Abschlussaufgabe V - Partikelfilter in ARNL

Nils Petersohn, Matrikel: 20022749

3. Feb. 2010

1 Aufgabenstellung

Es soll die monte-carlo-lokalisierung in Arnl untersucht werden. Durch Experimente und Recherche soll der genaue Ablauf des Partikelfilters und die Wirkung der Parameter dargestellt werden. Ausserdem soll geklaert werden wie die lokalisierung in eigenen programmen genutzt werden kann.

2 Einführung

2.1 ARIA

Die C++ Bibliothek ARIA ist eine API fuer alle MobileRobots/ActivMedia Plattformen. Mit der Hilfe von ARIA kann u.a. Geschwindigkeit, Richtung oder relative Richtung entweder durch einfache oder komplexe Anweisungen kontrolliert werden.

2.2 ArNetworking

ArNetworking ist eine API in ARIA um eine client-server Architektur zwischen dem Robotter(Server) und dem entfernten Rechner(Client) ueber das Netzwerk aufzubauen. Der Client berechnet Bewegungsbefehle und sendet Anfragen an den Server welche dann vom Robotter ausgefuehrt werden. Der Robotter hingegen sendet u.a. seine Odometriedaten.

2.3 MobileEyes

MobileEyes kann sich mit ARIA, ArNetworking und ARNL Server verbinden und dessen Umgebung also interne Karte anzeigen. Unter anderem kann man mit MobileEyes Bewegungskomandos an den Robotter schicken.[Whitbrook1]

2.4 ARNL

- 4 Module:
- BaseArnl: Infrastukture, Pfadplanung, Pfadfolgen
- Lokalisierung mit Sonar (SonArnl), Laserscanner (Arnl) und GPS (Mogs)

ARNL(Advanced Robotics Navigation and Localization System) ist eine Teilbibliothek in ARIA um Navigations und Lokalisationsschnittstellen zur Verfügung zu stellen. Die Lokalisationskomponente bietet eine Schnittstelle um den angeschlossenen Roboter zu lokalisieren. Sensor und Odometriedaten des Roboters werden in Verbindung mit einer vorhanden Umgebungskarte verwendet um die wahrscheinlichste Position des Roboters in der Karte zu bestimmen. [Wiki-ARNL] Die ARNL Bibliothek beinhaltet die Klasse ArLocalizationTask. Diese Klasse kann Daten vom Laserscanner (ArSick) auswerten um die Lokalisierung durchzufuehren. Abgesehen vom ArLocalizationTask gibt es noch andere Klassen fuer die Lokalisierung. Die Elternklasse ist ArBaseLocalizationTask in der BaseARNL Bibliothek. [Wiki-ARNL] Um den ArLocalizationTask zu instanzieren müssen Pointer von Instanzen der Typen ArRobot, ArRangeDevice und ArMapInterface (ArMap) uerbgeben werden.

```
#include "ArLocalizationTask.h"
...
ArRobot robot;
...
ArLaser *firstLaser = robot.findLaser(1);
...
ArMap map(fileDir);
...
ArLocalizationTask locTask(&robot, firstLaser, &map);
```

Um eine initiale Lokalierung durchzufuehren muss die Methode localizeRobotAtHomeBlocking ohne Parameter aufgerufen werden.

3 Theoretischer Hintergrund

Das Lokalisierungsmodul von ARNL benutzt die Monte-Carlo-Lokalisierung. Diese basiert auf der Markov-Annahme welche wiederum auf die Bayes-Regel fundiert.

