



Mapmaster2001

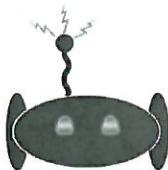
12 maj 2014

# Simultan positionering och kartläggning

Tobias Grundström och Hans-Filip Elo  
Kandidatprojekt Y - Grupp 8 - VT2014  
Version 0.5

Status

Granskad	hanel742	12 maj 2014
Godkänd	-	-



# PROJEKTIDENTITET

Grupp 8, 2014/VT, MapMaster2001  
Linköpings tekniska högskola, Institutionen för Systemteknik (ISY)

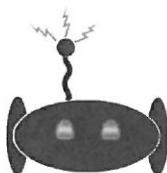
Namn	Ansvar	Telefon	E-post
Jens Edhammar	Dokumentansvarig (DOK)	076-030 67 80	jened502@student.liu.se
Erik Ekelund	Designansvarig (DES)	073-682 43 06	eriek984@student.liu.se
David Habrman		976-017 71 15	davha227@student.liu.se
Tobias Grundström	Testansvarig (TES)	073-830 44 45	tobgr602@student.liu.se
Hans-Filip Elo		073-385 22 32	hanel742@student.liu.se
Niklas Ericson	Projektledare (PL)	073-052 27 05	niker917@student.liu.se

**E-postlista för hela gruppen:** mapmaster2001@cyd.liu.se

**Kund:** Mattias Krysander, Linköpings universitet, 581 83 LINKÖPING,  
013-28 21 98, matkr@isy.liu.se

**Kontaktperson hos kund:** Mattias Krysander, 013-28 21 98, matkr@isy.liu.se  
**Kursansvarig:** Tomas Svensson, 3B:528, 013 28 21 59, tomass@isy.liu.se

**Handledare:** Peter Johansson, 013-28 1345 peter.a.johansson@liu.se



## Innehåll

<b>1 Inledning . . . . .</b>	<b>1</b>
1.1 Syfte . . . . .	1
1.2 Historia . . . . .	1
<b>2 Problemformulering . . . . .</b>	<b>2</b>
2.1 Mätningar . . . . .	3
2.1.1 Avståndssensor . . . . .	3
2.1.2 Gyroskop och accelerometer . . . . .	4
2.1.3 Hjulsensorer . . . . .	5
2.1.4 Databehandling . . . . .	5
2.1.5 Andra typer av sensorer . . . . .	5
<b>3 Skattningsmetoder för SLAM-problemet . . . . .</b>	<b>7</b>
3.1 Tillståndsrepresentation av reglersystem . . . . .	7
3.1.1 Tillståndsåterkoppling och observatörer . . . . .	8
3.1.2 Kalmanfilter . . . . .	8
3.1.3 EKF - Det utökade kalmanfiltret . . . . .	9
3.2 FastSLAM . . . . .	10
<b>4 Slutsats . . . . .</b>	<b>11</b>
<b>5 Reflektion . . . . .</b>	<b>11</b>
<b>Källförteckning . . . . .</b>	<b>12</b>



# 1 Inledning

Simultan positionering och kartläggning (SLAM) är ett problem som grundar sig i följande frågeställning: Utan att veta var vi är - hur kartlägger vi då vår omgivning? Åt andra hållet får man ställa sig frågan - hur vet vi var vi befinner oss utan en karta?

Det är inte helt enkelt att lösa dessa frågor, men det finns approximativa lösningar på SLAM-problemet. Gemensamt för alla lösningar är att de bygger på möjligheten att läsa av sin omgivning i kombination med sannolikhetsteori. Då sensordata aldrig kan antas vara exakt använder man sannolikhetsteori för att göra rimlighetsbedömningar i de stickprov av mätningar sensorerna ger.

SLAM-problemet är alltså oftast inte entydigt lösbart rent matematiskt, utan bygger på sannolikhetsteori i kombination med att moderna processorer och minnen kan hantera en stor mängd data. Moderna processorer möjliggör alltså ett stort stickprov vilket kan leda till en mindre osäkerhet.

## 1.1 Syfte

Syftet med denna rapport är att ge läsaren en introduktion till de algoritmer och tekniker som används för kartläggning och positionsbestämning i ett mikroprocessorsystem.

