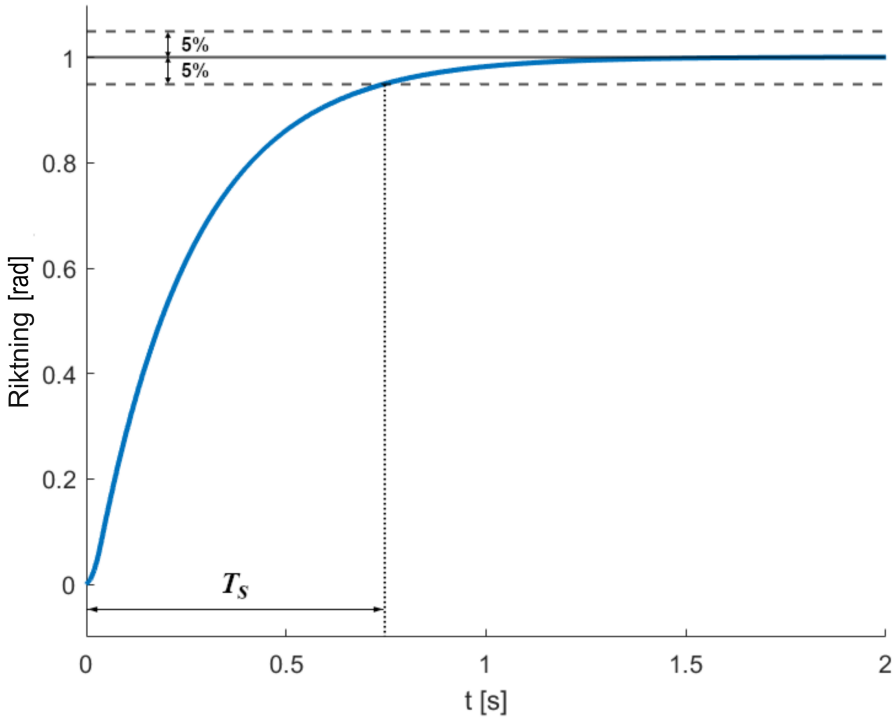


Figur 12: Stegsvar för en robot som förflyttar sig från sin startposition till en önskad position en meter bort, där y-axeln beskriver avståndet från startpositionen och x-axeln tiden som förflutit. Insvängningstiden T_s är cirka 7.8 s, vilket är tiden det tar för roboten att komma inom 5 % av den önskade förflyttningen.



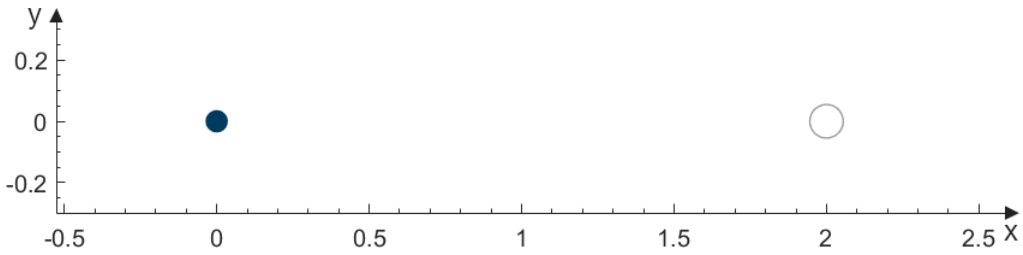
Figur 13: Robotens avstånd till den önskade positionen då den önskade positionen förflyttas med motsvarande 0 %, 20 %, 40 %, 60 % respektive 80 % av robotens maxfart. Avståndet ökar först då roboten startar från stillastående och behöver accelerera för att komma ikapp. När roboten når sin maxfart blir grafen linjär, vilket tydligt syns i testet med 80 % av v_{max}.

Regleringen av robotens riktning önskas vara snabbare än avståndsregleringen då det är lönlöst att roboten har en fart om den inte är i riktningen av den önskade positionen. Stegsvaret till riktningens reglersystem visas i figur 14. Insvängningstiden T_s är cirka 0.75 s vilket är cirka 10 gånger kortare än avståndregleringens insvängningstid och i enlighet med det som önskas av förhållandet mellan de två systemen. Detta kan optimeras ytterligare för att få en kortare insvängningstid, men för ändamålet anses detta vara bra nog.

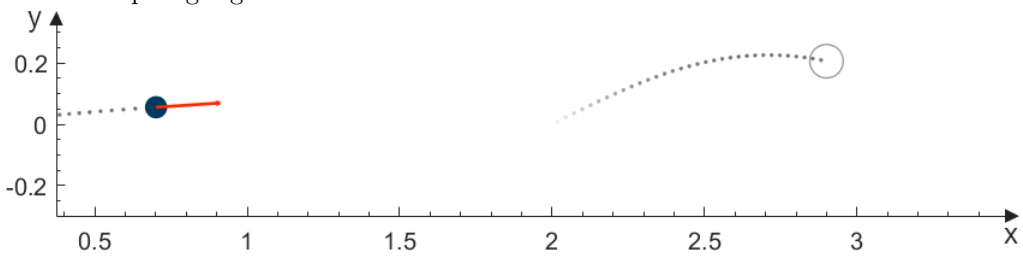


Figur 14: Stegsvar för roboten då den roterar från 0 till 1 rad, där y-axeln visar robotens riktning och x-axeln tiden som förflutit. Insvängningstiden T_s är cirka 0.75 s, vilket är tiden det tar för roboten att komma inom 5 % av den önskade förflyttningen.

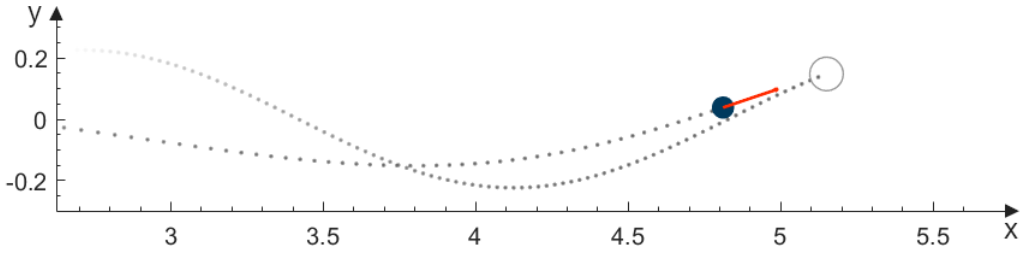
För att visa att de båda reglersystemen fungerar tillsammans simuleras ett test. Den önskade positionen väljs så att den förflyttas enligt ekvationen $0.3v_{max}[1, \frac{\cos t}{2}]$, för att tvinga roboten att alltid reglera både vinkel och fart. Detta test återfinns i figur 15.



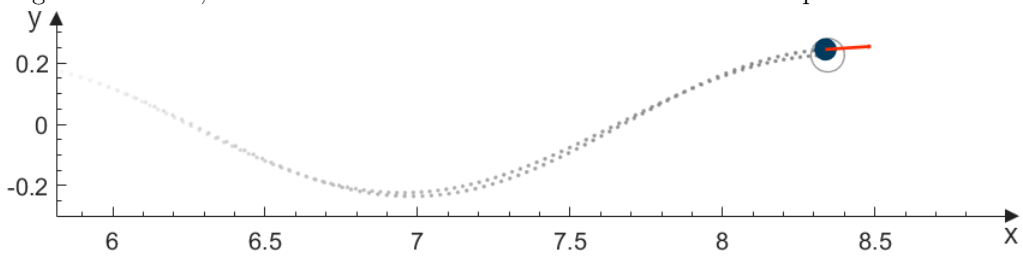
(a) Vid tiden 0 s befinner sig både roboten (ifylld mörkblå cirkel) och den önskade positionen (grå ring) i deras ursprungsläge.



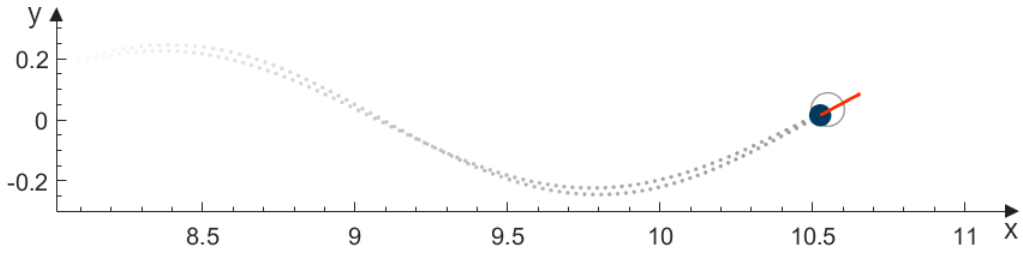
(b) Vid tiden 2 s har robotens båda reglersystemen aktiverats. Den röda pilen ur roboten representerar robotens fart och riktning. Här regleras främst farten då skillnaden i riktningen till den önskade positionen är relativt liten.



(c) Vid tiden 7 s har roboten närmat sig den önskade positionen. Här är det viktigare med regleringen av vinkeln, då den ändras mer när avståndet till den önskade positionen blir mindre.