3.1 Probabilistische Lokalisierung

Um die Position eines Roboters zu berechnen, gibt es verschiedene Anstze. Sie beruhen alle auf der Fusionierung von Daten, die durch die Odometrie und weitere Sensoren des Roboters geliefert werden. Gebruchliche Sensoren fr autonome Roboter sind Ultraschallsensoren oder Laserscanner. Diese Anstze verwenden Methoden der Wahrscheinlichkeitsrechnung zur Bestimmung eine Wahrscheinlichkeitsverteilung ber alle mglichen Positionen, an denen der Roboter sich befinden kann.

in ARIA umgesetzt in ARNL

leistungsfhiger, verbreiteter Ansatz vereint relative und absolute Postionierung als Bewegungsmodellierung und Wahrnehmung der Umwelt

3.2 Beyes Filter

3.3 Markov-Lokalisierung

Das Ziel der Markov Lokalisierung ist es, jeder mglichen Roboterposition einen Wahrscheinlichkeitswert zuzuordnen.

3.4 Partikelfilter

Bei der Lsung mittels Partikel-Filern wird die Pose des Roboters ber eine Partikelwolke reprsentiert. Jeder Partikel stellt eine mgliche Pose des Roboters dar. ber den Partikelfilter wird jeder Partikel, also jede dadurch reprsentierte Pose, auf ihre Plausibilitt berprft. Die Wahrscheinlichkeit plausibler Partikel wird heraufgesetzt, die Wahrscheinlichkeit wenig plausibler Partikel wird reduziert. Fallen Partikel unter einen bestimmten Wahrscheinlichkeits-Schwellwert, werden sie verworfen.

3.5 Monte-Carlo-Lokalisierung

Der Vorteil des MCL - Ansatzes ist, dass zur Laufzeit die Grösse der Stichprobenmenge variabel sein kann. Je unsicherer die Roboterposition ist, desto grer fllt die Stichprobenmenge aus. In Analogie zur Gitterbasierenden (grid-based) Methode, müsste bei einer hohen Sicherheit (belief) nur ein Teil des Zustandsraumes aktualisiert werden.[Delipetkos1] Der Grundgedanke bei der Monte Carlo Lokalisierung ist, das Belief (Vertrauen) mittels einer Stichprobenmenge, welche auch als samples oder auch particles bezeichnet wird, darzustellen. MCL ist ein iterativer Bayesscher Filter, welcher als ein Schtzer für die zukünftige Wahrscheinlichkeitsverteilung der Roboterposition verwendet wird. Monte-Carlo-Lokalisierung

- MCL erlaubt elegante Fusion der Daten verschiedener Sensorik
- MCL skalierbar (Partikelanzahl)
- MCL effizient, bspw. dynamische Anpassung der Partikelanzahl (adaptiv)
- Particle filters are an implementation of recursive Bayesian filtering
- They represent the posterior by a set of weighted samples.
- In the context of localization, the particles are propagated according to the motion model.
- They are then weighted according to the likelihood of the observations.
- In a re-sampling step, new particles are drawn with a probability proportional to the likelihood of the observation. Monte Carlo Localization = Markov-Lokalisierung mit Abbildung der
- Wahrscheinlichkeit als Samplemenge (Partikelwolke)
- Sample = eine Position (x,y,)
- Prognose-Schritt: jedes Sample erzeugt ein neues Sample entsprechend unscharfem Bewegungsmodell -¿ neue Samplemenge bel
- Korekktur-Schritt:
- Jedes Partikel aus bel erhlt ein Gewicht nach dem Sensormodell (wie eine Fitness, bereinstimmung mit Wahrnehmung)
- neue Samplemenge bel wird durch zufllige Selektion anhand der Fitness aus bel erzeugt
- stndige Evolution einer Partikelwolke anhand von Bewegungsbefehlen und Sensorwahrnehmung,