## 1.2 Historia

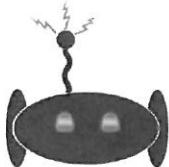
Principerna för SLAM formulerades för första gången 1986<sup>1</sup>. Redan vid formuleringen av problemet beskrevs SLAM som en inexakt vetenskap. SLAM handlar om att skaffa sig en approximativ uppfattning av sin omgivning och position som är tillräckligt bra för att fatta ett beslut kring färdväg och/eller kartläggning.

Utvecklingen på området har sedan dess accelererat kraftigt tack vare mikrokontrollers förmåga att hantera mer data. Då sannolikhetsberäkningarna i SLAM inbefattar gynnas kraftigt av att arbeta med stora stickprov.

På senare år har man, precis som inom många andra vetenskapliga områden, sett utvecklingen ta ytterligare ett kliv tack vare internet och öppna källkodsprojekt som Github och OpenSLAM. Att göra en sökning på SLAM på Github resulterar i en mängd aktiva projekt på området. Eftersom källkoden där också finns tillgänglig är detta ett utmärkt exempel för de som vill studera implementeringar av SLAM-algoritmer.

<sup>1</sup>Smith, R.C.; Cheeseman, P. (1986). "On the Representation and Estimation of Spatial Uncertainty". The International Journal of Robotics Research, 5(4), sida 56–68. Hämtad 28 mars 2014

Läs om utveckling i literatur här!



## 2 Problemformulering

Vanligtvis där SLAM implementeras ska en robot försöka kartlägga sin omgivning och utföra en uppgift. Antag att en robot med fyra hjul och tillgång till ett antal avståndssensorer och ett gyro placeras i ett okänt slutet område. Den ska, med en vägg som startpunkt, helt autonomt kartlägga området genom att markera ut var väggar finns och var det finns områden som inte går att nå. Om inte hela det slutna området är kartlagt ska den upptäcka vilka segment som är upptäckta, färdas dit och lägga till detta i kartan. När området är kartlagt ska roboten ta sig tillbaka till startpunkten för att kunna återhämtas och på så sätt avsluta arbetet. Marken under roboten antas vara slät med förhållandevis stor friktion mot robotens däck.

SLAM är en teknik där ett system på något sätt kan uppfatta landmärken. Med landmärken menas alla typer av objekt eller liknande som kan användas som referenser. Landmärkenas position i förhållande till roboten ska noteras och uppmäts med regelbundna samplingar. Genom att positionsbestämma dessa landmärken kan roboten snabbare bestämma sin position.

Tänkbar definition av problemet kan vara.

$$\begin{aligned}x_t^r &= \{\text{robotens position i förhållande till utgångspunkt } r \text{ vid tidpunkten } t\} & (1) \\m &= \{\text{landmärkens position}\} \\u_t &= \{\text{styrsignaler}\} \\y_t &= \{\text{mätningar med sensorer}\}\end{aligned}$$

Målet är att skatta robotens position ~~vid aktuell tidpunkt~~ <sup>och de</sup> ~~eventuella~~ <sup>och att</sup> ~~styrstörningar~~ <sup>att</sup> ~~positioner~~ <sup>system</sup> ~~gives~~ <sup>ges</sup> ~~utan~~ <sup>utan</sup> ~~att~~ <sup>att</sup> ~~ha~~ <sup>ha</sup> ~~statisk~~ <sup>statisk</sup> ~~position~~ <sup>position</sup>.

(Bestyrande text: ) I många fall kan man beskriva SLAM problemet med ~~hjältestandomdelen~~

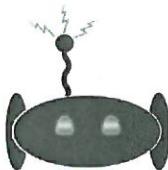
$$x_t^r = f(x_{t-1}^r, u_{t-1}, v_{t-1}) \quad (5)$$

$$m_t = m_{t-1} \quad (6)$$

$$y_t = h(x_t^r, m_t^i) + e_t \quad (7)$$

Vad är det då för typ av mätningar som kan ~~tas in~~? För att kunna utföra SLAM krävs det att man gör någon form av odometri, det vill säga att man kontinuerligt uppskattar vägen som färdats relativt sin senaste position. Odometri kan göras på olika sätt - exempelvis genom att ~~visuellt~~ mäta ~~avstånd till objekt i sin omgivning~~, vinkelhastigheter ~~på~~ hjul med känd storlek eller steg med given längd.