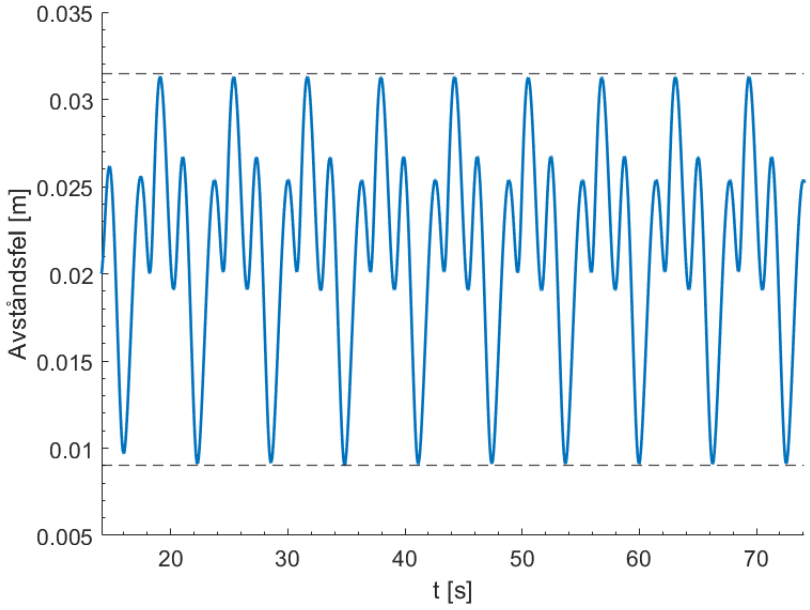
(d) Vid tiden 14.1 s kan roboten anses vara ikapp den önskade positionen och måste nu bibehålla den.



(e) Vid tiden 19 s kan ett kvarstående fel observeras. Detta uppkommer då den önskade positionens hastighetsriktning ändras konstant.

Figur 15: Simulering av en robots förflyttning i xy-planet över tid då den regleras mot en önskad position som förflyttas med hastigheten $0.3v_{max}[1, \frac{1}{2}\cos t]$.

När roboten kommit ikapp en position som rör sig i en olinjär bana, kommer roboten att variera runt den önskade positionen. I testet som illustrerades i figur 15 rör sig den önskade positionen i en sådan bana. För att visa hur felet variera för denna bana har felet plottats i figur 16. Felet håller sig inom ett intervall på 2.03 ± 1.13 cm. Detta kan dock anses som tillräckligt nära den önskade positionen.



Figur 16: Avståndet från roboten till den önskade positionen för testet som illustreras i figur 15. Observera att figuren illustrerar då roboten varierar kring den önskad position. Avståndet förblir inom intervallet 2.03 ± 1.13 cm.

Reglersystemets konstanter har anpassats för att erhålla ett acceptabelt beteende i syfte att kunna demonstrera andra aspekter av arbetet. Exempelvis är avståndsregleringen mycket långsammare än vad den behöver vara, i ett försök att minimera risken för möjliga problem som kan uppstå på grund av det. Krav på robotens beteende avgör hur reglersystemet ställs in och i arbetets syfte fungerar de konstanter som har valts.

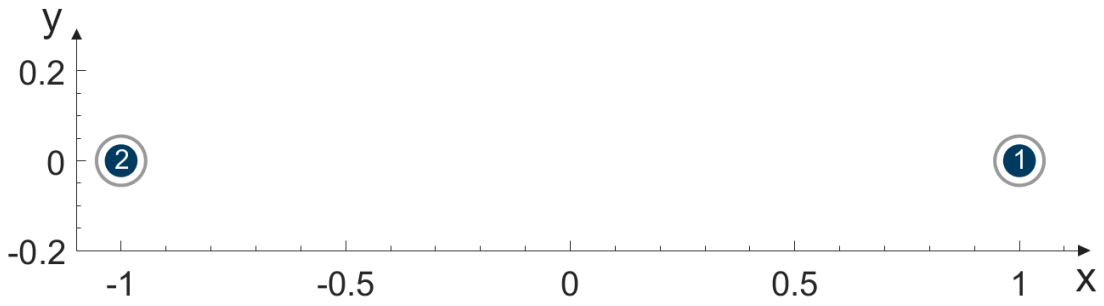
5.3 Simuleringsresultat för kollisionssystem

För att få flera robotar att nå fram till sina mål krävs det att de undviker varandra och ej kolliderar under tiden. Teorin kring detta beskrivs i kapitel 4 och resultatet av det simuleras här.

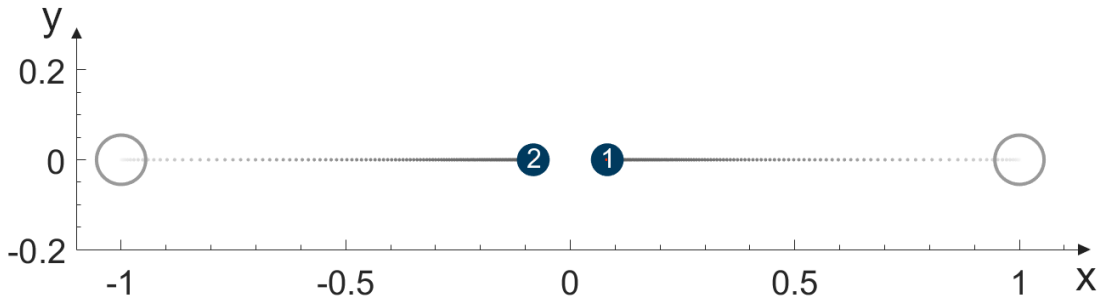
5.3.1 Införande av brus

I simuleringsprogrammet simuleras inte störningar som exempelvis friktion, ojämnt underlag eller luftmotstånd. På grund av detta uppstår situationer där symmetri i systemet orsakar problem.

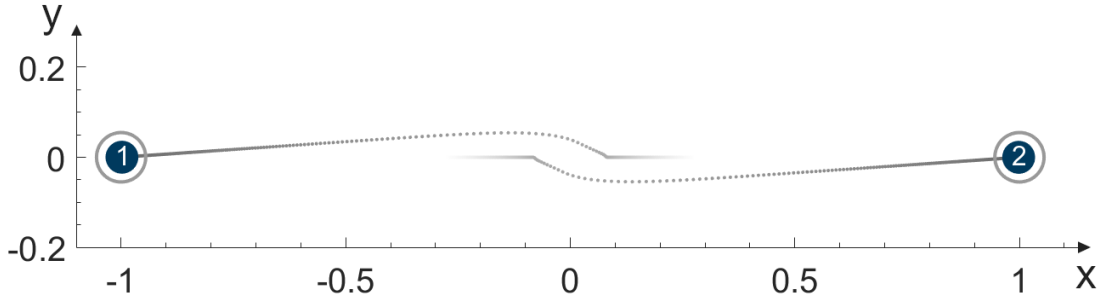
Ett typiskt exempel på detta är då två robotar börjar riktade mot varandra och vill byta plats. Om deras beräkningar sker synkroniserade och utan en slumpmässig inverkan kommer de vid samma tidpunkter ta samma beslut. Detta gör att de teoretiskt inte skulle kunna passera varandra då de speglar varandras beteende, de hamnar i ett dödläge. Däremot, om en skillnad i någon av robotarnas egenskaper existerar så bryts symmetrin. I verkligheten framträder inte liknande symmetri då störningar alltid påverkar robotarna olika så att deras egenskaper skiljer sig från varandra. För att lösa detta i simulationerna tilläggs ett slumpmässigt brus på ± 0.02 rad/s på riktningens styrsignal ur regulatorn, vilket är högt nog för att motverka symmetriska fall men lågt nog för att inte påverka robotens övriga beteende. Påverkan av bruset demonstreras med två simulationer där två robotar byter plats med varandra.



(a) Robot 1 och 2 startar med två meters mellanrum och riktade mot varandra. Deras önskade positioner sätts så att robotarna byter plats.



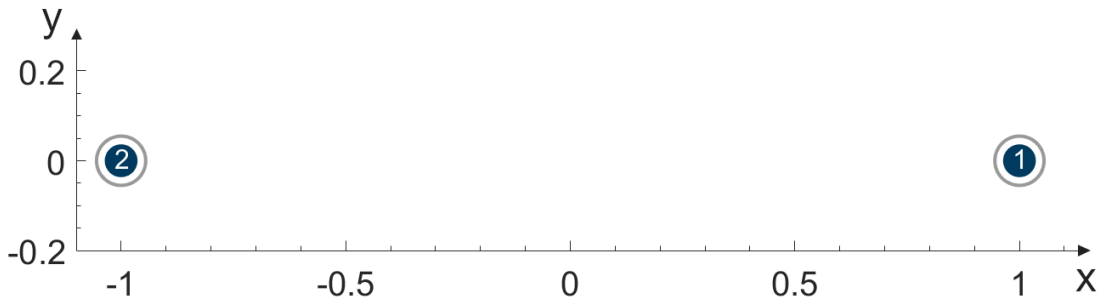
(b) Vid tiden 8 s. Nu har robotarna närmat sig varandra och saktat ner till nära stillastående.



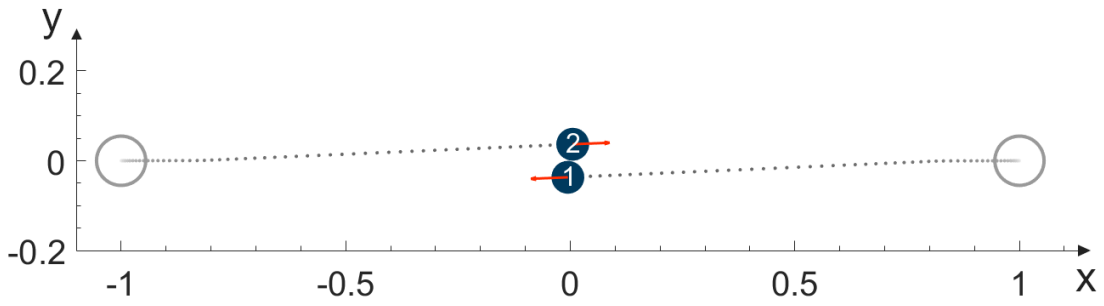
(c) Robotarna har enats om vilken sida de ska passera varandra på och når fram till sina önskade positioner efter 39 s.

