IMPLETACIÓN DE UN ALGORITMO DE LOCALIZACIÓN BASADO EN UN MÉTODO DE MONTECARLO PARA UN ROBOT MOVIL OMIDIRECCCIONAL

J. Gómez-Moreno, A. Soriano, M. Vallés, A. Valera, M. Martínez
{jagmemo, ansovi, mvalles, giuprog}@ai2.upv.es, mimarma5@upvnet.upv.es
Instituto de Automática e Informática Industrial
Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

Resumen

Una de las áreas de la robótica móvil en la que se está dedicando mucho trabajo de investigación en los últimos años está relacionada con la localización de los robots. Para el trabajo en ambientes exteriores existe una amplia gama de sensores GPS que permiten determinar de forma absoluta y cada vez con mayor precisión, la ubicación de un elemento. Sin embargo, cuando la misión a realizar por el robot tiene lugar en interiores, resolver el problema de la localización absoluta con una cierta exactitud se convierte en algo crítico si se quiere dotar al robot de una cierta autonomía y precisión en su movimiento.

Para resolver el problema de la localización se pueden encontrar dos soluciones. La primera se basa en el estudio y desarrollo de algoritmos que tratan de localizar un elemento en un mapa a la vez que tratan de construir el propio mapa del entorno, dando lugar a los conocidos algoritmos SLAM (Simultaneous Location and Mapping [13]). La segunda solución se basa en el refinamiento de la localización de un elemento en un mapa conocido.

El presente artículo propone una implementación de un método de Montecarlo [2] para la localización robusta en un mapa conocido mediante la integración de un sensor laser de pequeño tamaño en una plataforma móvil robotizada [3].

Palabras Clave: Robots móviles, localización en interiores, algoritmos probabilistas, método de Montecarlo, sensorización de robots.

1 INTRODUCCIÓN

El problema de la localización en la robótica móvil se puede subdividir en dos: la estimación de la posición absoluta global del robot y la capacidad del robot de hacer un seguimiento de su posición de forma local. La posición absoluta, que es el caso que se aborda en este artículo, se define como la ubicación del robot dentro de un espacio acotado y conocido, sin más información disponible a priori para el robot, que el conocimiento de que se encuentra dentro de dicho mapa. Si se considera que el robot se encuentra localizado dentro del mapa, la segunda parte del problema, consiste en poder realizar de forma local un seguimiento de su posición. La subdivisión del problema en dos, permite utilizar el posicionamiento global para que, haciendo uso del mapa disponible, sea posible planificar una serie de movimientos que el robot debe realizar para cumplir con su misión, mientras que el seguimiento local de su posición le permite realizar una navegación eficiente. Esto, le facilita el poder tener en cuenta imprevistos no reflejados en el mapa, como por ejemplo cualquier tipo de obstáculo móvil.

Para solucionar el problema de estimar la posición absoluta global, se estima que los métodos probabilísticos tienen mucho que ofrecer en cuanto a robustez en la navegación, [1] [7], ya que permiten una representación constante y en tiempo de real de la incertidumbre de la posición global. Esto último posibilita también la definición de estrategias inteligentes que permitan reducir esta incertidumbre una vez se hayan evitado los problemas que la originaron en un principio (como el encuentro inesperado con un obstáculo móvil dentro del mapa, por ejemplo).

Otros aspectos clave de los enfoques probabilísticos es que proporcionan una localización global más precisa que con modelos de localización basados en la subdivisión del mapa en celdas fijas, permitiendo además una fácil implementación. Sin embargo, esto no está exento de contrapartidas. En el caso del método Montecarlo expuesto en este trabajo, éste puede dar lugar a falsas localizaciones, sobre todo en mapas simétricos, donde es imposible diferenciar la posición del robot exclusivamente con información geométrica. También, por norma general, el método probabilístico aquí descrito requiere de una mayor potencia computacional para la realización de los múltiples cálculos necesarios para localizar al robot.

