TSEA56 - Kandidatprojekt i elektronik
LIPS Förstudie: Reglering

Version 1.2

Grupp 4

Hynén Ulfsjöö, Olle `ollul666`

Strömberg, Isak `isast763`

2 juni 2016

Status

Granskad	IS	29 april 2016
Godkänd	-	-

PROJEKTIDENTITET

2016/VT, Undsättningsrobot Gr. 4

Linköpings tekniska högskola, ISY

Namn	**Ansvar**	**Telefon**	**E-post**
Isak Strömberg (IS)	Projektledare	073-980 38 50	isast763@student.liu.se
Olle Hynén Ulfsjöö (OHU)	Dokumentansvarig	070-072 91 84	ollul666@student.liu.se
Emil Wasteson (EW)	Hårdvaruansvarig	076-836 61 66	emiwa068@student.liu.se
Elena Tronje (ET)	Mjukvaruansvarig	072-276 92 93	eletr654@student.liu.se
Zimon Inge (ZI)	Testansvarig	070-171 35 18	zimin415@student.liu.se
Lovisa Gustafsson (LG)	Leveransansvarig	070-210 32 53	lovgu777@student.liu.se

E-postlista för hela gruppen: isast763@student.liu.se

Kund: ISY, Linköpings universitet
tel: 013-28 10 00, fax: 013-13 92 82
Kontaktperson hos kund: Mattias Krysander
tel: 013-28 21 98, e-post: matkr@isy.liu.se

Kursansvarig: Tomas Svensson
tel: 013-28 13 68, e-post: tomass@isy.liu.se
Handledare: Peter Johansson
tel: 013-28 13 45, e-post: peter.a.johansson@liu.se

Innehåll

e-post: ollul666@student.liu.se
isast763@student.liu.se

Dokumenthistorik

Version	**Datum**	**Utförda förändringar**	**Utförda av**	**Granskad**
1.2	2016-04-29	Fjärde utkastet	Grupp 4	Isak Strömberg
1.1	2016-04-24	Tredje utkastet	Grupp 4	Olle Hynén Ulfsjöö
1.0	2016-04-06	Andra utkastet	Grupp 4	Isak Strömberg
0.1	2016-03-03	Första utkastet	Grupp 4	Isak Strömberg

e-post: ollul666@student.liu.se
isast763@student.liu.se

1 Inledning

Denna rapport kommer att avhandla två olika, övergripande frågeställningar. Den har därför strukturerats så att varje huvudrubrik (*Problemformulering, Kunskapsbas, etc.*) är uppdelad i antingen *Reglering* eller *Kartläggning och vägoptimering* under vilken textmassan kopplad till respektive område och rubriken placerats.

2 Problemformulering

Nedan presenteras de problemformuleringar som förstudien tar avstamp i.

2.1 Reglering

Att skapa system med hög grad av automation, som klarar av att operera i ostrukturerade miljöer är enligt [**?**] en av grundförutsättningarna för att öka robotars användningsområde. För att nå denna grad av självständighet krävs det att roboten, med hjälp av väldesignade modeller, kan förstå sin omgivning. Det krävs också att roboten, genom sensorer, kan utvinna meningsfull information ur sin omgivning för att få modellerna att fungera som avsett. I denna rapport läggs fokus på reglermodellerna och hur dessa, förutsatt att de förses med korrekt sensordata, kan utformas för att styra roboten på ett önskat sätt.

2.1.1 Frågeställningar

De frågeställningar som behandlas i rapportens huvuddel återfinns nedan.

- Hur kan man reglera en robot så att den kör rakt i en korridor?
- Hur kan styrning ske i kurvor och under rotationer?

2.1.2 Avgränsning

Rapporten kommer enbart behandla de reglertekniska problem och frågeställningar som kan kopplas till autonom navigation av en labyrint enligt banspecifikationen, i appendix B, utan öppna rum. Reglerad körning på öppna ytor kommer alltså inte behandlas.

2.2 Kartläggning och vägoptimering

Simultan lokalisering och kartläggning anses av [**?**] vara en av de större utmaningarna med att bygga en autonom robot. Att kunna navigera och kartlägga en okänd omgivning ställer höga krav på både mjukvara och hårdvara. Förutsatt att sensorerna kan mäta med godkänd upplösning behöver roboten intelligent kunna tolka mätvärdena och översätta dessa till en karta och position.

e-post: ollul666@student.liu.se
isast763@student.liu.se

2.2.1 Frågeställningar

De frågeställningar som behandlas i rapportens huvuddel återfinns nedan.

- Hur kan en robot som enbart har tillgång till sensordata kartlägga en okänd labyrint?
- Vilka kartsökningsalgoritmer ger en effektiv avsökningssekvens av en okänd labyrints korridorer?
- Givet en karta över en labyrint, hur kan då den kortaste vägen mellan två punkter beräknas?

2.2.2 Avgränsning

De labyrinter som beaktas i förstudien är utformade enligt en banspecifikationen, appendix C. Labyrinten är begränsad, enkelt sammanhängande och har alla korridorer i samma plan. Dessutom finns inga öppna rum vilket avgränsar samtliga tre underrubriker till att enbart kartlägga, följa och optimera en väg genom korridorer.

e-post: ollul666@student.liu.se
isast763@student.liu.se

3 Kunskapsbas – reglering

Nedan presenteras en kunskapsbas för regleringen.

3.1 Icke-holonom robot

En robot anses vara holonom om alla dess frihetsgrader går att styra. En landgående robot (i xy-planet) har tre frihetsgrader: förflyttning i x- och y-led samt rotation kring z-axeln. Det krävs alltså att roboten direkt kan förflytta sig i alla dessa riktningar för att den ska vara holonom. Roboten som kommer att användas i detta project är icke-holonom, eftersom den inte kan förflytta sig ortogonalt mot hjulens rotationsriktning utan att först rotera. För att den här typen av robot ska vara holonom krävs det att den är utrustad med så kallade omnihjul.