Figur 17: Simulering där två robotar byter position utan något brus på riktningens styrsignal. Simuleringen görs för att kunna jämföras med simuleringen med brus, som återfinns i figur 18.

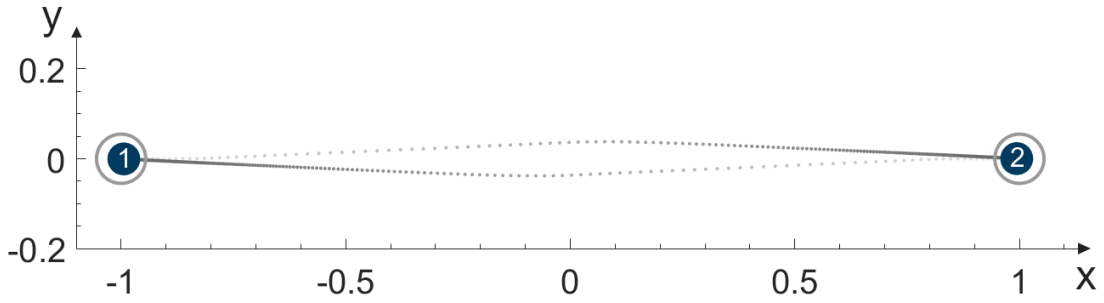
Den första simulationen är utan brus och visas i figur 17. Simulationen visar att robotarna har svårt att välja vilken som ska väja åt vilket håll, då deras banor är exakt symmetriska. De väljer istället att sänka sin fart till nära 0. Att robotarna sedan tar ett beslut och når fram till sina önskade positioner efter 39 s antas bero på bakomliggande processer i MATLAB som skapar variation.



(a) Robot 1 och 2 startar med två meters mellanrum och riktade mot varandra. Deras önskade positioner sätts så att robotarna byter plats.



(b) Vid tiden 3 s. Robotarna enas tidigt vilken som ska väja åt vilket håll och slipper sakta ner.



(c) Robotarna når sina önskade positioner redan efter 15 s, vilket är betydligt snabbare än simuleringen i figur 17 som saknar brus.

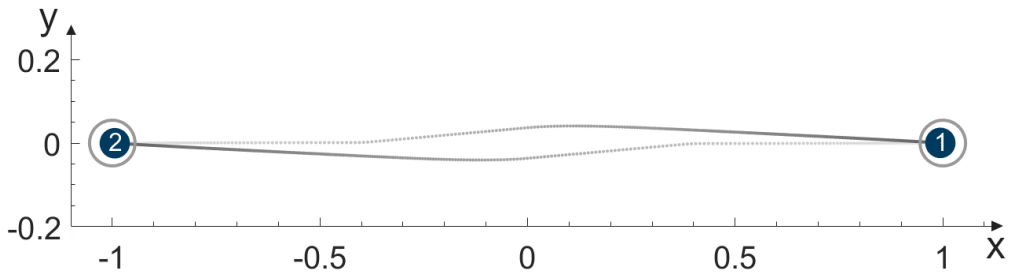
Figur 18: Simulering där två robotar byter position där ett brus mellan ±0.02 rad/s har lagts till på riktningens styrsignal. Denna simulering görs för att kunna jämföras med simuleringen utan brus, som återfinns i figur 17.

I den andra simulationen har ett slumpartat brus inom intervallet ±0.02 rad/s införts på riktningens styrsignal, vilket visas i figur 18. Bruset gör att robotarnas banor inte blir exakt symmetriska, utan det blir fördelaktigt att väja åt ena hållet. Detta leder till att robotarna tidigt ändrar sina banor och slipper sänka farten lika mycket som i första simuleringen. I den här simulering nådde robotarna sina önskade positioner efter 15 s vilket är betydligt snabbare än de 39 s det tog i den första simuleringen.

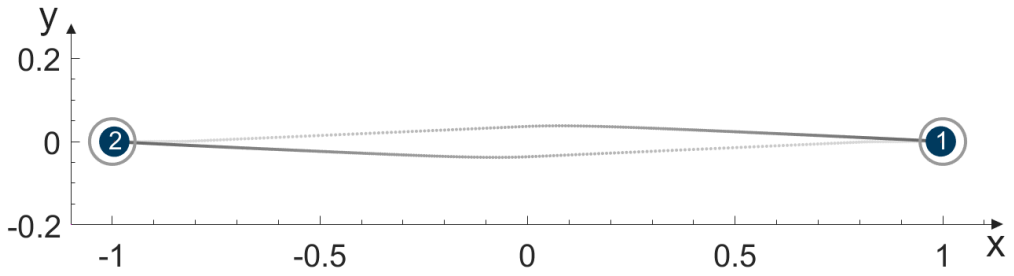
Vid implementation i en fysisk robot kommer ett litet brus att förekomma naturligt då allt ifrån ojämt underlag, brusig sensordata samt fördröjningar i elektriska och mekaniska system leder till störningar. Att en perfekt symmetrisk rörelse skulle uppstå i ett sådant system vore väldigt osannolikt.

5.3.2 Påverkan av τ på robotens rörelse

Ett viktigt designval vid kollisionshanteringen är valet av τ, vilket avgör hur långt in i framtiden kollisioner bör undvikas och därav hur försiktigt robotarna kör. I kapitel 4.8 redovisas en metod för beräkning av det minimala värdet på τ. Detta τ implementeras i simuleringen tillsammans med simuleringens valda konstanter. I figur 19 visas hur val av ett lägre τ än det beräknade τ_{min} gör att robotarna börjar anpassa sin bana till varandra när de är närmare varandra än om τ_{min} används.



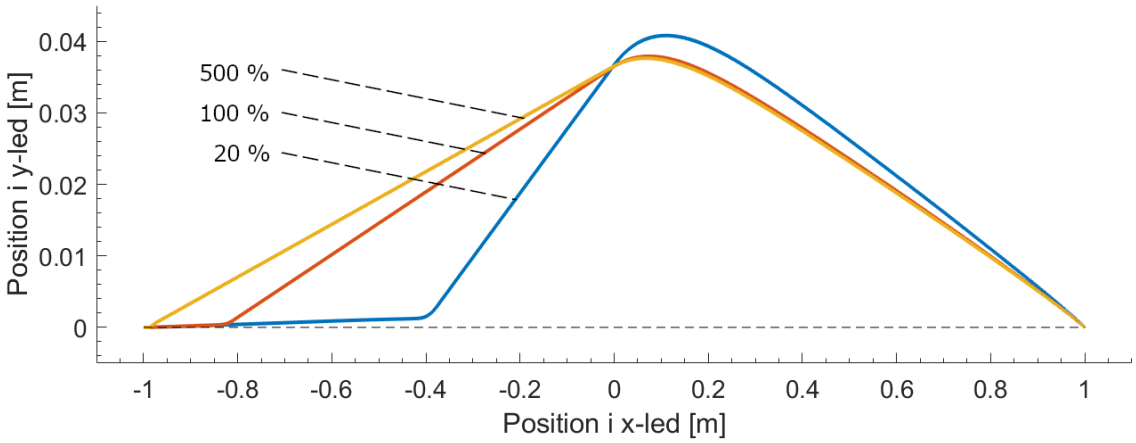
(a) Bana för robotar som byter plats med $\tau = 20\,\% \cdot \tau_{min}$.



(b) Bana för robotar som byter plats med $\tau = \tau_{min}$.

Figur 19: Här genomförs samma test som beskrivs i figur 18, men med varierande τ. Syftet är att se hur rörelsemönstret förändras beroende på vilket τ som väljs. För att förtydliga finns det även en graf som illustrerar detta i figur 20.

För att illustrera påverkan τ har på robotens rörelsebana ytterligare visas rörelsebanan för robot 1 för tre olika värden på τ i figur 20. Det syns tydligt att då τ ökar som procent av τ_{min} så börjar roboten justera riktningen tidigare. Vid högre τ utökas tidspannet vilket resulterar i tidigare undvikning. Skillnaden är dock ej lika markant som vid lägre värden.

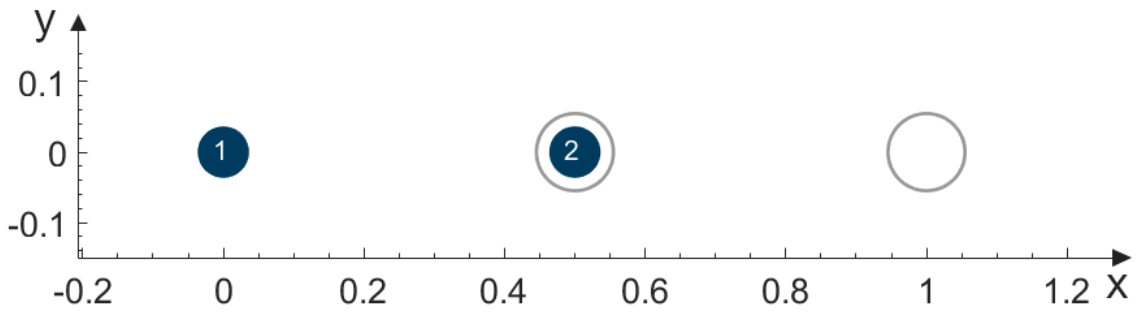


Figur 20: Banan för robot 1 i figur 19 med olika τ som procentuell andel av τ_{min}. Här förtydligas att roboten börjar väja tidigare för den mötande roboten när ett större τ väljs.