4 Experimente

In einem Beispielprogramm wird ARNL und ArNetworking verwendet um einen Server zu erstellen. Clients wie MobileEyes koennen sich mit diesem Server verbinden. MobileEyes sendet anfragen an den Server um den Robotter zu steuern, die aktuelle Karte abzufragen, Lokalisierungsparameter zu setzen, Sensordaten abzufragen und den Robotter manuell wieder neu zu lokalisieren. Die automatische lokalisierung des Robotters erfolgt durch eine Instanz von ArLocalizationTask welcher also Teil des Serverprograms ist. Wie in 2 deutlich gemacht wurde, ist das Monte-Carlo Verfahren in der Klasse ArLocalizationTask implementiert. Durch die Verwendung von MobileEyes kann eine Anpassung der Lokalisierungsparameter Einfluss auf den Partikelfilter genommen werden. Die Einstellungen befinden sich in MobileEyes unter Lokalisierungseinstellungen. Eine liste und deren Beschreibung befinden sich in Appendix I. Der Monte Carlo Filter verlaesst sich unter anderem auf die Odometriedaten die von den Raedern kommen. Dabei kann bestimmt werden wie weit der Robotter gefahren ist oder wenn sich die Raeder unterschiedlich bewegen, wie weit sich der Robotter um seine Achse gedreht hat. wenn man die genaue Startposition des Robotters kennt und auch die Odometriedaten der Raeder, so kann man theoretisch die genaue Position des Robotters zum Zeitpunk t bestimmen. In der Praxis ist dies aber nur schwer zu realisieren. Die Unsicherheit wchst also mit zunehmender Entfernung vom Startpunkt. generell gibt es drei Odometrie Fehlerarten:

- Entfernungsfehler: Fehler beim Zurücklegen einer geraden Strecke
- Drehfehler: Fehler beim Drehen des Roboters
- Driftfehler: Orientierungsfehler beim Zurücklegen einer geraden Strecke

Das Bewegungsmodell in ARIA/ARNL bildet genau diese Fehler ab. Um die Bewegungsunschaerfe zu testen muessen verschiedene Initialalparameter verwendet werden:

- PassThreshold muss auf 0 gesetzt werden um zu vermeiden, dass der Roboter in den Lost Modus geht
- Um die Partikelwolke besser sichtbar zu machen wird der Wert NumSamples auf 5000 gesetzt
- PeturbX, PeturbY, PeturbTh sind Paramter um neue Samples samples nach dem resampling prozess hizuzufügen. Diese Streuwerte werden auf null gesetzt um den Rechenanspruch zu verrringern.
- per default wird die Neuberechnung der Position erst nach einer Bewegung von 20 cm gemacht. Eine Neuberechnung soll erst nach 2m stattfinden. Somit wird der Paramter Triggerdistance auf 2000 gesetzt
- die neuberechnung nach einer drehnung soll nicht erst ab 5 Grad erfolgen sondern schon ab 0 Grad. Der Paramter **TriggerAngle** wird also auf 0 gesetzt.

Um den Entfernungsfehlerverhaeltnissparameter zu testen wird der Drehfehlerparameter und der Driftfehler zuerst ausgeschaltet. Um den Parameter zu untersuchen wird er auf 0.5 gesetzt. Dies soll bewirken, dass die Partikelwolke sich um die Hälfte der zurueckgelegten Strecke ausdehnt. Der Roboter wird zuerst an einem Punkt lokalisiert. Mit einem Bewegungsbefehl wird der Robotter nun 1 Meter gerade aus bewegt. Wie angenommen hat die Partikelwolke nun eine Länge von 50 cm.

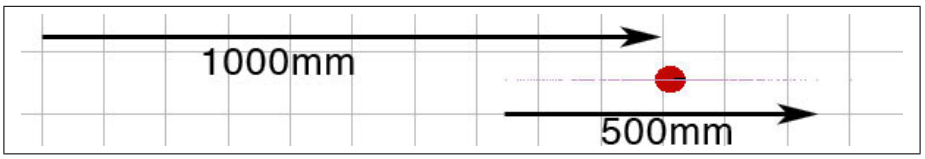
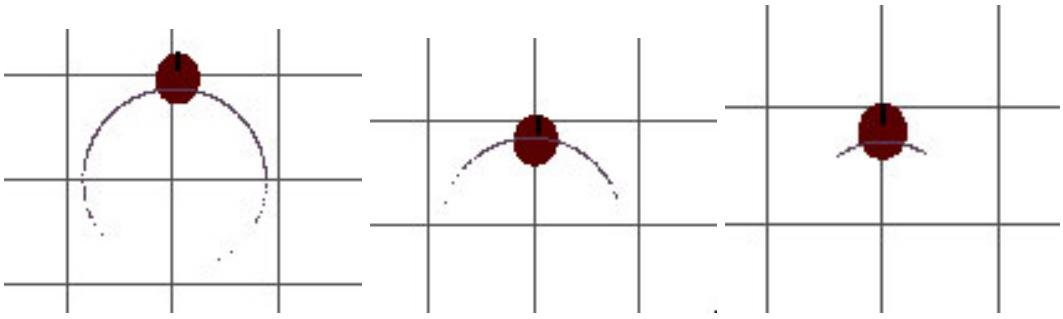