Utöver relativa mätningar över kortare ~~stunder~~ används också absoluta mätningar ~~till~~ <sup>av</sup> ~~sitt~~ <sup>av</sup> omgivning för att korrigera de fel som relativa mätningar orsakar över tid. Många



robotar nyttjar förmågan att optiskt mäta objekt i sin omgivning, men det förekommer även implementeringar med radar, sonar eller GPS.

- ✗ Med absoluta mätningar till omvärlden finns alltid möjligheten att korrigera tidigare felaktiga mätvärden genom att samla in mer mätdata för att på ett korrektare sätt kunna beskriva sin omgivning. Av den anledningen är all typ av SLAM beroende av att på något sätt granska sin omgivning.

## 2.1 Mätningar

- ✗ För att kunna avgöra var robotten befinner sig behövs ett eller flera sätt för att känna sin omgivning och sitt tillstånd. Robotten kan till exempel behöva utföra mätningar ~~på~~ <sup>av</sup> avstånd, hastighet och vinkelhastighet för att åstadkomma detta.

### 2.1.1 Avståndssensor

Som avståndsmätare kan någon typ av optisk sensor användas. Att sensorn är optisk innebär att den använder sig av ljus för att utföra mätningarna. De finns till exempel sensorer som belyser en yta som avstånd ska mätas till, tar emot reflekterat ljus och sedan med hjälp av tiden det tagit för ljuset att färdas beräkna avståndet. Denna metod kallas time-of-flight. Ytterligare en metod som används kallas optisk triangulering och det innebär att med hjälp av vinkeln mellan sändare och mottagare bestämma avståndet till objektet.



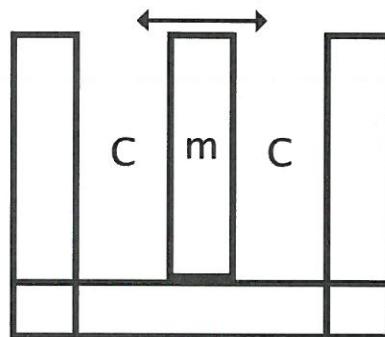
### 2.1.2 Gyroskop och accelerometer

Gyroskop och accelerometrar kan användas i samverkan för att bestämma relativ rörelse. En gyroskop kan känna av ett objekts vinkelhastighet kring tre axlar. En accelerometer kan i sin tur känna av ett objekts förändring i hastighet i alla tre dimensioner. Förutsatt att initiala hastigheten är känd kan en accelerometer användas för att beräkna färdad sträcka genom tvåstegs tidsintegration.

*Pacelations-  
varvande* Gemensamt för gyroskop och accelerometrar är att de använder sig av liknande tekniker. Antingen består de av en eller flera massor i ledande material som i kombination med ett hålrum skapar en kapacitans<sup>2</sup>. En annan teknik är att använda piezoelektriska material för att generera en förändring i kapacitans då en kraft verkar på en massa.

För en accelerometer används en massa som hålls fast av en arm med en viss elasticitet. Massan omges av strömförande ledare och skapar tillsammans med dem en induktans. Då accelerometern rör sig förändras kapacitansen - vilket är mätbart. Principen illustreras väldigt enkelt i bilden nedan.

*Figu 1*



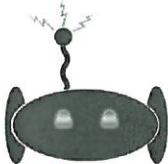
Figur 1: Modell av accelerometer

En gyrosensor av MEMS-typ (Micro Electro-Mechanical System) utnyttjar att två lika stora massor som oscillerar konstant i motsatta riktningar och sedan påverkas av Korioliskraften vid rotation. Kraften verkar i olika riktningar på de två massorna vilket leder till en förändring i kapacitans. Skillnaden i kapacitans har visat sig vara proportionell mot vinkelhastigheten, och därför går det att bestämma vinkelhastigheten med hjälp av ett gyro<sup>4</sup>.

<sup>2</sup>Weinberg, Harvey (1999), "Dual Axis, Low g, Fully Integrated Accelerometers", <http://www.analog.com/library/analogdialogue/archives/33-01/accel/index.html>. Hämtad 2014-05-12.

<sup>3</sup>Piezoelectricity, Wikipedia. <http://en.wikipedia.org/wiki/Piezoelectricity>. Hämtad 2014-05-12.