3.2 Väggföljning

Betrakta en icke-holonom, differentiellt driven robot vars koordinater (x, y, θ) är relaterade enligt

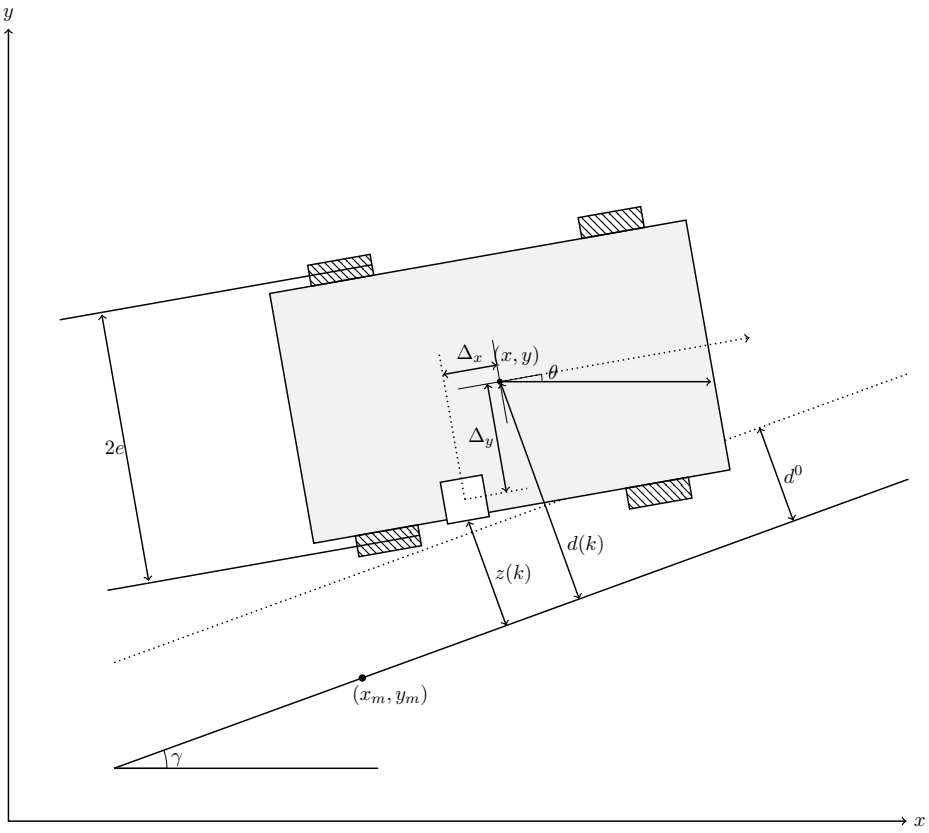
$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{1}$$

där x och y representerar positionen i ett kartesiskt koordinatsystem, θ vinkeln mellan roboten och x-axeln, v hastigheten och ω vinkelhastigheten. Dessa beror i sin tur på vinkelhastigheterna ω_1, ω_2 enligt

$$\begin{aligned} r\omega_1 &= v + e\omega \\ r\omega_2 &= v - e\omega \end{aligned} \tag{2}$$

där r är radien på robotens hjul och e är halva robotens bredd.

e-post: ollul666@student.liu.se
isast763@student.liu.se



Figur 1: Väggföljningsproblemet

Väggföljningsproblemet: *För en perfekt rak och oändligt lång vägg beskriven i det Kartesiska planet av ekvationen*

$$(x - x_m)\sin\gamma - (y - y_m)\cos\gamma = 0 \tag{3}$$

bestäm en regulator sådan att en mobil robot (1) *rör sig med en konstant hastighet* v_{des} *längs väggen och på konstant avstånd* d^0 *från den.*

I ekvation (3) representerar (x_m, y_m) en punkt längs väggen och γ väggens orientering i samma kartesiska plan som robotens position är angiven i.

3.2.1 Ideala väggföljarproblemet (IWFP)

Bemporad et al. föreslår i [**?**] en lösning på det ideala väggföljarproblemet. Antag att koordinaterna (x, y, θ) är mätbara utan fel och robotens dynamik är försumbar, det vill säga att insignaler kan ges i form av hastighet och vinkelhastighet. En lösning till väggföljningsproblemet ges i detta idealfall av återkopplingen

$$\begin{cases} v = v_{des} \\ \omega = -\frac{k(d-d^0)}{v_{des}\cos(\theta-\gamma)} - \beta\tan(\theta-\gamma) \end{cases} \tag{4}$$

där k och β är positiva skalärer och d är robotens avstånd från väggen, det vill säga $d = (y - y_m)\cos\gamma - (x - x_m)\sin\gamma$. Författarna inför sedan tillstånden $x_1 \triangleq d - d^0$ och $x_2 \triangleq \dot{d}$. Det återkopplade ekvationssystemet (1)-(4), tillsammans med antagandena ovan, kan då skrivas som

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -kx_1 - \beta x_2. \end{cases} \tag{5}$$

Detta ger att $d(t) \to d^0$, $\dot{d}(t) \to 0$ och $\sin(\theta(t) - \gamma) \to 0$ då $t \to \infty$ för alla positiva k och β.

3.2.2 Verkliga väggföljarproblemet (RWFP)

Betrakta vårt fall, figur 1, med en robotplattform där spänningen till DC-motorer på vardera sida av roboten är insignalen, en främre avståndssensor ger hastigheten, ett gyro ger vinkelhastigheten och IR-sensorer på sidorna ger avståndet till väggen. I det fallet finns ett antal, ytterligare begränsningar på systemet. Bland annat så begränsar motorernas egenskaper den maximala hastigheten och den maximala accelerationen. Vidare kommer data från sensorerna oundvikligen innehålla en viss andel brus. Slutligen är IR-sensorerna enbart verksamma då väggens normal ligger inom deras strålkon.