Abbildung 1: KMmPerMm = 0.5

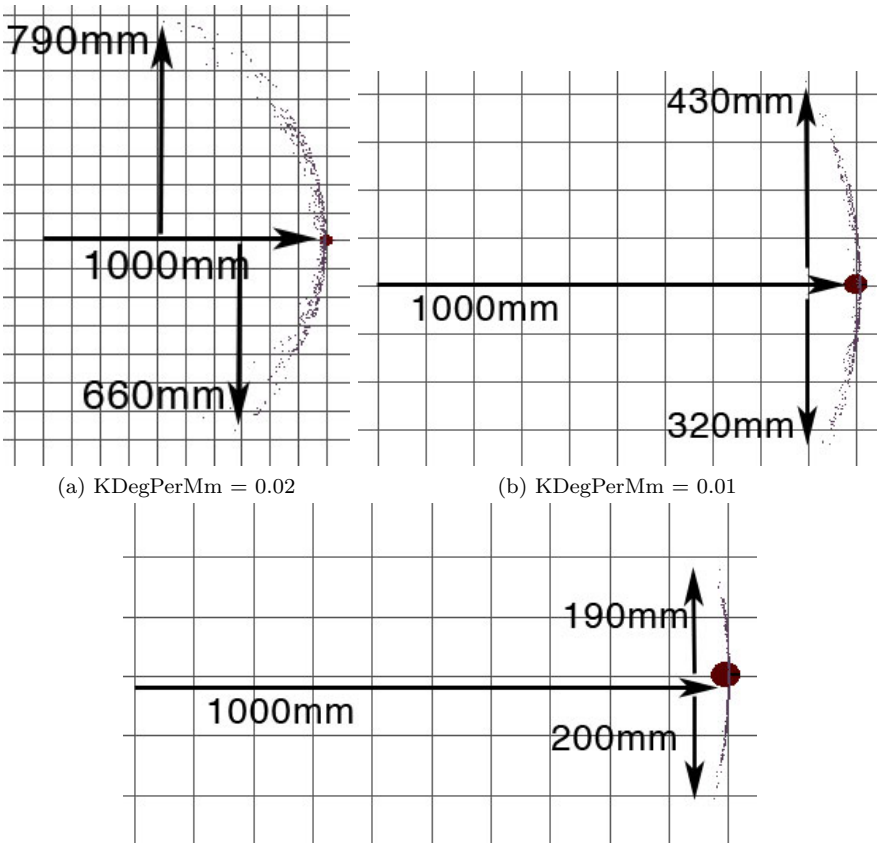


(a) KDegPerDeg = 1 (b) KDegPerDeg = 0.5 (c) KDegPerDeg = 0.25

Abbildung 2: KDegPerDeg Parameter Test

Dieser Parameter kann dazu verwendet werden um die Lokalisation in bezug auf die Oberflaechenbeschaffenheit anzupassen. In einem hügligem Gelände kann der Roboter eine Steigung hochfahren sich aber im eigentlichen Sinne nicht so schnell weiterbewegen wie die Odometriedaten es uebermitteln. Desshalb muss dieser Parameter erhöht werden.