Sin embargo, existen formas de solucionar ambos problemas. Los falsos positivos pueden solucionarse con estrategias de desambiguación, como fusionar el algoritmo con técnicas de localización "cricket" [4] [6], para diferenciar posiciones o incluir más información en el mapa y sensores extra para captarla. Por otra parte, la necesidad de potencia de cálculo es un problema menguante puesto que hoy en día es posible encontrar sistemas empotrados de tamaño reducido con una alta capacidad de cómputo o disponer de un PC remoto al que delegar los cálculos mediante cualquier tipo de comunicación inalámbrica. Esto, unido a que la aproximación del problema que aquí se expone permite la definición del tamaño del mismo [8], así como la posible paralelización de los cálculos necesarios, permite afirmar que el método presentado es capaz de resolver el problema de la localización de forma óptima.

La organización del artículo es la siguiente: en la próxima sección se presenta la plataforma de trabajo empleada y las modificaciones que han sido necesarias para sensorizarla, así como una breve descripción de la arquitectura del sistema implementado. En la sección 3 se expone el algoritmo de localización, haciendo hincapié en el tipo de representación empleado para la incertidumbre de la posición y el cálculo de la misma. Finalmente, en las secciones 4 y 5 se expondrán las pruebas realizadas y, los resultados y conclusiones derivados de la implementación.

2 CONSTRUCCIÓN DE LA PLATAFORMA DE TRABAJO

2.1 ROBOTINO

Robotino [5] es un robot móvil desarrollado por FESTO didactic dotado de un sistema de 3 ruedas omnidireccionales montadas a 120º que le permite desplazarse en cualquier dirección y rotar sobre sí mismo al mismo tiempo. La Figura 1 muestra dicho robot.

Cada una de las unidades que permiten el movimiento consiste en un motor de corriente continua unido a una reductora de ratio 16:1, una rueda omnidireccional, una correa dentada y un encoder incremental.

El robot está equipado con un sensor de colisión y un anillo de 9 sensores infrarrojos para la detección de distancias. Además de estos sensores, el robot móvil utilizado en este trabajo integra también un giróscopo extra suministrado por el fabricante que incrementa notablemente la precisión de la odometría.

Figura 1. Robotino de FESTO didactic

Robotino está controlado por un PC empotrado que cuenta con una versión de RT Linux (Robotino OS) como sistema operativo. Para acceder al terminal del sistema operativo instalado se puede abrir un terminal de dicho sistema operativo a través de la red inalámbrica haciendo uso de un cliente de Secure Shell (SSH).

Para la programación y control del Robotino se disponen de varias opciones. Una de ellas es programarlo con el software *Robotino View* en un PC a través de una red de área local inalámbrica. *Robotino View* es un entorno de desarrollo gráfico que es capaz de transmitir señales al controlador del motor, así como visualizar, cambiar y evaluar valores de los sensores. Además, Festo pone a disposición diferentes APIs que permiten programar con otras herramientas como Matlab y Labview.

En el caso de que no se desee depender de un PC externo para establecer el control (como es el caso expuesto en este artículo para la parte de seguimiento de posición de forma local) se puede hacer uso de un juego de APIs proporcionadas por Festo para programar, compilar y ejecutar aplicaciones, que pueden estar escritas tanto en C++, Java o .Net, directamente en el PC empotrado del robot.

2.2 SENSOR HOKUYO URG-04LX-UG01

HOKUYO URG-04LX-UG01 [10] es uno de los range-finder láser mas pequeños disponibles en el mercado (ver Figura 2). Su bajo peso de 160 gramos y consumo de 2.5 vatios hacen que sea una solución sumamente interesante para la sensorización de cualquier robot móvil autónomo.

El URG-04LX-UG01 está específicamente diseñado para aplicaciones de interior, permitiendo medidas hasta 5.6 metros con un ángulo de visión de 240º. La alta precisión de las medidas suministradas por el sensor (3% del valor medido) y su alta resolución angular (0.352º) proporcionan una serie de medidas muy fiables.

Figura 2. HOKUYO URG-04LX-UG01

Para el control del range-finder, Robotino OS proporciona unos drivers basados en una implementación libre de los drivers para ROS (Robot Operative System [11]).

2.3 MONTAJE

Dado que los puertos USB del Robotino son incapaces de suministrar los 1.5A que el URG-04LX-UG01 necesita al inicio, se procedió al diseño y montaje de un pequeño soporte donde alojar un hub USB que alimenta al range-finder.