<sup>4</sup><http://electroiq.com/blog/2010/11/introduction-to-mems-gyroscopes/>, hämtad 2014-05-06



### 2.1.3 Hjulsensorer

Vid vissa tillfällen kan det vara så att avståndssensorer ej kan användas för att skatta rörelse relativt landmärken, exempelvis om alla landmärken är utanför sensorernas räckvidd. Man kan då behöva skatta sin rörelse med hjälp av någon annan sensor. En vanlig typ av sensor är då hjulsensorer. Hjulsensorer räknar hur många varv hjulen snurrat och skattar sedan avståndet roboten färdats utifrån detta.

Ett problem med hjulsensorer är att de ~~ej~~ ger en helt korrekt hastighets- och distansskattning då hjulen ofta slirar mot underlaget. Om underlaget är väldigt poröst är hjulsensorer ~~ej~~ särskilt tillämpbara. Det är då istället bättre att använda sig av en accelerometer för att skatta sin relativ rörelse.

*Det här avsnittet passar inte riktigt in... Annars inte heller. Förslag: Släga!*

### 2.1.4 Databehandling

Sensorerna producerar mängder med data som kan variera något vid varje mätning. Mätningar från sensorerna kan antagligen hämtas väldigt fort. Om man utgår från att en klump närliggande mätningar görs kring samma position för roboten kan den klumpen av mätningar, eller stickprovet, antas vara normalfördelade kring korrekta värdet.

Korrekte avståndet kan alltså skattas med hjälp av stickprovets medelvärde,  $\mu$ .

### 2.1.5 Andra typer av sensorer

I denna rapport berör vi mest området där SLAM baseras på optiska avståndssensorer, men det finns även andra typer av sensorer som går att använda. Till exempel finns det implementeringar som använder sig av kameror för att finna landmärken och sedan använda dessa som referenser. Det är dock mer avancerade algoritmer som krävs för att finna lämpliga landmärken.<sup>5</sup>

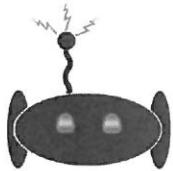
Något som också provats är SLAM utan användning av någon typ av optisk utrustning och istället förlita sig på känsel, med hjälp av känselsensorer som efterliknar djurs morrhår, för att kunna kartlägga ett helt mörklat rum. Denna metod ger dock inte så bra resultat med de tekniker som existerar i dag<sup>6</sup>.

*(eller  
som  
ett nytt  
avsnitt)*

**Absolut positionsskattning** Tidigare nämndes exempel på metoder som använder sig av GPS. Detta i kombination med WiFi-signaler har använts till exempel i mobiltelefoner för att ge en positionsskattning med absolut referens för var enheten befinner sig. För att

<sup>5</sup>Karlsson, N.; Goncalves, L.; Munich, M.E.; Pirjanian, P."The vSLAM Algorithm for Navigation in Natural Environments". Evolution Robotics, Inc. Hämtad 28 mars 2014

<sup>6</sup>Fox, C.; Evans, M.; Pearson, M.; Prescott, T. (2012) "Tactile SLAM with a biomimetic whiskered robot". 2012 IEEE International Conference on Robotics and Automation. Hämtad 28 mars 2014. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6224813>



WiFi-SLAM ska fungera krävs att accesspunkten har information om positionsdata för sig själv, alternativt att den anslutna enheten har information om var aktuellt WiFi är tillgängligt.

I tillämpningar där positionen inte behöver skattas med så hög nogrannhet är en GPS fullt tillräcklig och SLAM-algoritmer behöver därför inte implementeras. Om man dock ändå behöver mer nogrannhet än GPS kan uppnås detta genom att GPS användas som ett hjälpmittel för att snabbt uppskatta inom vilket område man befinner sig. Detta för att kraftigt förenkla SLAM-algoritmen.





### 3 Skattningsmetoder för SLAM-problemet

I denna fördjupande del går vi igenom metoder som kan användas för att utföra SLAM i en miljö lik den i problemställningen. Bland annat kommer tillståndsrepresentation och skattning av tillstånd med hjälp av observatörer samt Kalmanfilter att tas upp.