För att nå fram till en tillfredställande formulering av RWFP måste alltså även de ovanstående kraven tas med i beaktning, och översättas till lämpliga relationer mellan systemvariablerna. IR-sensorernas begränsning leder till följande begränsning

$$|\theta - \gamma| \leq \sigma_{max} \tag{6}$$

där vanligen $\sigma_{max} \in [10°, 15°]$. Om Ω_{max} är hjulens maximala tillåtna vinkelhastighet, det vill säga $|\omega_1|, |\omega_2| \leq \Omega_{max}$, så fås tillsammans med (2) följande begränsning

$$|v \pm e\omega| \leq r\Omega_{max}. \tag{7}$$

På grund av mätfel kan koordinaterna (x, y, θ) endast fås som skattningarna $(\hat{x}, \hat{y}, \hat{\theta})$. Dessa skattningar fås genom användning av ett utökat *Kalmanfilter* (EKF), som beskrivs mer ingående i avsnitt 5.3. Antag att mätvärden samplas varje T_c sekunder och låt $\omega(k)$ och $v(k)$ vara vinkelhastigheten och hastigheten vid tiden kT_c och $z(k)$ avståndet till väggen vid samma tidpunkt. Då fås en skattning av $(\hat{x}, \hat{y}, \hat{\theta})$ genom integration av (1) enligt Simpsons regel

$$\begin{cases} \hat{x}(k) \approx \hat{x}(k-1) + \frac{T_c}{2}(v(k)\cos\theta(k) + v(k-1)\cos\theta(k-1)) \\ \hat{y}(k) \approx \hat{y}(k-1) + \frac{T_c}{2}(v(k)\sin\theta(k) + v(k-1)\sin\theta(k-1)) \\ \hat{\theta}(k) \approx \hat{\theta}(k-1) + \frac{T_c}{2}(\omega(k) + \omega(k-1)). \end{cases} \tag{8}$$

Definierar vi D_x och D_y som komponenterna av avståndet mellan robotens mittpunkt och avståndssensorer enligt

e-post: ollul666@student.liu.se
isast763@student.liu.se

$$\begin{cases} D_x = \Delta_y \sin\theta - \Delta_x \cos\theta \\ D_y = -\Delta_y \cos\theta - \Delta_x \sin\theta \end{cases}$$

fås följande uttryck för $z(k)$

$$z(k) \approx |(x(k) + D_x - x_m)\sin\gamma - (y(k) + D_y - y_m)\cos\gamma|. \tag{9}$$

4 Kunskapsbas – kartläggning och vägoptimering

Nedan presenteras en kunskapsbas för optimeringen.

4.1 Jacobianen

Låt $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$ vara en vektorvärd funktion där $\mathbf{f} = (f_1, \ldots, f_m)$ och $\mathbf{x} = (x_1, \ldots, x_n)$. Jacobimatrisen till $\mathbf{f}$ är av storlek $m \times n$ och definieras enligt

$$\mathbf{J} = \frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

och betecknas ofta $\frac{\partial(f_1,\ldots,f_m)}{\partial(x_1,\ldots,x_n)}$.

4.2 Spår

Låt $\mathbf{A}$ vara en kvadratisk matris av storlek n med elementen $a_{i,j}$. Spåret (engelska *trace*) av $\mathbf{A}$ definieras då enligt

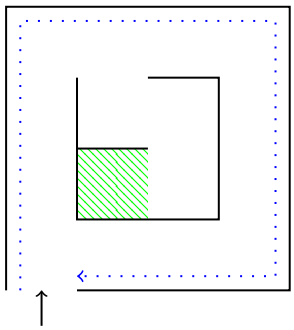
$$\operatorname{tr}\mathbf{A} = \sum_{k=1}^{n} a_{k,k}$$

och utgör summan av diagonalelementen.

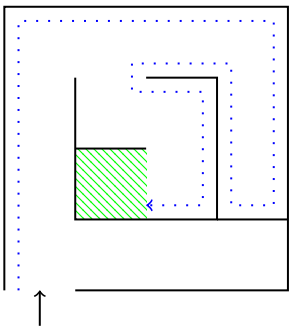
4.3 Enkelt sammanhängande labyrint

En labyrint kallas enkelt sammanhängande om alla väggar är kopplade till varandra. Se figur 2 för en grafisk beskrivning.

e-post: ollul666@student.liu.se
isast763@student.liu.se



(a) En inte enkelt sammanhängande labyrint



(b) En enkelt sammanhängande labyrint

Figur 2: Två exempel på labyrinter och den rutt som skulle tas av en väggföljningsalgoritm av vänstertyp

5 Empiri – reglering

För differentiellt driven robot av det slag som redan introducerats i den här rapporten är reglerad körning i en rak korridor en betydlig större utmaning än reglering av rotation. Förutsatt att de fyra motorernas egenskaper är relativt likvärdiga, vilket de antas vara, kan förflyttningen i x- och y-led vid rotation försummas. Detta leder till att reglerproblemet för rotation enbart innefattar vinkel och vinkelhastighet, varför det kan lösas med enkel PD-reglering.

Med detta i åtanke kommer detta avsnitt främst behandla problemet med reglerad körning i en rak korridor.

5.1 Återkoppling med riktningskrav

Som nämnt i avsnitt 3.2.2 ger IR-sensorerna endast signifikanta mätvärden då deras riktning uppfyller kravet i (6). En ändring av återkopplingen (4) ger

$$\begin{cases} v = v_{des} \\ \omega = -\frac{k(d-d^0)}{v_{des}} - (\beta_0 + \beta_1 |d - d^0|) \tan(\theta - \gamma) \end{cases} \tag{10}$$

där β_0, β_1 är skalärer.

Sats 1 *Bemporad et al föreslår i [?] följande sats. Betrakta det slutna systemet* (1), (10) *och låt* $\beta_0 \geq 0$,

$$\beta_1 \triangleq \frac{k}{v_{des} \tan \sigma_{max}} \tag{11}$$

Då kan, för varje initialvärde $(x(0), y(0), \theta(0)$ *med* $|\theta(0) - \gamma| \leq \sigma_{max}$*, RWFP lösas, eftersom*

e-post: ollul666@student.liu.se
isast763@student.liu.se

$$|\theta(t) - \gamma| \leq \sigma_{max} \tag{12}$$

för alla $t \geq 0$*, och*

$$\begin{aligned} d(t) &\to d^0 \\ \dot{d}(t) &\to 0 \\ \theta(t) &\to \gamma \end{aligned} \tag{13}$$

då $t \to \infty$*.*

Bevis 1 Se avsnitt 3.1 i [**?**].