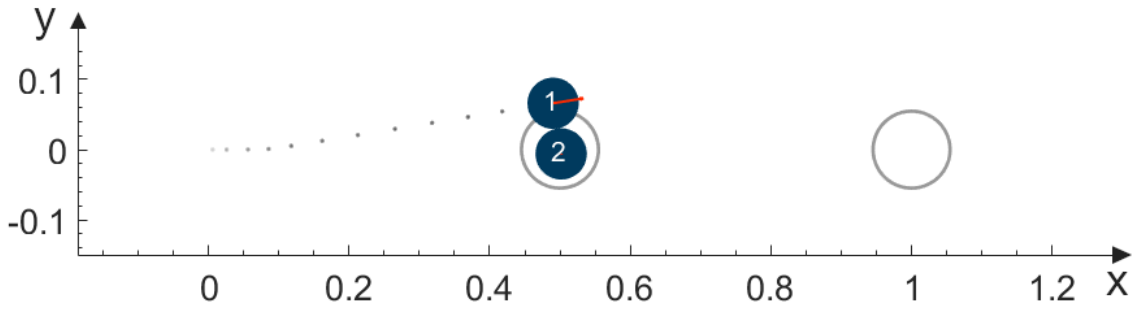
5.3.3 Kollisionsundvikning för följare och ledaren

Som nämndes i kapitel 2 har en ledare implementerats för att enkelt kunna styra formationen i det globala koordinatsystemet och även ha en punkt att bilda formationen kring. Ledarroboten tar ingen hänsyn till följarrobotarna utan följarna beaktar ledarens rörelse och förhåller sig till detta.

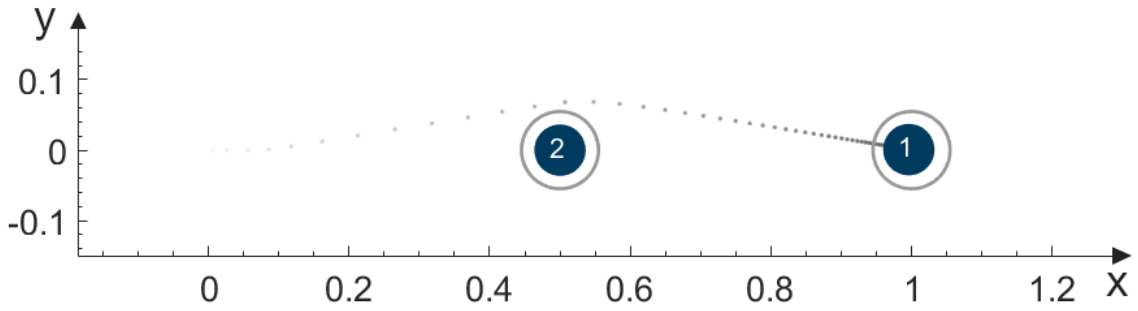
Detta illustreras vid jämförelse av figur 21 och figur 22, där följarroboten passerar en följare respektive en ledare. I figur 21b passerar en följarrobot en annan följarrobot där båda samarbetar för att så smidigt som möjligt ta sig fram till respektive önskad position. Detta sker med relativt liten marginal till följd av kollisionsundvikningen, då varje robot strävar efter så optimal körbana som möjligt. I figur 22b passerar följaren ledaren utan att ledaren förflyttar sig för att undvika följarroboten. Anledningen till att ledaren har implementerats så är för att den enbart ska fungera som ett riktmärke för formationen och därmed ej bör ta hänsyn till hur formationen bildas.



(a) Ursprungslägen för robotarna. Robot 2 står på sin önskade position. Robot 1 står still på vänster sida men har sin önskade position på högersida om robot 2.

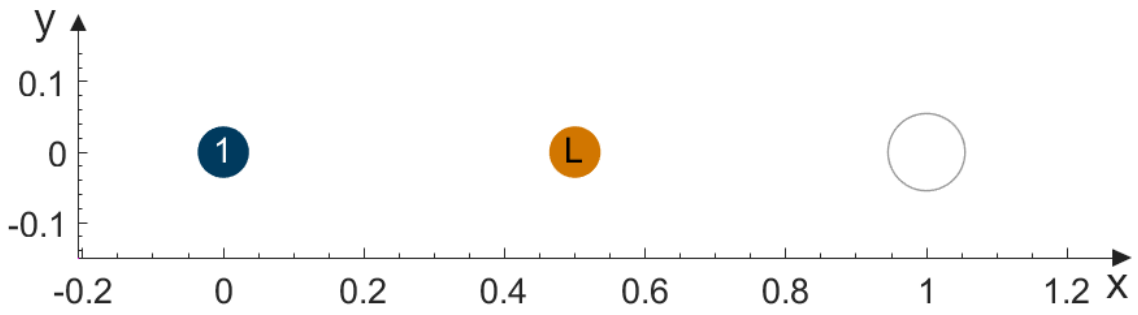


(b) När robotarna möts på mitten flyttar sig båda och hjälps åt för att undvika kollision.

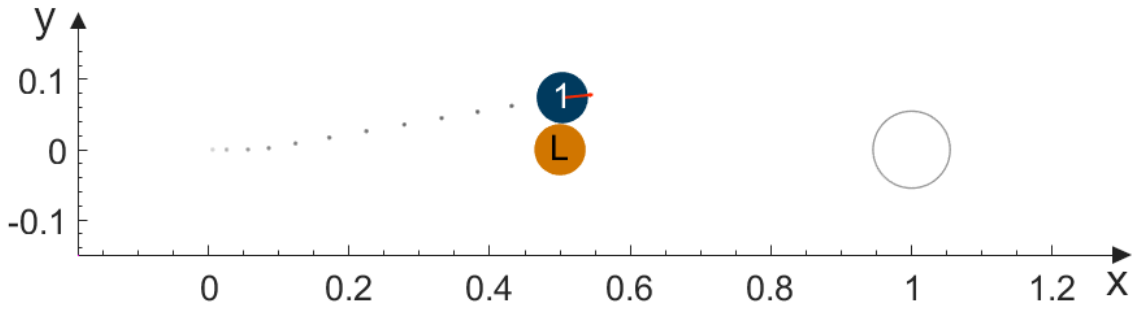


(c) Robot 2 åker tillbaka till sin önskade position i mitten och robot 1 fortsätter mot sin önskade position till höger.

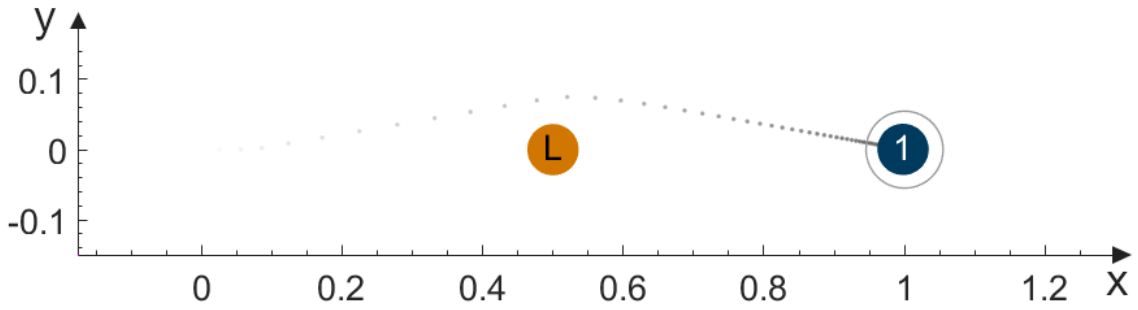
Figur 21: Enkelt exempel på kolisionsundvikning mellan två robotar.



(a) Ursprungslägen för robotarna. Robot 1 står till vänster om ledaren men har sin önskade position till höger om den.



(b) När robotarna möts på mitten flyttar sig inte ledaren utan robot 1 får köra runt den för att undvika kollision.

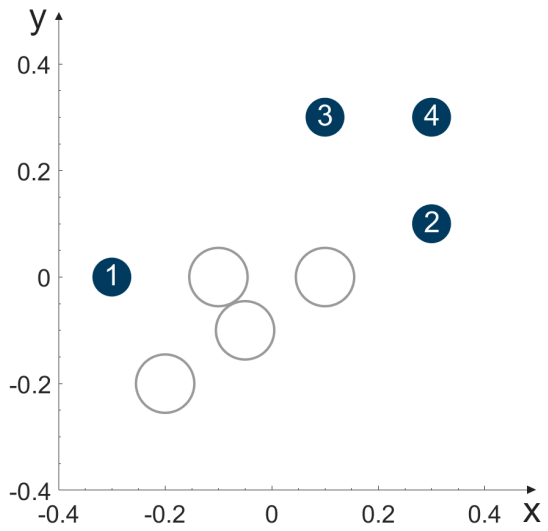


(c) Robot 1 fortsätter mot sin önskade position medan ledaren står kvar.