Además se ha integrado en la plataforma del robot móvil un circuito que transforma y suministra la alimentación necesaria de las propias baterías del Robotino, incorporándose adicionalmente un brazo de 4 grados de libertad (ver Figura 3).

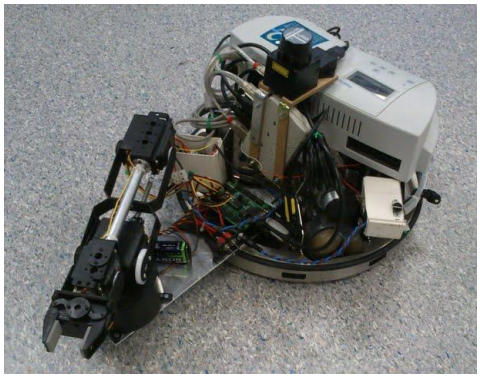


Figura 3. Robotino modificado

2.4 ARQUITECTURA DEL SISTEMA

Dado que el PC empotrado que controla al Robotino tiene una capacidad de cálculo limitada, en este trabajo se ha optado por el uso de un PC externo en el cual, mediante una conexión inalámbrica, el Robotino delega los cálculos del algoritmo de localización.

Por sencillez, y para mantener las comunicaciones lo más ligeras posible, también se alberga en el PC externo el mapa del entorno. Sin embargo, puesto que generalmente dicho mapa suele tener un tamaño reducido, se podría almacenar en la memoria del Robotino sin ningún tipo de problema.

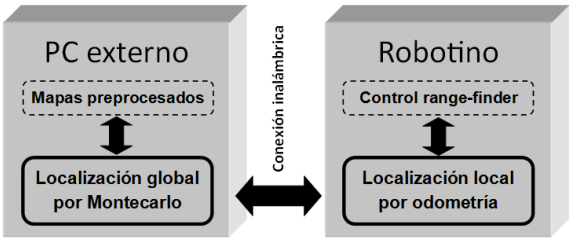


Figura 4. Arquitectura del sistema

Así pues, como se observa en la Figura 4, la subdivisión inicial en dos del problema de la localización del robot se asigna a dos actores diferentes. El PC empotrado en el Robotino es el encargado de realizar el seguimiento local de la posición mientras que el PC externo es quien, en base a las medidas obtenidas del range-finder y suministradas por el Robotino, calcula el posicionamiento global del robot.

3 ALGORITMO DE LOCALIZACIÓN POR MONTECARLO

3.1 PLANTEAMIENTO

El objetivo cuando se aborda el problema de la localización es la estimación de la posición en un instante de tiempo t con un cierto conocimiento sobre el estado inicial, la medida de los sensores en dicho instante (m_t) y con el conocimiento de todas las medidas $M_t = \{m_i, i = 0 \dots t\}$ obtenidas por el robot hasta el instante t . En este caso se trabaja con un vector de estado $s_t = [x\ y\ \theta]$ que contiene la posición y la orientación del robot. Por lo tanto, se puede definir la probabilidad de que una posición s_t sea la actual del robot en el instante t como $p(s_t|M_t)$. Si se abstrae este planteamiento, el cálculo de la probabilidad $p(s_t|M_t)$ para todas las posiciones posibles del robot proporciona la función de densidad de la probabilidad de la posición del robot en un momento dado.

3.2 REPRESENTACION DE LA FUNCIÓN DE PROBABILIDAD

Dado que es imposible calcular las probabilidades de todas las infinitas posiciones posibles del robot, lo que se hace es de escoger un numero N de posiciones (en adelante partículas) representativas del espacio. Este conjunto de posiciones permite establecer la densidad de la probabilidad (o lo que es lo mismo la posición del robot y su incertidumbre).

3.3 CÁLCULO DE LA FUNCIÓN DE PROBABILIDAD

La forma de calcular la función de densidad de la probabilidad se hace partiendo de un estado inicial y realizando un cálculo en dos fases de forma iterativa. La primera, se denomina fase de predicción y, la segunda, fase de actualización.

Así pues, inicialmente, se reparte de forma aleatoria y homogénea el número de partículas N por el espacio del mapa. Esto es así ya que inicialmente, como se había comentado anteriormente, la única información de la que se dispone es que el robot se encuentra dentro del mapa conocido a priori.