Kommentar 1 Sats 1 ger inget specifikt värde på β_0. Bemporad et al. skriver i [**?**] att för små värden på β_0 är vinkelhastighetens storlek tillräckligt liten för att undvika plötsliga rotationer både i stabila tillstånd och transienter. Större värden på β_0 ger dock större robusthet mot brus och modellfel. Parametern måste således justeras experimentiellt för den specifika applikationen. I [**?**] användes formeln $\beta_0 = \alpha\beta_1$ med $\alpha \in [0.05, 0.1]$.

5.2 Hastighetskrav

Återkopplingen i (10) betraktar inte de fysiska begränsningarna på hjulens vinkelhastighet (och därmed robotens hastighet) som visas i (7). Dessa krav kan emellertid uppfyllas genom att direkt skala av utsignalen enligt

$$\omega = \lambda\omega_\gamma \tag{14}$$

$$v = \lambda v_{des} \tag{15}$$

där ω_γ ges av högerledet i (10) och $v_{des} < r\Omega_{max}$ är den önskade linjära hastigheten. Bemporad et al. skriver i [**?**] att tanken är att välja $\lambda \in [0, 1]$ direkt så att kravet i (7) uppfylls. Detta görs genom att sätta

$$\lambda \triangleq \max\left\{1, \left|\frac{v_{des} \pm e\omega_\gamma}{r\Omega_{max}}\right|\right\}^{-1} \tag{16}$$

vilket uppfyller kraven i (7) utan att påverka regleringens asymptotiska egenskaper [**?**].

e-post: ollul666@student.liu.se
isast763@student.liu.se

5.3 Utökat *Kalmanfilter*

I [**?**] föreslår Bemporad et al. användningen av ett utökat *Kalmanfilter* (EKF) för att skatta robotens koordinater (x, y, θ). Låt oss införa vektorerna

$$X(k) \triangleq [x(k), y(k), \theta(k)]^{\mathrm{T}}$$
$$V(k) \triangleq [v(k), \omega(k)]^{\mathrm{T}}$$

and definiera $r(X(k))$ som avståndet från IR-sensorn till väggen, det vill säga

$$r(X(k)) \triangleq |(x(k) + D_x - x_m)\sin\gamma - (y(k) + D_y - y_m)\cos\gamma|.$$

Vidare, låt $E_x(k)$ vara en stokastisk vektor (med väntevärde noll och kovariansmatris $Q_x(k)$) som tar brus och modellfel i beaktning och $\xi(k)$ en stokastisk variabel (med väntevärde noll och kovarians σ_r^2) som modellerar bruset som påverkar avståndsmätningen.

Baserat på den icke-linjära modellen

$$\begin{cases} X(k) = F(X(k-1), V(k)) + E_x(k) \\ z(k) = r(X(k)) + \xi(k) \end{cases} \tag{17}$$

ges uppdateringsekvationerna för det tillhörandet EKF:et av

$$\begin{aligned} P_x(k|k-1) &= J_x(k-1)P_x(k-1|k-1)J_x'(k-1) + \\ &\quad + J_v(k)P_{v_r}(k|k) + Q_x(k) \\ G_x(k) &= \frac{P_x(k|k-1)H'(k)}{H(k)P_x(k|k-1)H'(k) + \sigma_r^2} \\ P_x(k|k) &= [I - G_x(k)H(k)]P_x(k|k-1) \\ \hat{X}(k|k-1) &= F\Big(\hat{X}(k-1|k-1), \hat{V}(k|k)\Big) \\ \hat{X}(k|k) &= \hat{X}(k|k-1) + G_x(k)\Big[z(k) - r(\hat{X}(k|k-1))\Big] \end{aligned}$$

där $P_{v_r}(i|j)$ är den skattade kovariansmatrisen för $V(k)$ vid tiden i baserat på information tillgänglig vid tiden j, $P_x(i|j)$ den skattade kovariansmatrisen för $X(k)$, $\hat{X}(i|j)$ skattningen av vektorn X, kalmanförstärkningen $G_x(k)$ och jacobianen till ekvation (8) $J_V(k)$, $J_X(k)$ och $H(k)$

$$\begin{aligned} J_X(k-1) &= \frac{\partial F(X,V)}{\partial X}\Big|_{X=X(k-1),V=V(k)} \\ J_V(k) &= \frac{\partial F(X,V)}{\partial V}\Big|_{X=X(k-1),V=V(k)} \\ H(k) &= \frac{\partial r(X(k))}{\partial X}. \end{aligned}$$

Filtret initieras genom att låta

$$P_x(0|0) = Var[X(0)]$$
$$\hat{X}(0|0) = E[X(0)]$$

där $Var[X(0)]$ beaktar precisionsgraden i skattningen $E[X(0)]$ av robotens startkoordinater.

5.4 Återkoppling med EKF

Avstånd- och hastighetsmätningar matas in i återkopplingen (10), (16) på följande vis:

$$\begin{cases} v = \lambda v_{des} \\ \omega = -\lambda \left[\frac{k(z+\Delta_y - d^0)}{v_{des}} + (\beta_0 + \beta_1 |z + \Delta_y - d^0|) \tan(\hat{\theta} - \gamma) \right] \end{cases} \tag{18}$$

där z är avståndet mätt med hjälp av IR-sensorn och $\hat{\theta}$ ges av EKF. Bemporad et al. skriver i [?] att trots att skattningarna $\hat{x}$ och $\hat{y}$ inte används i (18) kan de komma till nytta för att lösa andra av robotens uppgifter.

5.5 PID-regulator

I [?], [?], [?] presenterar författarna en PID-regulator för reglering av en robot i en rak korridor. De stora fördelarna med en regulator av den här typen, jämfört med den som presenterats ovan, är att den är enklare att implementera och kräver betydligt mindre beräkningskraft.