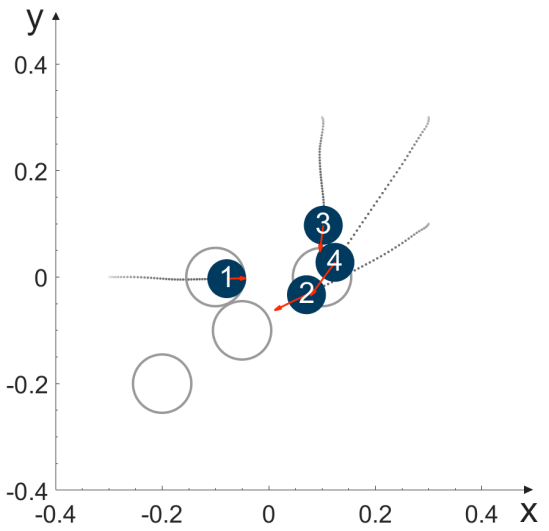
Figur 22: Enkelt exempel på kolisionsundvikning mellan en följare och en ledare.

5.3.4 Flera robotar som undviker varandra

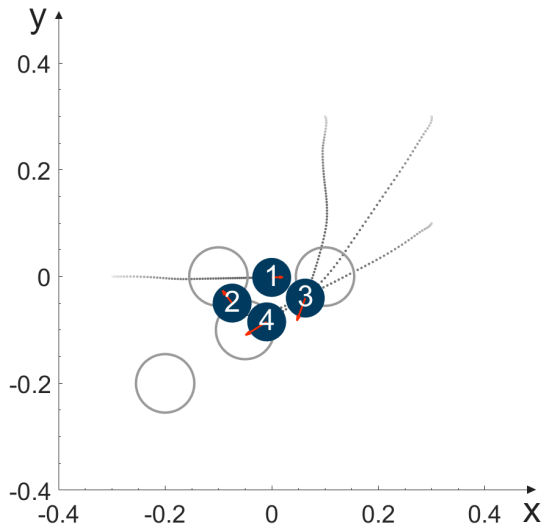
När grunden för kollisionsundvikning har etablerats kan mer komplexa situationer som involverar fler än två robotar simuleras. Som beskrivet i kapitel 4 är principen bakom kollisionsundvikningen för flera robotar densamma som för ett system med endast två, där mängden tillåtna hastigheter begränsas ytterligare för varje robot som tillkommer. Ett system med flera robotar som ska till sina önskade positioner simuleras i figur 23 där inga kollisionser sker trots att de önskade positionerna befinner sig nära varandra och att robotarnas rörelsebanor korsar varandra för att nå positionerna.



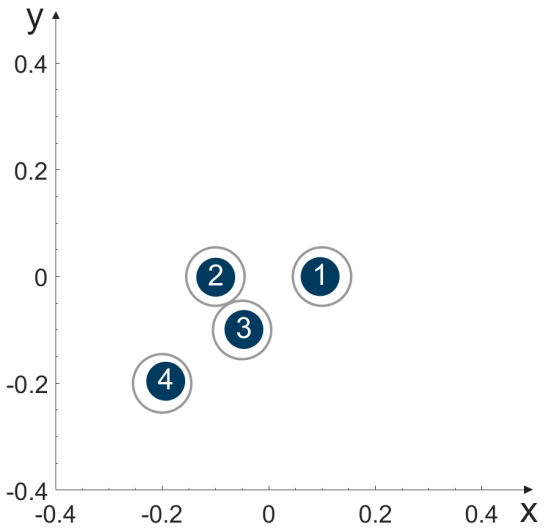
(a) Vid tiden 0 s. Fyra robotar börjar i slumpmässiga positioner och får slumpmässiga önskade positioner inom ett område.



(b) Vid tiden 2.6 s. Robotarna 2-4 har anpassat farten för att undvika att kollidera med varandra samtidigt som de tillsammans undviker robot 1 som rör sig mot den högra positionen.



(c) Vid tiden 4 s. Robot 2-4 rundar robot 1 som har sänkt farten för att undvika kollisioner



(d) Vid tiden 12 s. Robotarna är framme vid deras önskade position.

Figur 23: Flera robotar som undviker varandra vid stilla önskade positioner.

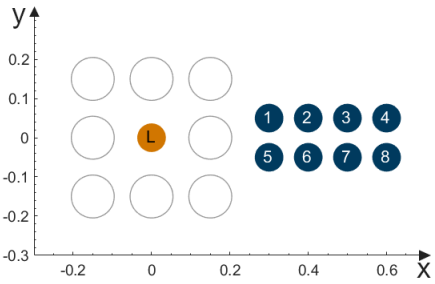
5.4 Simuleringsresultat för formationsbildning

Genom att dynamik-, styrning- och kollisionsundvikningsalgoritmerna är införda kan varje enskild robot beräkna var den bör åka för att ta sig till sin önskade position utan att kollidera med andra robotar. Nu återstår endast att bilda formationer.

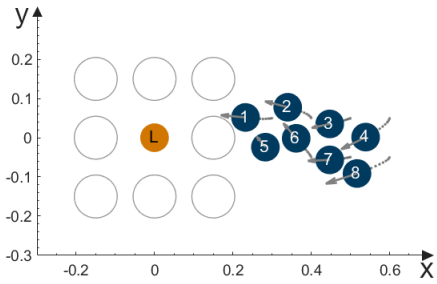
Formationerna är uppbyggda genom att varje följarrobot får en önskad position relativt ledaren som styrningen skall styra den mot. Dessa relativa positioner definieras med koordinater, därför kan formationerna anta godtyckliga former, förutsatt att

inga positioner överlappar så att robotarna inte får plats. När ledaren förflyttar sig förflyttas även de önskade positionerna, läs mer i kapitel 2.2.

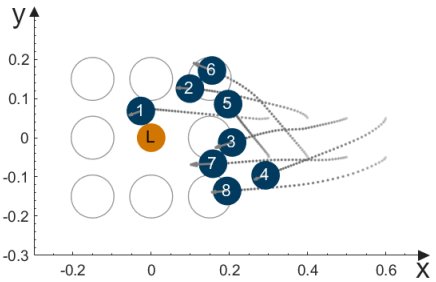
5.4.1 Stillastående formationsbildning



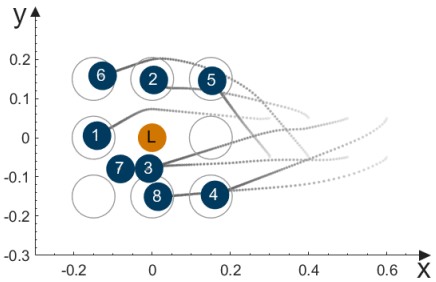
(a) Vid tiden 0 s. Åtta följarrobotar börjar vid sidan av en ledare med önskade positioner runt om den. Se figur 24f för önskade positioner för individuella robotar.



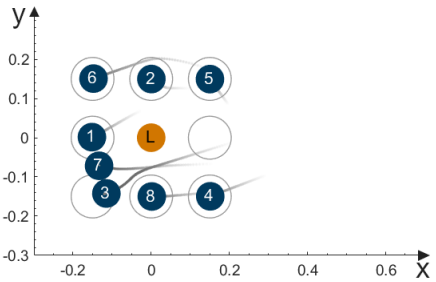
(b) Vid tiden 1.1 s. Robotarna börjar regleras mot sina positioner.



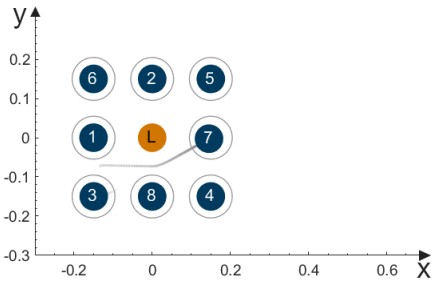
(c) Vid tiden 3.6 s. Robot 7 har en önskad position ovanför sig men robot 3 är i vägen.



(d) Vid tiden 9.8 s. Robot 7, som vill till positionen till höger, kan inte på grund av robot 3.



(e) Vid tiden 15.2 s. Robot 3 har kommit till sin önskad position och ur vägen för robot 7 som nu kan börja röra sig mot sin önskade position.



(f) Vid tiden 25.3 s. Alla robotar har nått sina önskade positioner.

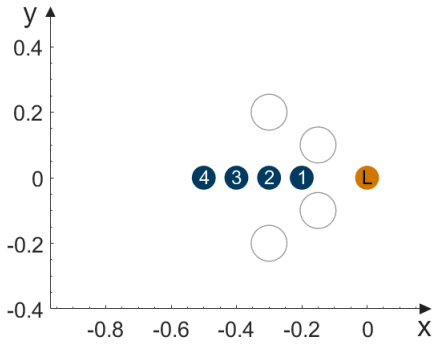
Figur 24: Simulering av formationsbildning runt en stillastående ledare.