#### 3.1 Tillståndsrepresentation av reglersystem

Inom reglerteknik kan linjära system beskrivas på så kallad tillståndsform<sup>7</sup>. I fallet med SLAM för en robot använder man robotens position som tillstånd som beskrivs i ekvation (1). Ett tidskontinuerligt linjärsystem kan beskrivas på tillståndsform enligt

$$\dot{x} = Ax + Bu \quad \text{linjärt} \quad \text{och orientering} \quad (8)$$

$$y = Cx + Du \quad (9)$$

där  $A$ ,  $B$ ,  $C$  och  $D$  är matriser.

Tillstånden för en robot mäts vid tidpunkter vanligt  
med tidsdiskreta värden då datorer endast hanterar tidsdiskreta mätningar och ej kontinuerliga. I dessa fall är det bättre att använda sig av en tidsdiskret tillståndsbeskrivning

$$x[n+1] = Ax[n] + Bu[n] \quad \text{slår på samma sätt över tillståndet} \quad (10)$$

$$y[n] = Cx[n] + Du[n] \quad (\text{Nu } x_n; (5)-(7) \text{ men } x[n] \text{ här}) \quad (11)$$

Om systemet istället skulle vara icke-linjärt så går det inte att beskriva det på samma sätt. Det kan dock approximeras genom att linjärisera systemet genom att använda sig av Taylorutveckling, vilket i slutändan leder till att  $A$ -,  $B$ -,  $C$ - och  $D$ -matriserna innehåller partiella derivator. Dessa matriser kallas jacobianer,  $J$  och skrivs

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \quad \text{Man kan få en approximativ linjärisad beskrivning av ett sådant system ...} \quad (12)$$

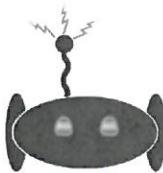
I ett praktiskt implementerbart system nyttjas till exempel sensorer som mäter signaler med en viss osäkerhet på mätdata för att bestämma systemets tillstånd. Man kan därmed enbart skatta systemets tillstånd och inte exakt beräkna dem. För att uppskatta dem används ofta en så kallad observatör.

<sup>7</sup>Glad, Torkel och Ljung, Lennart (2006), *Reglerteknik - Grundläggande teori*.

utan man får använda sig av en olinjär tillståndsmodell

$$x_{t+1} = f(x_t, u_t)$$

$$y_t = h(x_t, u_t)$$



### 3.1.1 Tillståndsåterkoppling och observatörer

Inom reglertechnik används en observatör för att ~~verifiera~~ skattade tillståndsvariabler i ett givet system ~~här tillstånden hos systemet ej kan bestämmas med säkerhet~~. Skattningen av variabler kan utföras genom tillståndsåterkoppling. Tillståndet för systemet ~~i ekvation (8)-(9)~~ kan skattas med hjälp av observatören ~~(8)-(9)~~

$$\dot{\hat{x}} = Ax + Bu + K(y - C\hat{x}) \quad (13)$$

Skattningsfelet  $\tilde{x} = x - \hat{x}$  ges ~~sedan~~ <sup>då</sup> av differentialekvationen ~~9~~

$$\dot{\tilde{x}} = (A - KC)\tilde{x} \quad (14)$$

Genom att sedan välja olika  $K$ -matriser ~~ges~~ <sup>för observatören</sup> systemet ~~har~~ <sup>har</sup> olika egenskaper i form av konvergenshastighet och störningskänslighet.

### 3.1.2 Kalmanfilter

Antag att systemet ~~i ekvation 9-11~~ störs av mätfelet ~~och~~ och systemstörningen som specificerades i problemformuleringen

$$\dot{x} = Ax + Bu + v \quad (15)$$

$$y = Cx + Du + e \quad (annars blir det fel nedan) \quad (16)$$

~~Skattnings-~~

Mätfelet ~~i ekvation 14~~ ges då istället av ~~v~~

$$\dot{\tilde{x}} = (A - KC)\tilde{x} + e - Kv \quad (17)$$

$R_1$  och  $R_2$

utgå från

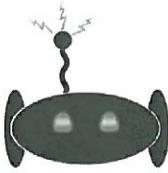
Då  $e$  och  $v$  antas vara vita brus kan man, genom att ~~uppskatta~~ <sup>beräkna</sup> störningarnas kovarians-matriser, beräkna mätfelets kovarians. Kovariansen hos mätfelet minimeras sedan genom att välja ~~skattning~~

$$K = PC^T R_2^{-1} \quad (18)$$

där matrisen  $P$  ges som den positivt semidefinita lösningen till ~~P~~

$$AP + PA^T + R_1 - PC^T R_2^{-1} CP = 0 \quad (19)$$

I många praktiska tillämpningar antas  $R_1$  och  $R_2$  vara diagonalmatriser för att förenkla implementeringar. Dessa matriser ~~viktas~~ sedan beroende på hur mycket systemet påverkas av yttre störningar. ~~av olika typ.~~ <sup>väljs</sup>