Betrakta roboten i avsnitt 3.2. Inför konstanterna K_p, K_i och K_d och reglerfelet $e_t = d^0 - d_t$. Den diskreta PID regulatorn ges då av följande ekvation

$$d_t = K_p e_t + K_i T_s \sum_{k=0}^{t} e_k + K_d \frac{e_t - e_{t-1}}{T_s} \tag{19}$$

Denna typ av regulator är lämplig att använda vid reglering av robotens rotationer. Vi inför i så fall vinkel γ^0 som är den önskade vinkeln efter rotationen och $e_t = \gamma^0 - \gamma_t$. Då ges regulatorn av ekvationen

$$\gamma_t = K_p e_t + K_i T_s \sum_{k=0}^{t} e_k + K_d \frac{e_t - e_{t-1}}{T_s} \tag{20}$$

6 Empiri – kartläggning och vägoptimering

Autonom styrning av alla dess slag medför flera problemställningar att ta hänsyn till. Den algoritm som styr en robotgräsklippare kan exempelvis inte styra en drönare inom försvarsindustrin eller *Curiosity* som navigerar Mars. Problemen i dess grundformer handlar om att effektivt kunna tolka och kartlägga sin omgivning enbart utgående från sensorvärden.

SLAM (*Simultaneous Localization and Mapping*) är samlingsnamnet för problem av denna typ och saknar än så länge en optimal lösning. Flera SLAM-metoder har utvecklats för att lösa lokaliseringsproblemet där EKF-SLAM och Fast-SLAM utgör de mest populära enligt [**?**]. Dessa utforskas vidare i avsnitt 6.1.

Utöver att kunna kartlägga sin omgivning bör en robot även minimera sin färdsträcka genom att utnyttja en effektiv kartläggningsalgoritm. Robotens typ av omgivning ställer olika krav på komplexiteten av kartläggningsalgoritmen och i avsnitt 6.2 diskuteras ett par av dessa.

Den sista utmaningen som roboten ställs inför är att finna en optimal väg mellan två objekt förutsatt att det existerar minst en sådan. En verklig applicering av detta vore en undsättningsrobot som ska förse en nödställd med ett föremål och behöver hitta den kortaste vägen mellan ingång och nödställde. Detta problem kan formuleras som ett minkostnadsnätverk och löses med en optimerande algoritm, se avsnitt 6.3.

6.1 SLAM

Enligt [**?**] är implementeringen av en SLAM-algoritm utmanande. Ett av problemen ligger i att sensorerna som undersöker omgivningen är mottagliga för störningar och har fysiska begränsningar. En IR-sensor som fungerar på korta avstånd blir obrukbar på långa avstånd och en ultraljuds-sensor som är exakt under rätt förhållanden blir känslig för störningar. Därför, för att kompensera för dessa avvikelser, gäller att både aktuella och historiska mätningar viktas och kombineras så att omgivningen presenteras med minsta möjliga felmarginal.

En SLAM-algoritms mål är att beräkna (approximera) sannolikhetsfördelningen

$$p\left(\mathbf{x}_t, \mathbf{M} | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{x}_0\right)$$

där $\mathbf{x}_t$ är positionen vid den tidsdiskreta tiden t, $\mathbf{M} = \theta_1, \ldots, \theta_n$ är positionerna för en mängd av landmärken, $\mathbf{Z}_{0:t}$ är en mängd av observationer från $0, \ldots, t$, $\mathbf{U}_{0:t}$ är en mängd av styrsignaler från $0, \ldots, t$ och $\mathbf{x}_0$ är den ursprungliga positionen. Vidare undersöks och jämförs två algoritmer som löser SLAM-problemet med olika angreppssätt.

6.1.1 *Kalmanfilter* och EKF-SLAM

EKF-SLAM är en SLAM-algoritm som använder ett *Extended Kalman Filter*, en icke-linjär utvidgning av ett *Kalmanfilter*, för att i varje tidssteg linjärisera en övergångs- och observationsmodell.

Det ursprungliga *Kalmanfiltret* bygger på en linjär övergångs- och observationsmodell enligt

$$\begin{aligned}\mathbf{x}_t &= \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \boldsymbol{\epsilon}_t\\ \mathbf{z}_t &= \mathbf{C}_t\mathbf{x}_t + \boldsymbol{\delta}_t\end{aligned}$$

där $\mathbf{x}_t$ är det nuvarande tillståndet, $\mathbf{A}_t$ en matris, $\mathbf{x}_{t-1}$ det föregående tillståndet, $\mathbf{B}_t$ en matris, $\mathbf{u}_t$ nuvarande styrsignal, $\boldsymbol{\epsilon}_t$ en brusterm, $\mathbf{z}_t$ nuvarande observation, $\mathbf{C}_t$ en matris och $\boldsymbol{\delta}_t$ en brusterm. Brusen $\boldsymbol{\epsilon}_t$ och $\boldsymbol{\delta}_t$ är oberoende, normalfördelade, vita och med väntevärde noll samt kovarians $\mathbf{R}_t$ respektive $\mathbf{Q}_t$, det vill säga $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{R}_t)$ och $\boldsymbol{\delta}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$.

Eftersom *Kalmanfiltret* är en rekursiv estimator skattar den nästkommande tillstånd genom att bygga vidare på information från föregående tillstånd. Låt $\hat{\mathbf{x}}_{t|\tilde{t}}$ vara skattningen av tillståndet $\mathbf{x}$ och $\mathbf{P}_{t|\tilde{t}}$ vara kovariansmatrisen för felet i skattningen av tillståndet vid tiden t givet ovservationer upp till och med tiden $\tilde{t} \leq t$. Då ges skattningen av dessa variabler i tidpunkt t av

$$\begin{aligned}\hat{\mathbf{x}}_{t|t-1} &= \mathbf{A}_t\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t\\ \mathbf{P}_{t|t-1} &= \mathbf{A}_t\mathbf{P}_{t-1|t-1}\mathbf{A}_t^{\mathrm{T}} + \mathbf{R}_t\end{aligned}$$

där $\mathbf{P}_{t|t-1}$ härleds ur

$$\begin{aligned}\mathbf{P}_{t|t-1} &= \operatorname{cov}\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1}\right)\\ &= \operatorname{cov}\left(\mathbf{x}_t - \left(\mathbf{A}_t\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t\right)\right)\\ &\vdots\\ &= \mathbf{A}_t\,\mathbf{P}_{t-1|t-1}\,\mathbf{A}_t^{\mathrm{T}} + \mathbf{R}_t.\end{aligned}$$