Vid all typ av formationsbildning strävar följarrobotarna efter att komma fram till sina önskade positioner men prioriterar undvikning av kollisioner med varandra och ledaren. Som beskrivet i avsnitt 5.3.3 beaktar ledaren inte de andra robotarna utan allt ansvar för kollisionsundvikning faller på följarrobotarna. Ett exempel på formationsbildning runt en stillatående ledare visas i figur 24. Figurerna 24 c - f visar

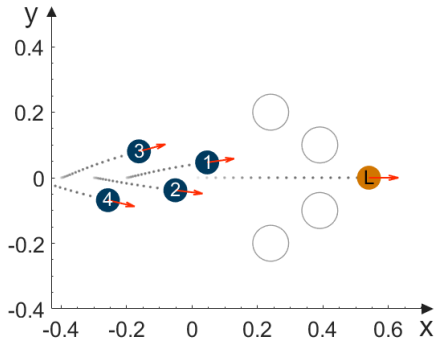
tydligt att kollisionsundvikning prioriteras före styrningen mot en önskad position. Robot 7 befinner sig tidigt nära sin önskade position, men om den skulle stanna där hade en kollision med robot 3 inträffat. För att undvika kollision behåller robot 7 sin fart vilket leder till att den kommer bort ifrån sin önskade position. Först när robot 3 svängt av mot sin önskade position och är ur vägen för robot 7 kan den säkert ta sig tillbaka till sin egna önskade position.

5.4.2 Formationsbildning under rörelse

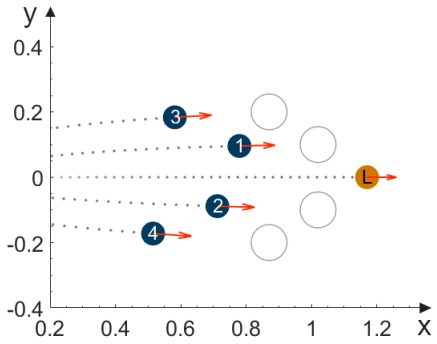
Då formationer bildas runt en ledare som är i rörelse sker styrning och kollisionsundvikning för följarrobotarna på samma sätt som när ledaren är stilla. Skillnaden blir att hela systemet rör sig och på så sätt måste följarrobotarna hinna ikapp ledaren och sedan kontinuerligt anpassa sig efter hur den önskade positionen förflyttas för att behålla formationen.



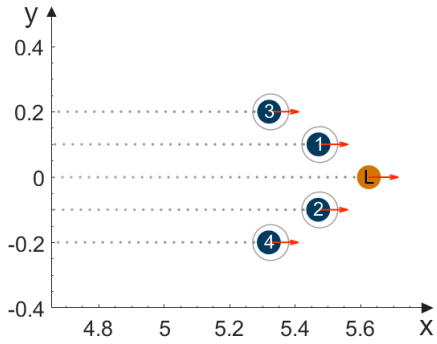
(a) Vid tiden 0 s. Robotarna befinner sig i ursprungsläget.



(b) Vid tiden 1.2 s. Robotarna börjar åka mot de önskade positionera.



(c) Vid tiden 2.6 s. Robotarna närmar sig sina önskade positioner.



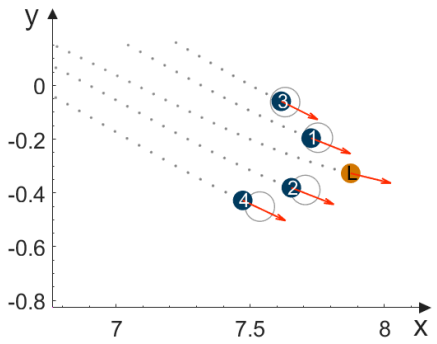
(d) Vid tiden 12.5 s. Robotarna är framme vid de önskade positionerna och formationen har bildats.

Figur 25: Simulering av formationsbildning under rörelse i x-led där ledaren kör i 30 % av maxfart.

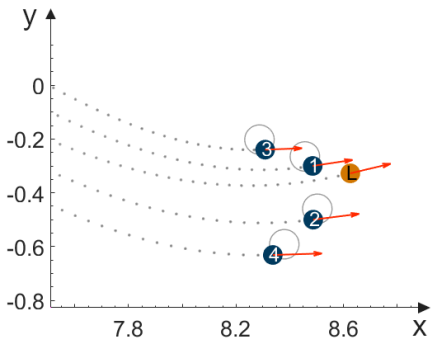
I figur 25 visas hur fyra följarrobotar accelererar för att hinna ikapp en ledaren som kör i 30 % av robotens maxfart. Följarrobotarna hittar till sina önskade positioner och anpassar sin fart för att behålla sina positioner i förhållande till ledaren.

Om ledaren svänger ökas svårigheten för följarrobotarna att hålla formationen eftersom de önskade positionerna förändras av både ledarens fart och rotation istället

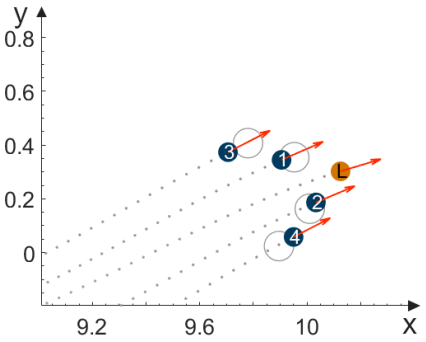
för endast farten. Ett exempel på detta är när ledaren roterar medurs roterar även de önskade positionerna medurs runt ledaren. De önskade positionerna som befinner sig längre ifrån ledaren förflyttas med en högre fart än de som är nära. Eftersom begränsningen på robotarnas rotationshastighet inte tar hänsyn till de önskade positionernas avstånd till ledaren kan det hända att en önskad position förflyttas med en fart högre än robotens maxfart vilket medför att den tillhörande följarroboten inte hinner ikapp sin önskade position.



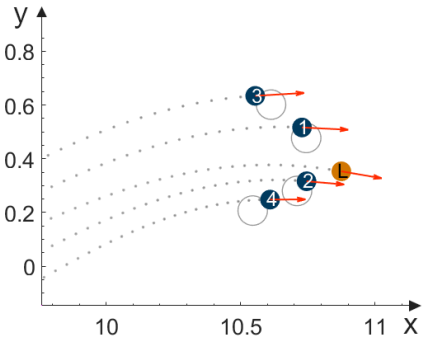
(a) Vid tiden 10.5 s. Robotarna håller en formation runt en ledare i rörelse.



(b) Vid tiden 11.5 s. Följarna längst ifrån ledaren påverkas mest av ledarens rotation då de hamnar längst ifrån sina önskade positioner.



(c) Vid tiden 13.5 s. Robotarna i innerkurvan får önskade positioner som retarderas snabbare än vad robotarna klarar av.



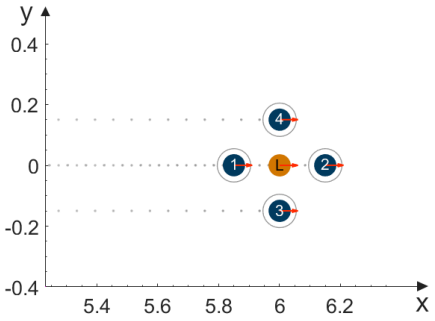
(d) Vid tiden 14.5 s. Robotarna som befinner sig i innerkurva behöver sakta ner och de i ytterkurva behöver skynda på för att hamna på rätt position igen.

Figur 26: Simulering av formationsbildning under rörelse längs en sinus-kurva där ledaren kör i 50 % av maxfarten.

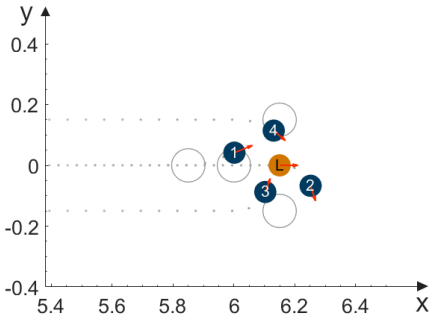
I figur 26 visas samma formation som i figur 25, men här rör ledaren sig i en sinuskurva med hastighetsfunktionen $0.5v_{max}[1, \frac{1}{2}\cos(t)]$. Då ledaren svänger till vänster, vilket visas i figur 26b, minskas hastigheten för de önskade positionerna som befinner sig på vänster sidan om ledaren i dess färdriktning och följarrobotarna vid dessa positioner måste sakta ner för att hålla positionen. Samtidigt ökar hastigheten för de önskade positionerna på höger sidan vilket gör att robotarna på denna sida behöver öka sin fart för att följa med.

5.4.3 Formationsbyte under rörelse

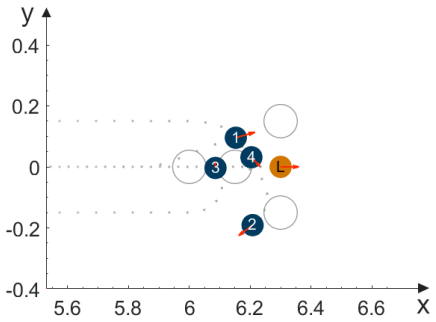
Systemet kan även hantera byte av formation vid rörelse. Den tidigare formationen likställs då med startpositioner som till skillnad från tidigare fall har en hastighet då den nya formationen efterfrågas. Detta är en fördel vid exempelvis trånga situationer där formationer behöver bytas under färd för att underlätta framfart. Detta illustreras i 24 genom att börja i en viss formation, se figur 27a, för att sedan bilda en annan formation.



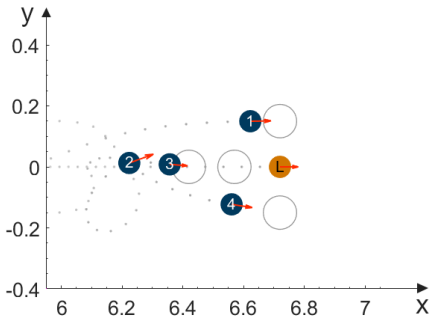
(a) Vid tiden 0 s. Robotarna befinner sig i ursprunglig position.