3.1.3

~~- variabelt  $K$ -~~ ~~där för tids-~~ ~~Det vanliga~~  
**Tidsdiskret och tidsinvariant Kalmanfilter** Inom SLAM är både hastighet och störningskänslighet essentiellt och man använder ~~där en variabel  $K$ -matris.~~ Kalmanfiltret är tyvärr inte effektivt vid SLAM, eftersom filtret endast kan användas för linjära system. Alla tillstånd kan dessutom inte skattas utefter samma förutsättningar som tidigare tillstånd. För att skatta nästkommande tillstånd med variabel  $K$ -matris används vanligen ett utökat Kalmanfilter vid implementeringar av SLAM.

För att beskriva det utökade Kalmanfiltret behöver vi först beskriva det tidsdiskreta och tidsvarianta Kalmanfiltret. Systemet beskrivs då som

$$x[n+1] = Ax[n] + Bu[n] \quad (20)$$

$$y[n] = Cx[n] + Du[n] \quad (21)$$

och skattningen ges av

$$\hat{x}[n+1|n] = A\hat{x}[n|n-1] + K[n](y[n] - C_n) \quad (22)$$

$$K[n] = AP[n|n-1]C^T([CP[n|n-1]C^T + R])^{-1} \quad (23)$$

$$P[n+1|n] = AP[n|n-1]A^T + R_1 - AP[n|n-1]C^T(CP[n|n-1]C^T + R)^{-1}CP[n|n-1]A^T \quad (24)$$

Använd Alg.

där  $R_1$  och  $R_2$  är kovariansmatriserna för respektive v.  $A$ -,  $B$ -,  $C$ - och  $D$ -matriserna varierar med tiden.

$\rightarrow$  ~~slim~~  $A_n$   $B_n$   $C_n$   $D_n$  mättestif  
*i sällskap*

### 3.1.3 EKF - Det utökade kalmanfiltret

~~det utökade kalmanfiltret fungerar för att approximera~~

*Det som möjliggör detta är att*

Till skillnad från Kalmanfiltret kan EKF lösa icke-linjära SLAM-problem. Detta då filtrets matriser beräknas om för varje nytt tillstånd. Matriserna för filtret beräknas med hjälp av jacobianer (se ekvation 3.1.3), och linjäriseras därmed kring varje tillstånd för att kunna skatta nästa tillstånd.

$R_1$  och  $R_2$ ? Initalt sätts kovariansen,  $R$ , till ett väldigt stort värde innan man vet vilken kovarians sina mätningar i miljön har. Därefter plockas mätdata in och man beräknar sin  $K$ -matris för att beräkna nästkommande tillstånd enligt:

$$K[n+1] = P[n]J[n]^T(J[n]P[n]J[n]^T + R[n])^{-1} \quad (25)$$

Där  $J$  är jacobianen som ges av partiella derivatorna hos mätdatat för avstånd till landmärken,  $y[n] = h(x[n]^r, m[n])$ .<sup>8</sup>

<sup>8</sup>David Törnqvist. Estimation and Detection with Applications to Navigation. PhD thesis, Linköping University, 2008, <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-14956>. Hämtad 15 april 2014.