Vidare uppdateras $\hat{\mathbf{x}}_{t|t}$ och $\mathbf{P}_{t|t}$ enligt

$$\begin{aligned}\hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\left(\mathbf{z}_t - \mathbf{C}_t\hat{\mathbf{x}}_{t|t-1}\right)\\ \mathbf{P}_{t|t} &= \left(\mathbf{I} - \mathbf{K}_t\mathbf{C}_t\right)\mathbf{P}_{t-1|t-1}\end{aligned}$$

där $\mathbf{I}$ är enhetsmatrisen och $\mathbf{K}_t$ väljs så att medelkvadratfelet hos $\mathbf{P}_{t|t}$ minimeras, vilket är ekvivalent med att minimera spåret av $\mathbf{P}_{t|t}$ med hänsyn till $\mathbf{K}_t$. Alltså väljs $\mathbf{K}_t$ så att

$$\frac{\partial \operatorname{tr}\left(\mathbf{P}_{t|t}\right)}{\partial\,\mathbf{K}_t} = 0$$

enligt [**?**], vilket leder till

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{C}_t^{\mathrm{T}}\left(\mathbf{C}_t\mathbf{P}_{t|t-1}\mathbf{C}_t^{\mathrm{T}} + \mathbf{Q}_t\right)^{-1}$$

där härledningen återfinns i [**?**].

Den utvidgning som görs av EKF är att övergångs- och observationsmodellen antas se ut enligt

$$\begin{aligned}\mathbf{x}_t &= f\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right) + \boldsymbol{\epsilon}_t\\ \mathbf{z}_t &= h\left(\mathbf{x}_t\right) + \boldsymbol{\delta}_t\end{aligned}$$

där f och h är differentierbara men inte nödvändigtvis linjära. De nya skattningarna av $\hat{\mathbf{x}}_{t:|t-1}$ och $\mathbf{P}_{t|t-1}$ ges då av

$$\hat{\mathbf{x}}_{t|t-1} = f\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t\right)$$
$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \, \mathbf{P}_{t-1|t-1} \, \mathbf{F}_t^{\mathrm{T}} + \mathbf{R}_t$$

där $\mathbf{F}_t = \frac{\partial f}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t}$ utgör jacobianen till f. Vidare uppdateras $\hat{\mathbf{x}}_{t|t}$ och $\mathbf{P}_{t|t}$ enligt de nya sambanden

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t \left(\mathbf{z}_t - h\left(\hat{\mathbf{x}}_{t|t-1}\right)\right)$$
$$\mathbf{P}_{t|t} = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \mathbf{P}_{t|t-1}$$

där $\mathbf{H}_t = \frac{\partial h}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{t|t-1}}$ utgör jacobianen till h och $\mathbf{K}_t$ väljs till
$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t^{\mathrm{T}} \left(\mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^{\mathrm{T}} + \mathbf{Q}_t\right)^{-1}$ för att minimera spåret av $\mathbf{P}_{t|t}$ med hänsyn till $\mathbf{K}_t$.

EKF har fått sin populäritet från att den passar i sammanhang då modellen, mätningar eller bådadera är icke-linjära och appliceras exempelvis i navigationssystem (GPS) enligt [**?**]. Nackdelar med algoritmen är bland annat beräkningskapaciteten som behövs. I komplexitet är algoritmen av storlek $\mathcal{O}\left(n^2\right)$ där n är antal landmärken. En vidareutveckling av EKF finns i [**?**] där komplexiteten har minskats till $\mathcal{O}\left(n\right)$.

6.1.2 FastSLAM

FastSLAM är en alternativ lösning på SLAM-problemet som bygger på partikelfiltrering. Ett partikelfilter har inget krav på korrelation mellan landmärkena och behöver därför inte beräkna kovariansmatrisen vilket sänker beräkningskapaciteten för algoritmen då SLAM-problemet ska lösas. Istället utgör varje partikel en unik skattning av robotens position där de partiklar som anses ha en rimlig position omsamplas och de andra dör ut.

Precis som EKF-SLAM så använder FastSLAM ett *Extended Kalman filter* för att skatta landmärkenas position. För att minska matrisernas dimension och därmed algoritmens beräkningskapacitet använder FastSLAM istället ett filter per landmärke. Kartläggningen består med andra ord av en mängd oberoende fördelningar med linjär komplexitet istället för en kovarierande karta med kvadratisk komplexitet.

Med utgångspunkt i grundproblemet så beräknar FastSLAM position och karta enligt

$$\begin{aligned} p\left(\mathbf{x}_t, \mathbf{M} | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{x}_0\right) &= p\left(\mathbf{x}_t | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{x}_0\right) p\left(\mathbf{M} | \mathbf{Z}_{0:t}, \mathbf{x}_t, \mathbf{x}_0\right) \\ &= p\left(\mathbf{x}_t | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{x}_0\right) \prod_{i=1}^{n} p\left(\theta_i | \mathbf{Z}_{0:t}, \mathbf{x}_t, \mathbf{x}_0\right) \end{aligned}$$

där sista likheten gäller på grund av obereoende-egenskapen hos fördelningarna. Problemet har nu övergått till att bestå av $n+1$ antal subproblem. Dessa är enklare att lösa individuellt och sätta samman än att lösa grundproblemet.

Utmaningen för algoritmen blir istället att intelligent välja ut vilka partiklar som ska samplas och vilka som ska ignoreras. Grisetti, et al. föreslår i [**?**] en algoritm för att filtrera

bort partiklar med en hög mängd brus. Genom att beräkna en tröskelnivå som beror på både mängden partiklar och dess vikt, kan samtliga partiklar som överskrider nivån ignoreras.

Av komplexitetsskäl gör sig FastSLAM bättre på stora system jämfört med EKF-SLAM som inte skalar lika väl. Att omsampla partiklar utgör flaskhalsen för FastSLAM vilket görs med komplexitet $\mathcal{O}(n_p \log(n_m))$ där n_p är antalet partiklar och n_m är antalet landmärken.