Seguidamente se procede a la fase de predicción. Durante esta fase, basándose en el modelo de movimiento del control local del robot, se aplica a cada una de las partículas el movimiento realizado por el propio robot durante el intervalo de tiempo entre el instante inicial y el actual. A dicho desplazamiento se le aplica un pequeño ruido gaussiano que tiene como objetivo modelar el error de un desplazamiento calculado exclusivamente a través de la odometría del robot, así como dispersar las partículas que por azar se encuentren muy próximas [9].

Seguidamente se realiza la fase de la actualización. Durante esta fase se cotejan las medidas del sensor láser con las medidas de un sensor láser virtual aplicado a cada partícula. Dicho láser virtual traza rectas en base a la posición y orientación de la partícula y devuelve la distancia mínima a la que intersectan con el mapa. A continuación se calcula la probabilidad de que cada partícula pr corresponda a la verdadera posición del robot, en función de sus k medidas m respecto a las k medidas reales m' del sensor láser de la siguiente manera:

Para cada pr_i

$Si\ pr_i\ fuera\ del\ mapa$

$$p(pr_i = s_t) = 0$$

$si\ no$

$$p(pr_i = s_t) = \prod_{j=1}^{k} e^{\left(-\frac{\left(m'_j - m_j\right)^2}{2\sigma^2}\right)}$$

Para cada par de medidas esta función calculará un peso entre 0 y 1 en función de la similitud que tienen. El productorio de dichos pesos proporcionará la probabilidad total de que dicha partícula se corresponda con la posición global del robot.

Hay que tener en cuenta que al tratarse de un productorio, dependiendo del valor de k escogido, es posible provocar un subdesbordamiento en el cálculo de la probabilidad para un número de k elevado. Este problema se puede evitar si se reconvierte dicha función a una suma de logaritmos [12].

Una vez obtenidas todas las probabilidades se escoge un umbral aleatorio, eliminándose del conjunto todas las partículas con una probabilidad inferior. Seguidamente las partículas supervivientes son clonadas hasta volver a disponer de las N partículas iniciales y se estima la posición del robot en base al cálculo de la media ponderada de todas ellas. En caso de no haber ninguna partícula que supere el umbral se mantiene el conjunto de partículas tal y como está.

Este cálculo en dos fases se repetirá indefinidamente. Como ya se ha comentado, mediante la introducción de ruido gaussiano en la fase de predicción, las partículas clonadas se dispersarán fomentando que las partículas se agrupen en torno a áreas de interés, en lugar de estar repartidas por todo el mapa. Eventualmente todas las partículas convergerán a una zona muy concreta del mapa determinando la posición del robot real.

4 EXPERIMENTOS REALIZADOS

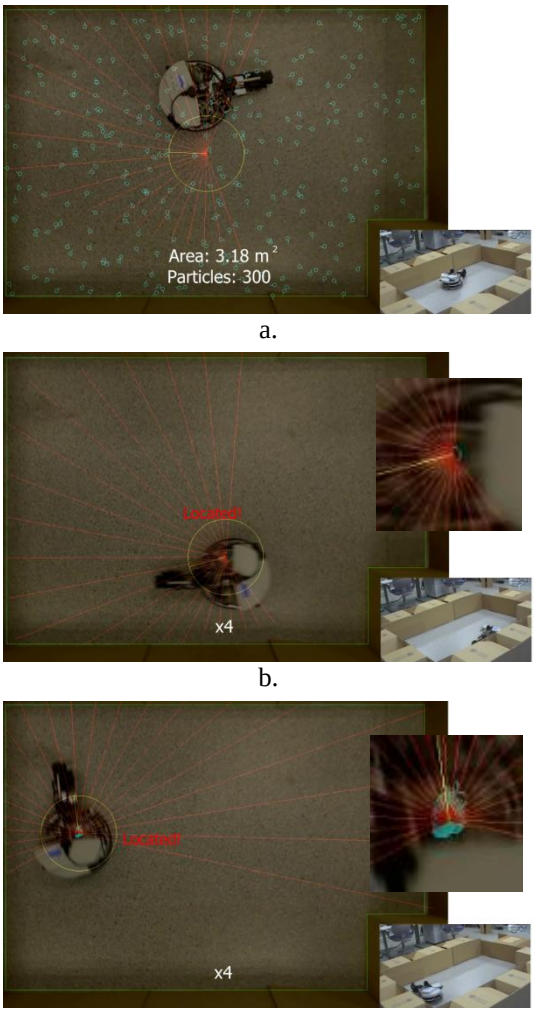
4.1 LOCALIZACIÓN EN ESCENARIO SENCILLO

Para el primer experimento se diseñó un escenario pequeño de unos $3.18\ m^2$ aproximadamente y se calculó el mapa correspondiente al mismo de forma manual. Para el cálculo de posición en este experimento se emplearon 300 partículas.