### 3.2 FastSLAM

Förkortat/förslag på ett

FastSLAM är en relativt modern teknik som använder ett så kallat partikelfilter för att filtrera ~~givet~~ mätdata, där varje tillstånd för systemet ses som en partikel. De mest sannolika värdena sparar och de minst sannolika tillstånden förkastas. Man kallar det här för en omsampling ~~av data~~ då man utgår från en sampelmängd av tillstånd och sedan filtrerar denna till en än mindre mängd tillstånd. Antag att partikelfiltret plockar ut  $N$  tillstånd ur den totala mängden tillstånd. ~~så att~~ upptäcker dem när man ~~är~~ nya mätningar

$$x^i[n] \sim p(x[n]|y[1:n]), \quad i = 1, \dots, N \quad (26)$$

För att effektivisera filtreringen av mätvärden delas olika distinkt upptäckta objekt i robotens miljö in i olika zoner. Dessa zoner filtreras sedan individuellt för att få fram den mest sannolika positionen för objektet i zonen. Totala sannolikheten för tillståndet kan alltså delas upp enligt:

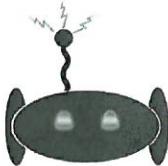
$$p(x[n], m) = p(x[n]|y[n])p(m|y[n]) \quad (27)$$

Där, som tidigare nämnts,  $x$  är robotens tillstånd,  $m$  är landmärkens position och  $y$  är mätningarna för givet tillstånd.

Tillstånden körs sedan flera gånger genom filtret tills dess att man med tillräckligt stor sannolikhet kan bestämma kartans utseende. Metoden har visat sig ~~väldigt~~ effektiv i praktiken. Nackdelen med metoden är att man genom omsampling av mätvärdena förlorar mätinformation som skulle kunna vara korrekt. Beroende på hur filtret prioriterar är det möjligt att man får en skev bild av omgivningen.

De matematiska grunderna för FastSLAM ligger utanför vad denna rapport täcker in - men för den intresserade läsaren rekommenderas David Törnqvists doktorsavhandling *Estimation and Detection with Applications to Navigation*<sup>9</sup>.

<sup>9</sup>David Törnqvist. Estimation and Detection with Applications to Navigation. PhD thesis, Linköping University, 2008. Hämtad 15 april 2014.



## 4 Utsats

Till slut kan vi konstatera att SLAM används i många tillämpningar man kanske inte tänker på. Man kan också konstatera att algoritmerna, filtren och mjukvaran som används för att implementera SLAM ibland kan vara väldigt komplexa. Algoritmerna som viktat vilka partiklar ska sparas respektive förkastas kan byggas ut i stort sett hur mycket som helst.

I ett SLAM-problem likt det i rapportens problemställning skulle positionen kunna skattas med hjälp av sensorvärdet i form av medelvärdet för vinkel- och avståndssensorer. Eftersom området är slutet kan det uppskattas klart mycket enklare än ett godtyckligt stort område.

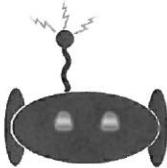
ECP? En sådan lösning skulle kunna baseras på en tillståndsrepresentation med ett tidsvariant, tidsdiskret Kalmanfilter. Genom att medelvärdesbilda sensordata kan man antagligen försumma mätstörningar och ändå få en acceptabel skattning av omgivningen utan partikelfilter. Reglerfel kan i sin tur sedan minimeras med bra modellering av hårdvaran implementeringen nyttjar.

## 5 Reflektion

Sett från det perspektiv vi granskar SLAM-problemet, genom ögonen på en grupp mänskor som gör ett kandidatprojekt, är avancerade FastSLAM-algoritmer för stor att implementera. Det måste dock alltid finnas någon typ av filter i en SLAM-implementering. Då beräkningar man gjort tidigare under kartläggningen ger konflikt med nuvarande mätning om huruvida en vägg faktiskt finns där eller ej. Enklare fall av partikelfiltrering kommer alltså behöva lösas vid vår framtida implementering. Vi kommer då att lösa det genom att låta robotten besöka området på nytt och ta ett beslut utifrån de nya mätdata vi får in.

Slutligen kan man säga att denna djupdykning inom SLAM givit oss bra förutsättningar att skapa en implementerbar modell för hur vår robot ska kartlägga sitt slutna område. Särskilt sektionen om Kalmanfilter och tillståndsrepresentering känns relevant vid en enklare implementering.

ge igenom denna text så att den blir begriplig, logiken hänger inte ihop nu



## Källförteckning

Alla referenser måste nämnas i texen!  
Hur har ni sorterat referenserna? (Vid bokstavordning eller nummerordning)

(nämnd) Weinberg, Harvey (1999), "Dual Axis, Low g, Fully Integrated Accelerometers", <http://www.analog.com/library/analogdialogue/archives/33-01/accel/index.html>. Hämtad 2014-05-12.