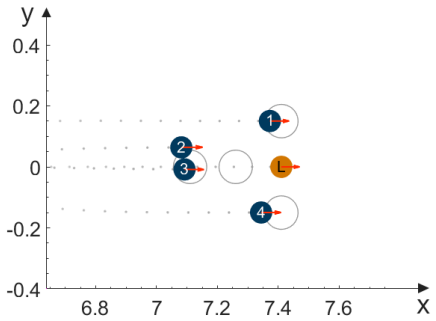
(b) Vid tiden 0.5 s. En ny formation har önskats och robotarna försöker ta sig till sina nya positioner.



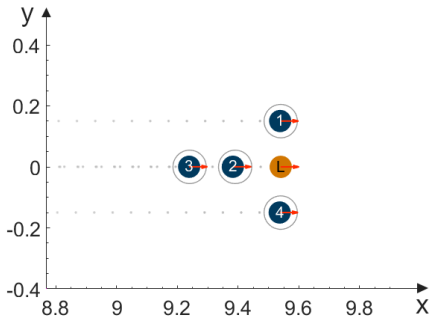
(c) Vid tiden 1 s.



(d) Vid tiden 2.4 s. Robotarna försöker nå sina önskade positioner.



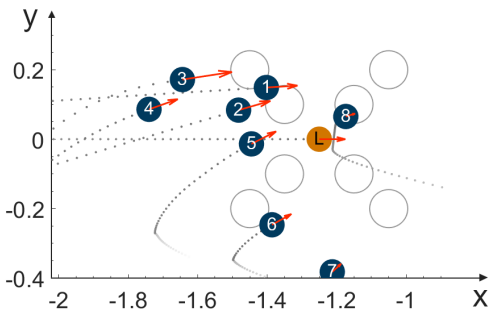
(e) Vid tiden 4.7 s.Robotarna försöker nå sina önskade positioner.



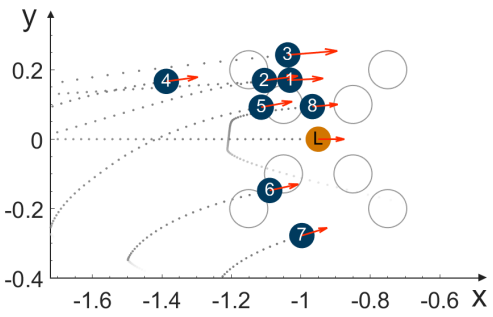
(f) Vid tiden 11.8 s. Robotarna är framme vid sina önskade positioner och har bildat den nya formationen.

Figur 27: Formationsbyte under rörelse, då ledare åker med 20 % av maxfarten.

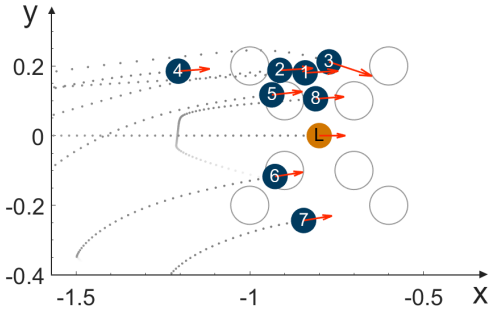
5.5 Kompakta situationer



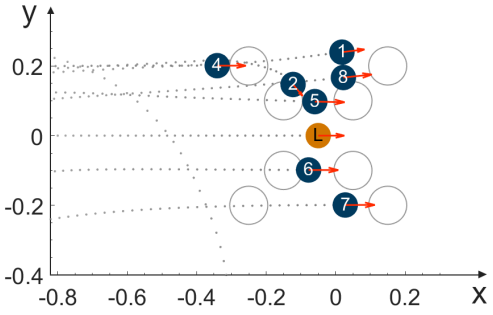
(a) Vid tiden 10 s. Robotarna rör sig mot sina positioner men tar omvägar för att undvika kollision. Robot 1 är nära sin önskade position, men fortsätter för att ge plats åt robot 2, som i sin tur undviker kollision med robot 5.



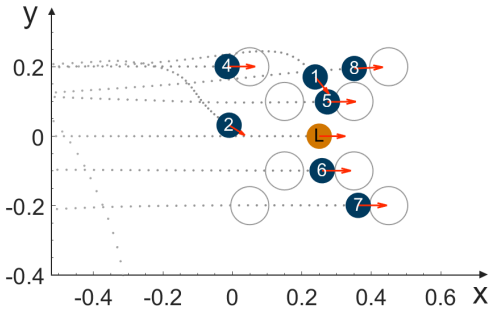
(b) Vid tiden 10.8 s. Robot 1 och 2 vill sakta ner för att komma till sina önskade positioner, men blockeras av robot 5 som vill förbi ledaren.



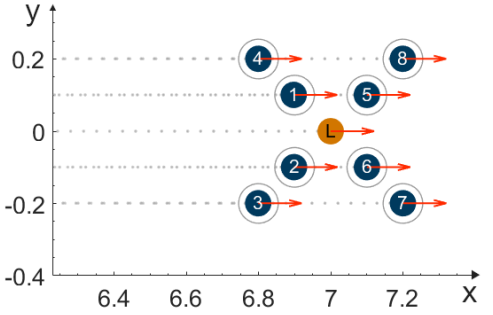
(c) Vid tiden 11.2 s. Robot 1 är helt omgiven av andra robotar och kan endast försöka följa med strömmen för att minimera kollisionsrisken.



(d) Vid tiden 13.2 s. Robot 3 har rundat klungan och försvunnit ur sikte. Detta har lämnat fri väg framåt för robot 1 och 8. Robot 5 fortsätter framåt och låter robot 2 falla tillbaka utan att kollidera.



(e) Vid tiden 14 s. Den kompakta situationen är löst och alla robotar kan säkert ta sig mot sina önskade positioner. Robot 3 befinner sig utanför bild men har nu fri väg till sin önskade position.



(f) Vid tiden 25 s. Alla följarrobotar har nått sina önskade positioner i formationen och håller dessa medan ledaren rör sig.

Figur 28: Kollisionsundvikning vid kompakta situationer under formationsbildning kring ledarrobot som rör sig i 40 % av robotens maxfart.

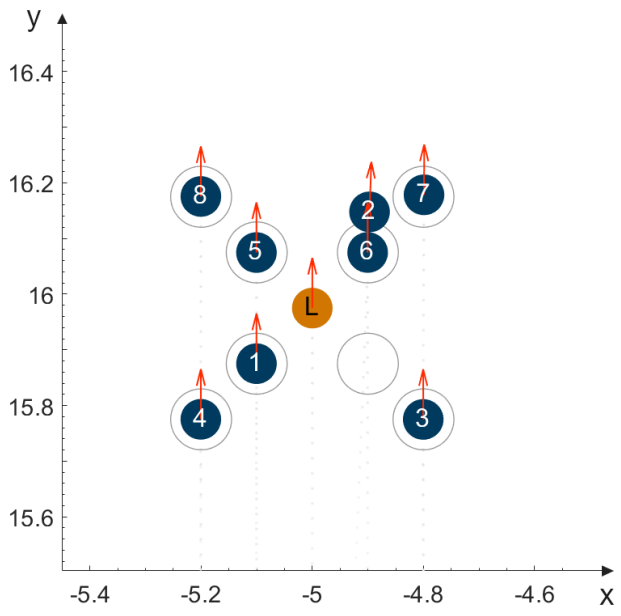
Det finns specialfall då robotarna ställs inför kompakta situationer. Exempel på det är när slutgiltiga positioner är nära varandra eller många robotar är på en liten yta.

Detta ställer ytterligare krav på systemet och här simuleras teorin från kapitel 4.7. Ett exempel på detta ses i figur 28, där åtta följarrobotar ska formas relativt nära varandra i formation kring ledaren. I sådana situationer kommer robotarna beräkna säkrast möjliga hastighet att färdas i och resultatet blir att de följer med strömmen. I detta fall garanteras inte kollisionsfrihet utan hastigheten kan leda till kollision inom tiden τ om alla robotar fortsätter i samma riktning. Eftersom Δt är mindre än τ kommer dock många situationer lösas med tiden då hastigheten hela tiden justeras tills någon robot kan svänga undan och situationen därmed är löst.

Något som skulle kunna förhindra eller i alla fall mildra dessa typer av situationer skulle varit om varje robot inte tilldelades en önskad position i formationen godtyckligt utan att dessa istället tilldelades enligt någon typ av prioritering. Prioriteringen kan exempelvis bestå av att minimera antalet korsande banor eller att minimera det längsta avståndet mellan en startposition och motsvarande önskade position.

5.6 Robot i låst läge

I de flesta fall fungerar det väl att prioritera kollisionsundvikningen över formationsbildningen. Om önskade positioner exempelvis överlappar så kolliderar inte robotarna i strävan efter att bilda formationen. Däremot är en negativ bieffekt av detta att situationer uppstår då formationer inte bildas trots att önskade positioner inte överlappar. Ett sådant fall klassas som ett låst läge, varav en visas i figur 29. Robot 2 hamnat så att det ej är möjligt att nå önskad position. Hastigheten är för hög för att kunna parera förändringar och kollisionsundvikning prioriteras istället. Då formationen är relativt tät finns ingen möjlighet att åka mellan de andra robotarna till följd av kollisionsundvikningen.