Como se puede apreciar en la sucesión de imágenes de la Figura 5, en el estado inicial (a), las partículas acaban distribuidas por todo el mapa, representando que durante este estado la única certidumbre que se tiene es que el robot se encuentra dentro del mapa.

Tras algunos desplazamientos alrededor del mapa, el robot consigue finalmente localizarse (b). En este caso, todas las partículas han acabado siendo clones de una sola partícula como se aprecia en el detalle a la derecha de la imagen.

Conforme pasan las iteraciones, a pesar de que el ruido gaussiano introducido tiende a esparcir las partículas, la fase de actualización tiende a agruparlas. Así pues, como se observa en la última imagen de la serie (c), el sistema es capaz de mantener la posición del robot con una incertidumbre muy pequeña.



a.

b.

c.

Figura 5. Resultados del primer experimento.

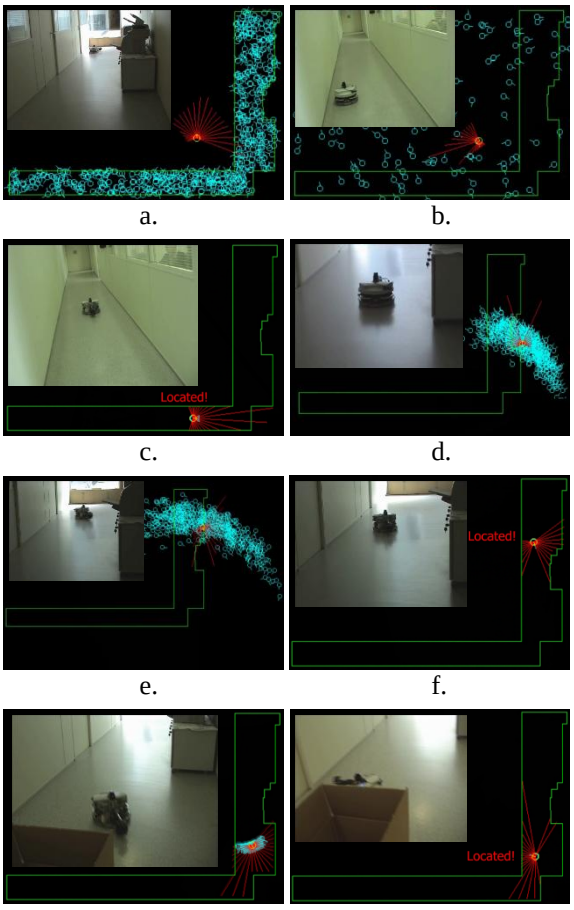
4.2 LOCALIZACIÓN EN ESCENARIO COMPLEJO

Para el segundo experimento se calculó el mapa correspondiente a un espacio acotado de parte de los pasillos del Instituto Universitario de Automática e Informática Industrial de la Universidad Politécnica de Valencia. El área de dicha zona abarca unos 33 m^2 aproximadamente. En este caso se emplearon 500 partículas para el cálculo de la posición.

Al igual que en el experimento anterior, tal y como se puede apreciar en la serie de capturas de los datos del experimento de la Figura 6, en el estado inicial (a), las partículas son esparcidas de forma uniforme por todo el mapa.

Dada la complejidad añadida del mismo, a las pocas iteraciones las partículas están totalmente esparcidas y muchas de ellas fuera del mapa (b). Aun así el sistema es capaz de encontrar una primera aproximación (c), aunque es claramente visible (observando las medidas del sensor láser) que ésta no es del todo correcta. Debido a esto, en (d) y (e) el ruido gaussiano llega a esparcir nuevamente las partículas de forma notable, mientras el sistema busca de una aproximación mejor.