6.1.3 En jämförelse mellan EKF-SLAM och FastSLAM

EKF-SLAM bygger på en tillståndsvektor som innehar information om robotens position och alla landmärken. FastSLAM betraktar istället en mängd partiklar där varje partikel har en egen uppsättning med position och landmärken. EKF-SLAM behöver uppdatera sin vektor i varje tidssteg medan FastSLAM endast behöver sampla om de partiklar som algoritmen anser korrekta.

För EKF-SLAM utgörs alla skattningar av normalfördelningar. FastSLAM delar upp skattningsproblemet i $n + 1$ subproblem och löser dessa individuellt. EKF-SLAM förutsätter med andra ord att bruset är normalfördelat medan FastSLAM inte gör något antagande om brusets fördelning.

6.2 Labyrintnavigering

Att autonomnt navigera en labyrint kräver någon typ av smart algoritm. Vilken typ av algoritm som passar sig beror på både labyrintens komplexitet och robotens målsättning. Den typ av labyrint som studeras i detta avsnitt har inga öppna rum, en ingång och en markering istället för utgång. Alla korsningar och svängar är ortogonala och det kan finnas mer än en väg mellan ingång och markering. Robotens målsättning är att ta sig från början av labyrinten och åtminstone kartlägga den kortaste vägen till markeringen.

I detta avsnitt behandlas algoritmer för hur en robot kan fatta beslut under kartläggning. Beräkning av den optimala vägen återfinns i avsnitt 6.3.

6.2.1 Väggföljare

Ifall labyrinten är enkelt sammanhängande räcker en väggföljar-algoritm för att kartlägga alla korridorer. En väggföljar-algoritm är antingen av höger- eller vänstertyp och bygger på att man alltid följer en sida av korridoren. Ifall algoritmen är av högertyp följer man höger sida av korridoren och svänger höger där det är möjligt, vice versa för vänstertyp.

Fördelar med algoritmen är att den är enkel att implementera. Men saknandet av intelligens gör att utforskandet av labyrinten tar lång tid. Algoritmen klarar endast att utforska de allra enklaste typer av labyrinter och ifall labyrinten inte är enkelt sammanhängande kan roboten fastna i en oändlig loop.

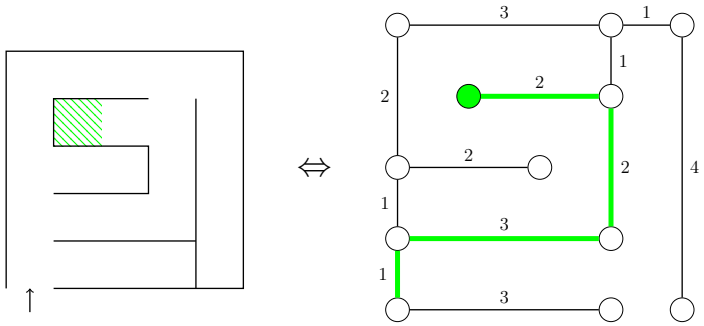
6.2.2 *Dead-end filling*

Dead-end filling är en djup-först-algoritm som utforskar en väg tills en återvändsgränd upptäcks. Därefter åker roboten tillbaka till den första korsningen som har outforskade utgångar och kartlägger dessa tills en återvändsgränd upptäcks. Denna process återupprepas tills hela labyrinten är kartlagd.

En vidareutveckling av dessa algoritmer återfinns i [**?**] där mer komplexa labyrinter studeras.

6.3 Kortaste-väg-beräkning

Givet att vi har kartlagt labyrinten återstår uppgiften att finna den optimala vägen mellan ingång och markering. Väljer vi att representera kartan som noder (korsningar samt återvändsgränder) och bågar (korridorer mellan korsningar) kan existerande algoritmer utnyttjas för att finna den kortaste vägen mellan två noder. Figur 3 visar ett exempel på denna översättning från labyrint till nätverk.



Figur 3: En labyrint och dess motsvarande nätverk. Bågkostnaderna är det relativa avståndet och den optimala vägen är fetmarkerat grön.

6.3.1 Dijkstras algoritm

Dijkstras algoritm är en girig, bäst-först-algoritm. Genom algoritmen beräknar man den billigaste (kortaste) vägen från startnoden till samtliga noder och den fungerar på nätverk med icke-negativa bågkostnader.

Vid starten definieras två mängder, en mängd som innehåller de avsökta noderna och en de oavsökta noderna. De oavsökta noderna får ett oändligt nodpris och startnoden nodpriset noll. Med ursprung i startnoden sökes alla grannar genom. Ifall det nuvarande nodpriset plus bågkostnaden understiger grannens nodpris så uppdateras nodpriset med den billigaste kostnaden.

Ovanstående görs successivt med ursprung i den nod med billigast nodpris tills slutnoden har undersökts. Låt V vara mängden av samtliga noder, S mängden av avsökta noder och Q mängden av oavsökta noder. Då ges algoritmens pseudokod av algoritm 1.

e-post: ollul666@student.liu.se
isast763@student.liu.se

Algorithm 1 Dijkstras algoritm

```
1:  function DijkstrasAlgorithm(Graph, s)
2:      dist(s) ← 0
3:      for all v ∈ V − {s} do
4:          dist(v) ← ∞
5:      S ← ∅
6:      Q ← V
7:      while Q ≠ ∅ do
8:          u ← minDistance(Q, dist)              ▷ Välj nod med lägst nodpris
9:          S ← S + {u}
10:         Q ← Q − {u}
11:         for all v ∈ neighbours(u) do          ▷ Iterera över grannar till u
12:             if g(u) + w(u, v) < g(v) then
13:                 g(v) ← g(u) + w(u, v)
        return dist
```

Den kortaste vägen kan nästas upp om man även sparar vilken föregångare som bidrog till en kortare väg. I implementeringsväg är algoritmen enkel, dock kräver den mycket minne eftersom alla noder och dess nodpris behöver sparas. En vidareutveckling av Dijkstras algoritm är A* som, med hjälp av en heuristik, väljer vilka noder som ska avsökas vidare.