Figur 29: Robot 2 fastnar i fel position men kan inte ta sig till sin önskade position. Ledaren rör sig i 30 % av robotens maxfart.

Systemet har inget kollektivt ansvar för formationsbildning och prioriterar därför inte att alla robotar ska nå sina positioner. Detta skulle kunna lösas genom att exempelvis sakta ner formationen så att roboten kan passera framför eller dela formationen så att roboten kan passera. En vidareutveckling av systemet där robotarna samverkar ytterligare hade gjort det möjligt att undvika denna typen av situationer med låsta lägen.

6 Framtida utvecklingsområden

Vid eventuell vidareutveckling av projektet finns flera delområden som kan undersökas. Exempelvis fysisk implementation i en sfärisk robot, utbyggnad av kommunikation och mer delat ansvar, samt möjligheter för systemet klara andra typer av robotar och situationer. I detta avsnitt diskuteras hur projektets olika delar kan vidareutvecklas och vilka framtida utvecklingsområden som existerar.

6.1 Fysisk implementation

En möjlig vidareutveckling av projektet hade varit en fysisk implementation av systemet i sfäriska robotar. I dagsläget är det till följd av tekniska begränsningar ej möjligt att utföra detta med Sphero BOLT men när ett fullständigt JavaScript API finns tillgängligt eller om MATLAB får stöd för Bluetooth Low Energy (BLE) skulle det vara möjligt. Med hjälp av sensordata i form av hastighet och kompassriktning eller sensordata för position kan hastigheten eller positionen regleras för att se till så formationer utformas och bibehålls vid förflyttning.

En vidareutveckling av simuleringen som kan göras innan implementation i ett fysiskt system är införandet av en säkerhetsradie, vilket innebär att robotarna håller större marginaler till varandra för att undvika kollision. Algoritmen som utvecklats i rapporten leder till att robotarna väljer att väja så lite som möjligt för att optimera robotens rörelsebana. Vid en fysisk implementation finns yttre faktorer som kan påverka robotens rörelsebana och då kan ökad säkerhet vara nödvändig för att undvika kollision.

6.2 Styrning och robotens dynamik

Den matematiska modellen som presenterades i ekvation 1 är generell för alla *unicycle models*, däribland sfäriska robotar. Resultatet av en generell modell ger möjlighet till implementation i många olika typer av robotar och skapar stora friheter vid eventuell vidareutveckling av projektet.

Ytterligare något som är värt att undersöka vidare är delegerandet av robotarnas önskade positioner i formationen då det kan lösa kompakta situationer som nämnts i 5.5. I dagens system är de förutbestämda vid start och väljs godtyckligt, men detta kan vidareutvecklas för att optimera systemet.

Till framtida tester så kan en jämförelse mellan den framtagna styrningen med en robots redan existerande styrning erhålla ett mått på det framtagna systemets egenskaper. Målet med dessa tester skulle delvis vara att undersöka det framtagna systemets framgång men likaså att få någon typ av referens för vad som är rimligt att förvänta sig i form av prestanda.

6.3 Kommunikationssystem

Något som måste tas hänsyn till vid fysisk implementering är begränsning av kommunikationssystemets räckvidd. Om sensordatan eller kommunikationssystemet har liten räckvidd skulle risken för kollision med både andra agenter och hinder öka drastiskt eftersom robotarna inte skulle kunna ta hänsyn till robotar utanför räckvidden. Å andra sidan kan ett välfungerande kommunikationssystem bidra med nya funktioner i systemet.

Om något problem uppstår, till exempel om en följarrobot har kört in i en vägg eller inte kan minska avståndet mellan sin nuvarande och önskade position trots att den kör i maxhastighet, kan det vara hjälpsamt med ett kommunikationssystem för att skicka felmeddelanden mellan följarrobotar och ledarroboten. Då skulle följarroboten kunna be ledaren om en ny tillfällig önskad position som inte orsakar samma problem, alternativt be ledaren att vänta om det är hastigheten som skapar problem. Det här skulle även kunna lösa situationer med låst läge då följarroboten kan meddela att den inte når fram till sin önskade position och ledaren då kan beordra övriga robotar om att tillfälligt anta andra mer utspridda positioner så att den vilsna roboten kan komma till sin önskade position. När problemet är löst kan då den ursprungliga formationen återupptas.

6.4 Anpassning till andra förutsättningar

Ännu en vidareutveckling av systemet hade varit att utöver kollsionsundvikning av andra kända robotar även kunna undvika okända objekt i planet. Det hade ökat värdet av projektets resultat och med det även dess möjliga implementeringsområden.

I denna rapport behandlas system för en homogen grupp sfäriska robotar. Detta system är dock inte begränsat till detta, utan kan enkelt justeras till att behandla även grupper med olika individer som har olika egenskaper. Varje robot hade då haft egna styrsystem men alla följt samma formationsbildningsprincip och kollisionundvikningsalgoritm. Däremot existerar möjliga komplikationer för detta i kollisionundvikningsalgoritmen då en robot antar att de andra robotarna har samma maxhastighet som sig själv. För en heterogen grupp sfäriska robotar, där de olika robotarna har olika maxhastigheter, finns högre risk för kollision om maxhastigheterna skiljer sig mycket mellan individerna.

7 Slutsats

Syftet med detta projekt var att utveckla och implementera en algoritm för att få sfäriska robotar att följa en ledarrobot medan olika formationer utformas och bibehålls. Styrningen har implementerats för att få sfäriska robotar att röra sig från nuvarande position till önskad position. Detta utfördes genom att utgå ifrån matematiska samband för *unicycle robots* och ta fram ett reglersystem. Den ömsesidiga kollisionsundvikningsalgoritmen gör att formationsbildandet sker med minimal risk för kollisioner. Algoritmen fungerar även vid kompakta och trånga situationer. En ledarrobot får sedan formationen att färdas i det globala koordinatsystemet och fungerar även som ett riktmärke för formationen att bildas kring. Kombinationen av styrning och kollisionsundvikningsalgoritm simulerades och visar att de två delarna kan kollaborera på ett önskvärt, effektivt och smidigt sätt.

Referenser

[1] U. S. och Amblin Entertainment, ”Jurassic world”, [Online]. Tillgänglig: http://www.jurassicworldfilmen.se/ hämtad: 2019-05-15.

[2] A. Halme, T. Schonberg och Y. Wang, ”Motion control of a spherical mobile robot”, i *Proceedings of 4th IEEE International Workshop on Advanced Motion Control - AMC '96 - MIE*, vol. 1, mars 1996, 259–264 vol.1.

[3] A. K. Bhoopalam, N. Agatz och R. Zuidwijk, ”Planning of truck platoons: A literature review and directions for future research”, *Transportation Research Part B: Methodological*, vol. 107, s. 212–228, 2018.

[4] M. Pérez-Ruiz, P. Gonzalez-de-Santos, A. Ribeiro, C. Fernandez-Quintanilla, A. Peruzzi, M. Vieri, S. Tomic och J. Agüera, ”Highlights and preliminary results for autonomous crop protection”, *Computers and Electronics in Agriculture*, vol. 110, s. 150–161, 2015. [Online]. Tillgänglig: http://www.sciencedirect.com/science/article/pii/S0168169914002920 hämtad: 2019-05-15.

[5] M. G. Hinchey, J. L. R. and Walter F. Truszkowski, C. A. Rouff och R. Sterritt, ”Autonomous and autonomic swarms”, *TitleMIT Technology Review*, jan. 2005. [Online]. Tillgänglig: https://ntrs.nasa.gov/search.jsp?R=20050210015 hämtad: 2019-02-06.

[6] T. Abu-Lebdeh, ”Implementation of autonomous navigation algorithms on two wheeled ground mobile robot”, *American Journal of Engineering and Applied Sciences*, vol. 7, s. 149–164, april 2014.

[7] B. Lennartsson, *Reglerteknikens grunder*, 4. utg. Studentlitteratur AB, 2002.

[8] J. van den Berg, S. J. Guy, M. Lin och D. Manocha, ”Reciprocal n-body collision avoidance”, i *Robotics Research*, C. Pradalier, R. Siegwart och G. Hirzinger, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, s. 3–19.

[9] A. Kushleyev, D. Mellinger, C. Powers och V. Kumar, ”Towards a swarm of agile micro quadrotors”, *Autonomous Robots*, vol. 35, nr 4, s. 287–300, nov. 2013. [Online]. Tillgänglig: https://doi.org/10.1007/s10514-013-9349-9 hämtad: 2019-05-15.

[10] R. Narain, A. Golas, S. Curtis och M. C. Lin, ”Aggregate dynamics for dense crowd simulation”, *ACM Trans. Graph.*, vol. 28, nr 5, 122:1–122:8, dec. 2009. [Online]. Tillgänglig: http://doi.acm.org/10.1145/1618452.1618468 hämtad: 2019-05-15.

[11] M. de Berg, M. van Kreveld, M. Overmars och O. Schwarzkopf, ”Linear programming”, i *Computational Geometry: Algorithms and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, s. 63–92. [Online]. Tillgänglig: https://doi.org/10.1007/978-3-662-03427-9_4 hämtad: 2019-05-15.

[12] Sphero, ”Sphero bolt”, 2019. [Online]. Tillgänglig: https://www.sphero.com/sphero-bolt hämtad: 2019-04-01.

[13] ——, ”Bot comparison - sphero education”, u.å. [Online]. Tillgänglig: https://www.sphero.com/education/bot-comparison/ hämtad: 2019-04-05.