Mit dem Drehfehlerparameter kann nach der Dokumentation in Anhang A die Unsicherheit der Drehung beeinflusst werden. Wenn der Robotter sich z.B. auf einem leicht unebenen Untergrund dreht ist die Drehung nicht eindeutig an den Odometriedaten nachvollziehbar. Um diesen zu testen wird der Parameter zuerst auf 1 gesetzt und Entfernungsfehlerverhaeltnissparameter wieder auf null. In Abbildung 2a wurde der Roboter um 360 Grad gedreht und dann 1cm gerade aus gefahren. Der Parameter wurde in Abbildung 2b und 2c stufenweise halbiert um den Winkel der Partikelstreuung zu erkennen. Dieser Entspricht demnach dem Verhaeltnis zwischen der Drehung in Grad und der KDegPerDeg Einstellung. Also bei einer Drehnung von 360 Grad und einer einstellung von 1 hat der Partikelfilter eine breite von auch ca. 360 Grad bei 0.5 von ca. 180 Grad und bei 0.25 von ca. 90 Grad. Wenn also der Untergrund hüglig ist, sollte die Unsicherheit also der KDegPerDeg Paramter hoeher eingestellt sein.



(a) KDegPerMm = 0.02

(b) KDegPerMm = 0.01

(c) KDegPerMm = 0.005

Abbildung 3: KDegPerMm Parameter Test

Der Parameter KMmPerMm betrifft die unsicherheit der Odometriedaten der Raeder. Bei einem differenziellem Antrieb koennen leichte unterschiede auftreten, wenn der Robotter sich gerade aus bewegt. Wenn der Robotter sich z.B. auf einem unebenen Boden fortbewegt, kann er nach links und rechts abdriften. Die Odometriedaten der Raeder geben aber ein stetiges geradeausfahren zurueck.

Durch eine einfache Winkelfunktionsrechnung in Abbildung 4 laesst sich der Winkel bestimmen. Zuerst wird der mittelwert der Abweichung durch das Arithmetische Mittel bestimmt siehe Formel 1

Um die Rechnungen zu bestaetigen Rechnen wir den Wert aus welcher zu einem Fehlerwinkel von 180° noetig ist und bilden dann davon das Arithmetische Mittel. Wenn dieses nicht sonderlich abweicht waren die Rechnungen richtig. Dies ist der Fall denn eine maximale Abweichung von weniger als 0.012 und eine Minimale Abweichung von 0.0009 ist anehmbar.

Die Bestaetigung liefert eine Einstellung von KDegPerMm = 0.04 welche eine 180° ausgedehnte Partikelwolke aufspannt siehe Abbildung 9.

Somit laesst sich der Parameter nachvollziehen. Wenn man einen Winkel α erreichen will muss man Rechnen: $KDegPerMm = \alpha \cdot 2.22 \cdot 10^{-4}$ Diese Rechnung ist nicht genau und wurde mit

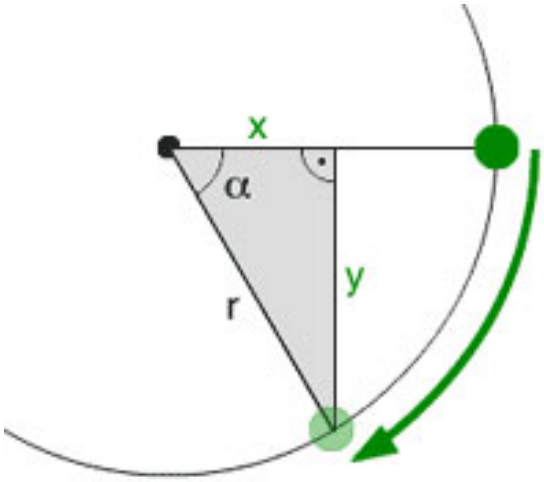


Abbildung 4: Winkelfunktionen im Einheitskreis
`http://www.ullala.at/experiments/movement/images/circ_1.gif`

$$\bar{y} = \frac{790 + 660}{2} = 725mm \tag{1}$$

$$\theta_1 = \sin^{-1}\left(\frac{y}{r}\right) \cdot 2 \tag{2}$$

$$\theta_1 = \sin^{-1}\left(\frac{725}{1000}\right) \cdot 2 \approx 93^\circ \tag{3}$$