(nämnd) David Törnqvist. Estimation and Detection with Applications to Navigation. PhD thesis, Linköping University, 2008, <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-14956>. Hämtad 15 april 2014.

(nämnd) FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, Stanford University. Hämtad 28 mars 2014. [http://robots.stanford.edu/papers/montemerlo\\_fastslam-tr.pdf](http://robots.stanford.edu/papers/montemerlo_fastslam-tr.pdf)

(nämnd) Fox, C.; Evans, M.; Pearson, M.; Prescott, T. (2012) "Tactile SLAM with a biomimetic whiskered robot". 2012 IEEE International Conference on Robotics and Automation. Hämtad 28 mars 2014. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6224813>

(nämnd) Glad, Torkel och Ljung, Lennart. 2006. *Reglerteknik - Grundläggande teori*. Upplaga 4:10. Lund. Studentlitteratur.

(nämnd) Gustavsson, Fredrik; Ljung, Lennart; Millnert, Mille. 2000. *Signalbehandling*. Andra upplagan. Lund. Studentlitteratur.

(nämnd) Kandidatprojekt Y: Elektronikprojekt, Tävlingsregler för katläggningsrobot. Hämtad 28 mars 2014. <https://drive.google.com/file/d/0B758zzcy4ZrTeG1wRTY4WG91TDQ/edit?usp=sharing>

(nämnd) Karlsson, N.; Goncalves, L.; Munich, M.E.; Pirjanian, P. "The vSLAM Algorithm for Navigation in Natural Environments". Evolution Robotics, Inc. Hämtad 28 mars 2014: <http://www.vision.caltech.edu/mariomu/research/papers/vSLAM-krs.pdf>

(nämnd) Openslam.org <http://www.openslam.org/>

(nämnd) Risgaard, S; Blas, M.R (2005). "SLAM for Dummies, A Tutorial Approach to Simultaneous Localization and Mapping". Hämtad 28 mars 2014: [http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam\\_blas\\_repo.pdf](http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf)

(nämnd) Smith, R.C.; Cheeseman, P. (1986). "On the Representation and Estimation of Spatial Uncertainty". The International Journal of Robotics Research, 5(4), sida 56–68. Hämtad 28 mars 2014: <http://www.frc.ri.cmu.edu/~hpm/project.archive/reference/file/Smith&Cheeseman.pdf>

(nämnd) Piezoelectricity, Wikipedia. <http://en.wikipedia.org/wiki/Piezoelectricity>. Hämtad 2014-05-12.



Söka på att ha med ekvationer i stl med dessa:

### Tidsdiskr. tidsvar. kalmanfilter

$$1) \hat{x}_{t|t-1} = A_{t-1} \hat{x}_{t-1|t-1} + B_{t-1} u_{t-1}$$

$$2) P_{t|t-1} = A_{t-1} P_{t-1|t-1} A_{t-1}^T + R_1$$

$$3) K_t = P_{t|t-1} C_t^T (C_t P_{t|t-1} C_t^T + R_2)^{-1}$$

$$4) \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t (y_t - C_t \hat{x}_{t|t-1})$$

$$5) P_{t|t} = (I - K_t C_t) P_{t|t-1}$$

### EKF

$$\text{Modell: } x_t = f(x_{t-1}, u_{t-1}) + v_{1,t-1}$$

$$y_t = h(x_t, u_t) + v_{2,t-1}$$

$$\text{EKF: 1)} \hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1}, u_{t-1})$$

$$2) P_{t|t-1} = F_{t-1} P_{t-1|t-1} F_{t-1}^T + R_1, \quad \text{då } F_t = \frac{\partial f(x, u)}{\partial x} \Big|_{(x, u) = (\hat{x}_{t-1|t-1}, u_{t-1})}$$

$$3) K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_2)^{-1} \quad \text{då } H_t = \frac{\partial h(x, u)}{\partial x} \Big|_{(x, u) = (\hat{x}_{t|t-1}, u_t)}$$

$$4) \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t (y_t - h(\hat{x}_{t|t-1}, u_t))$$

$$5) P_{t|t} = (I - K_t H_t) P_{t|t-1}$$

- Notera likheten mellan de båda algoritmens steg

- OBS: Ni måste ha någon annan källa än det har peppat..