6.3.2 A*

A* är ytterligare en bäst-först-algoritm med målet att finna den billigaste vägen mellan två noder. I grund och botten finns samma maskineri som i Dijkstras algoritm fast där noder avsöks i stigande ordning av

$$f(n) = g(n) + h(n)$$

där $g(n)$ utgör kostnaden av rutten från startnod till nod n och $h(n)$ en heuristik som estimerar den billigaste vägen från nod n till slutnod. Med andra ord är Dijkstras algoritm ett speciallfall av A* där $h(n) = 0$.

Vilken heuristik som är optimal beror på vilket typ av problem som lösas. [?] föreslår att det euklidiska avståndet mellan noden och slutnoden ska utgöra den optimala skattningen. Men i fallet av en labyrint med ortogonala korsningar passar *Manhattan*-avståndet bättre, det vill säga summan av de ortogonala avstånden.

$h(n)$ kan ses som en straffunktion som avtar nära slutnoden. A* kan alltså, till skillnad från Dijkstras algoritm, se framåt i nätverket och kan därför ta bättre beslut om vilka noder som ska uppdateras. Låt V vara mängden av samtliga noder, S mängden av avsökta noder och Q mängden av oavsökta noder. Då ges algoritmens pseudokod av algoritm 2.

Algorithm 2 A*

```
1:  function AStarAlgorithm(Graph, s)
2:      g(s) ← 0
3:      f(s) ← g(s) + h(s)
4:      for all v ∈ V − {s} do
5:          g(v) ← ∞
6:          f(v) ← ∞
7:      S ← ∅
8:      Q ← V
9:      while Q ≠ ∅ do
10:         u ← minFvalue(Q, f)                ▷ Välj nod med lägst f(n)
11:         S ← S + {u}
12:         Q ← Q − {u}
13:         for all v ∈ neighbours(u) do       ▷ Itererar över grannar till u
14:             if g(u) + w(u, v) < g(v) then
15:                 g(v) ← g(u) + w(u, v)
16:                 f(v) ← g(v) + h(v)
        return g
```

Sammanfattningsvis utgör A* en mer optimal algoritm för en navigeringsrobot. Eftersom man endast är intresserad av avståndet från startnod till slutnod (inte från startnod till samtliga noder) sparar man både minne och tid genom att avsöka färre noder.

e-post: ollul666@student.liu.se
isast763@student.liu.se

7 Resultat och slutsatser

Nedan presenteras förstudiens resultat och slutsatser.

7.1 Reglering

Som nämnts tidigare finns det ingen anledning använda något annat än en PID-regulator för att reglera rotationen av roboten. Robotens vinkelhastighet kan enkelt mätas av gyrot och genom att driva motorerna med samma hastighet, i motsatt riktning, kan vinkelhastigheten även styras utan att påverka robotens läge i x- eller y-led. Därför räcker en PID-regulator.

Gällande reglering av rak körning är inte valet lika självklart. Tillståndsåterkoppling ger i regel bättre reglering än en PID-regulator, men är svårare att implementera och mer beräkningstung. Erfarenheter från tidigare år talar för att prestandan från en PID-regulator är tillräcklig för en robot av den här typen, varför den är att föredra. Både i fallet med rak körning och rotation är elimineringen av statiska reglerfel inte en särskilt högt prioriterad egenskap hos regulatorn och därför kan den integrerande delen av regulatorn tas bort ($K_i = 0$).

Även i frågan om att minska risken för mätstörningar prioriteras enkelhet över robusthet. De sensorer som ska användas är relativt pålitliga och den miljö roboten kommer att befinna sig i bör inte ge upphov till några extraordinära störningar. Därför anses ett EKF vara överflödigt och ersätts med ett glidande medelvärde.

7.2 Kartläggning och vägoptimering

Av de två SLAM-lösningar som behandlas i rapportens huvuddel passar ett *Kalmanfilter* bäst för en autonom robot som utforskar en labyrint. Eftersom roboten i detta projekt endast bevakar avstånden till främre, höger och vänster vägg har den relativt få landmärken att förhålla sig till. Någon minnesbegränsning kommer med andra ord inte att uppstå och *FastSLAM* läggs därför åt sidan.

Labyrinten som kartläggs byggs sådan att både höger- och vänsterväggföljning ska leda roboten till markeringen på liknande tid, se appendix C. Valet faller på en förbättrad variant av högerväggföljning. En av förbättringarna är att roboten inte behöver utforska vidare i en korridor ifall fågelavståndet mellan nuvarande position och ingång är längre än en utforskad väg till markeringen. Vidare behöver roboten inte utforska en återvändsgränd ifall den främre sensorn indikerar att det endast är en modul framöver, förutsatt att markeringen inte upptäcks i modulen.

Då roboten åtminstone har utforskat den kortaste vägen mellan ingång och markeringen optimeras den slutgiltiga vägen med hjälp av algoritmen A*. A* används eftersom roboten under kartläggningsfasen kommer att representera kartan med hjälp av noder och bågar. Att implementera A* är dessutom programmeringsmässigt enkelt.

e-post: ollul666@student.liu.se
isast763@student.liu.se

Den heuristik som används är det så kallade *Manhattan*-avståndet, det vill säga summan av de ortogonala avståndet mellan aktiv nod och slutnod. Heuristiken väljs utifrån labyrintens utseende med ortogonala korsningar.

Sammanfattningsvis kan slutsatsen dras om att SLAM-problemet förenklas betydligt av att labyrinten inte har öppna rum och består av moduler. Till skillnad från ett verkligt scenario så har roboten i detta projekt alltid en vägg framåt att förhålla sig till. Det blir därför enklare att beräkna tillryggalagd sträcka. Att labyrinten inte innefattar öppna rum innebär att optimerande algoritmer som Dijkstras och A* kan användas. Vid kartläggningen kan roboten alltid följa en vägg och behöver inte utforska några öppna ytor.

Resultatet av förstudien är att både kartläggning och vägoptimering är möjligt att implementera i projektets robot.

e-post: ollul666@student.liu.se
isast763@student.liu.se