Abbildung 5: Berechnung fuer Abbildung 3a

$$\bar{y} = \frac{430 + 320}{2} = 375mm \tag{4}$$

$$\theta_2 = \sin^{-1}\left(\frac{y}{r}\right) \cdot 2 \tag{5}$$

$$\theta_2 = \sin^{-1}\left(\frac{375}{1000}\right) \cdot 2 \approx 44^\circ \tag{6}$$

Abbildung 6: Berechnung fuer Abbildung 3b

$$\bar{y} = \frac{190 + 200}{2} = 195mm \tag{7}$$

$$\theta_3 = \sin^{-1}\left(\frac{y}{r}\right) \cdot 2 \tag{8}$$

$$\theta_3 = \sin^{-1}\left(\frac{195}{1000}\right) \cdot 2 \approx 23^\circ \tag{9}$$

Abbildung 7: Berechnung fuer Abbildung 3c

$$\frac{\theta}{KDegPerMm} = \frac{180^\circ}{x} \tag{10}$$

$$x_0 = \frac{180^\circ \cdot 0.005}{23} = 0.0391 \tag{11}$$

$$x_1 = \frac{180^\circ \cdot 0.01}{44} = 0.0409 \tag{12}$$

$$x_2 = \frac{180^\circ \cdot 0.02}{93} = 0.0387 \tag{13}$$

$$\bar{x} = \frac{x_0 + x_1 + x_2}{3} \approx 0.04 \tag{14}$$

Abbildung 8: Test der Berechnungen in Abbildung 3,6,9

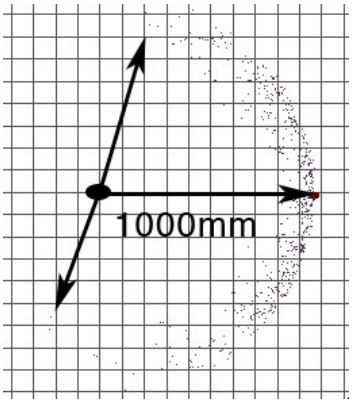


Abbildung 9: KDegPerMm = 0.04

Rundungswerten ermittelt.

A Appendix I

Map Karte der Umgebung um zu Navigieren.

NumSamples 2000 minumum 0, No of pose samples for MCL. The larger this number, the more computation will localization take. Too low a number will cause the robot to lose localization. This is also the maximum no of samples which will be used for localization if no of samples are varied along with the localization score.

GridRes 100 minimum 10, The resolution of the occupancy grid representing the map in mm. Smaller resolution results in more accuracy but more computation.

PassThreshold 0.2 range [0, 1], After MCL sensor correction, the sample with the maximum probablity will have a score based on the match between sensor and the map points. This is the minimum score out of 1.0 to be considered localized.

KMmPerMm 0.05 minimum 0, When the robot moves linearly, the error in distance is proportional to the distance moved. This error is is given as a fraction in mm per mm

KDegPerDeg 0.05 minimum 0, When the robot rotates, the error in the angle is proportional to the angle turned. This is expressed as a fraction in degs per deg.

KDegPerMm 0.0025 minimum 0, When the robot moves linearly it can also affect its orientation. This drift can be expressed as a fraction in degs per mm.

TriggerDistance 200 minimum 0, Since MCL localization is computationally expensive, it is triggered only when the robot has moved this far in mm.

TriggerAngle 5 minimum 0, Since MCL localization is computationally expensive, it is triggered only when the robot has rotated this far in degs.

TriggerTimeEnabled false This flag will decide if the localization will be called every 'TriggerTime' msecs. Once this flag is true the IdleTimeTrigger* parameters will take effect. This feature is meant to take care of cases when the robot has not moved much for a time and the position should be refined .

TriggerTime 10000 minimum 1500, Once the TriggerTimeFlag is set to true this parameter will decide how long the robot has been idle in milli seconds before it starts a localization near the last known robot pose.

IdleTimeTriggerX 200 minimum 0, When localization is triggered by idle time this parameter decides the range of the samples in X coords in mm.

IdleTimeTriggerY 200 minimum 0, When localization is triggered by idle time this parameter decides the range of the samples in Y coords in mm.

IdleTimeTriggerTh 15 minimum 0, When localization is triggered by idle time this parameter decides the range of the samples in Theta coords in degs.

RecoverOnFail false If localization fails, this flag will decide if a static localization is attempted around last known robot pose. Such a reinitialization can cause the robot to be hopelessly lost if the actual robot is very different from its known pose

FailedX 300 minimum 0, Range of the box in the X axis in mm to distribute samples after localization fails.

FailedY 300 minimum 0, Range of the box in the Y axis in mm to distribute samples after localization fails.

FailedTh 45 minimum 0, Range of the angle in degs to distribute samples after localization fails.

PeturbX 10 minimum 0, After sensor correction and resampling the chosen pose is perturbed to generate a new sample. This parameter decides the range to peturb the X axis in mm.

PeturbY 10 minimum 0, After sensor correction and resampling the chosen pose is perturbed to generate a new sample. This parameter decides the range to peturb the Y axis in mm.

PeturbTh 1 minimum 0, After sensor correction and resampling the chosen pose is perturbed to generate a new sample. This parameter decides the range to peturb the angle in degs.

PeakStdX 10 minimum 0, Extent of the ellipse in the X axis in mm beyond which the sample poses will be considered multiple localizations after resampling.

PeakStdY 10 minimum 0, Extent of the ellipse in the X axis in mm beyond which the sample poses will be considered multiple localizations after resampling.

PeakStdTh 1 minimum 0, Extent of the angle in degs beyond which the sample poses will be considered multiple localizations after resampling.

PeakFactor 1e-06 range [0, 1], When a no of samples have non zero probabilities such as when there is ambiguities in a corridor. This is the threshold below the maximum probablity to be considered a valid hypothesis.

StdX 400 minimum 0, The standard deviation of the gaussian ellipse in X axis in mm at start of localization.

StdY 400 minimum 0, The standard deviation of the gaussian ellipse in Y axis in mm at start of localization.

StdTh 30 minimum 0, The standard deviation of the gaussian angle in degs at start of localization.

SensorBelief 0.9 range [0, 1], Probablility that a range reading from the laser is valid. This is used in the correction of the probablities of the samples using the sensor.

OccThreshold 0.1 range [0, 1], The threshold value of the occupancy grid to consider as occupied.

AngleIncrement 0 range [0, 180], Only the laser readings which are this many degrees apart are used for the localization. The lower limit is decided by the LaserIncrement setting

DiscardThreshold 0.33 range [0.33, 1], A robot sample pose lying inside an occupancy grid cell with a value above this will be usually discarded Useful in cases where robot may intersect map points such as during patrolbot docking

Literatur

[Wiki-ARNL] MobileRobots Inc, *ARNL, SONARNL and MOGS*, wiki (9 September 2009), available at `http://robots.mobilerobots.com/wiki/ARNL,_SONARNL_and_MOGS`.

[Delipetkos1] Fraunhofer Gesellschaft AIS.ARC, Particle Filter Ein probabilistischer Ansatz zur Lokalisierung mobiler Roboter available at S. 12 `http://www.ais.fraunhofer.de/~delipetk`

[ARNL-Ref] MobileRobots Inc, *ARNL-Reference*, (1.7.0) available at `http://vigir.missouri.edu/~gdesouza/Research/MobileRobotics/Software/ARNL-SONARNL/Arnl-1.7.0+gcc41/docs/ARNL-Reference/index.html`.

[Whitbrook1] Dr. Amanda Whitbrook. *Programming Mobile Robots with Aria and Player*. Springer-Verlag London Limited , 2010, S. 50 `http://www.ais.fraunhofer.de/~delipetk`

Abbildungsverzeichnis