Finalmente, el algoritmo obtiene una segunda aproximación (f) bastante ajustada a la realidad, aunque no lo suficiente. Por ello, el ruido gaussiano, actúa de nuevo (g) expandiendo la nube de partículas hasta que, por último, el algoritmo devuelve una aproximación definitiva (h) con una incertidumbre lo bastante baja como para hacer un seguimiento veraz de la posición global del robot.



a. b.

c. d.

e. f.

g. h.

Figura 6. Resultados del segundo experimento.

4.3 ROBUSTEZ ANTE PERTURBACIONES

Como último experimento, se comprobó la capacidad de reacción del sistema (una vez localizado globalmente) ante perturbaciones puntuales. Al no encontrarse en el mapa, éstas provocarían que durante la fase de actualización, las medidas obtenidas no se correspondan con la de ninguna partícula, con lo que las partículas comenzarían a disgregarse debido al ruido gaussiano.

Como se aprecia en la Figura 7, ante una perturbación puntual (como puede ser que haya una o varias personas caminando por el área mapeada) el sistema se comporta de la forma anteriormente descrita, esparciendo las partículas alrededor de la última posición conocida con baja incertidumbre, con lo que la incertidumbre de la posición aumenta. Sin embargo, en el instante que la perturbación desaparece, el hecho de que el ruido disperse las partículas siempre alrededor de un posición de interés, permite al sistema recuperarse casi de forma instantánea en la mayoría de los casos, necesitando unos pocos segundos en el resto de casos.

Figura 7. Robustez ante perturbaciones.

5 CONCLUSIONES

Durante el presente artículo, se ha presentado el problema de la localización de un robot móvil en un marco de referencia absoluto, explicándose cómo se pueden utilizar funciones de densidad de probabilidad para modelar el posicionamiento del robot con una cierta incertidumbre y se ha expuesto un algoritmo basado en el método de Montecarlo que permite el cómputo de dicha densidad.

Los diversos experimentos planteados, han demostrado que el algoritmo implementado es capaz de dotar al robot de capacidad para localizarse en escenarios complejos como los que componen los espacios arquitectónicos por los que nos movemos a diario con una incertidumbre sobre su posición muy baja.

Por ultimo, se ha comentado el comportamiento que presenta el robot cuando se estresa al algoritmo con perturbaciones comunes en su entorno de trabajo, como es el tránsito de personas. En este aspecto el resultado ha sido muy satisfactorio ya que el sistema es capaz de computar de nuevo la posición global en cuanto la perturbación desaparece, de forma casi instantánea en la mayoría de los casos.

El único inconveniente observado durante el desarrollo de este algoritmo es su alta ineficacia ante mapas completamente simétricos debido a la imposibilidad de resolver la ambigüedad sobre su posición utilizando exclusivamente información sobre la geometría de su entorno. Aún así, como ya se ha expuesto, este problema puede ser resuelto mediante la inclusión de más información en el mapa. Ésta puede depender tanto de características físicas naturales del medio (colores, temperaturas, etc…), como artificiales. Por ejemplo, se podría especificar la posición de puntos de acceso a redes inalámbricas y eliminar la ambigüedad contrastando la intensidad de la señal (método cricket aplicado a señales WiFi).

Por todo lo expuesto, asumiendo que a día de hoy existen métodos que permiten obtener mapas del entorno fiables de forma sencilla y, dada la dificultad de encontrar espacios perfectamente simétricos dentro de las construcciones actuales, es posible afirmar que el método propuesto resuelve de forma sencilla, fiable y eficiente el problema de la localización en interiores.

Agradecimientos

Los autores desean expresar su agradecimiento al Ministerio de Ciencia e Innovación de España por la financiación parcial de este trabajo bajo los proyectos de investigación DPI2011-28507-C02-01 y DPI2010-20814-C02-02 (FEDER/CICYT).

Referencias

[1] Dellaerty, F., Foxy, D., Burgardz, W., Thruny, S., Robust Monte Carlo Localization for Mobile Robots. Artificial Intelligence, vol 128 Issue 1-2, May 2001.

[2] F. Dellaert, D. Fox, W. Burgard, S. Thrun. MonteCarlo localization for mobile robots. Proc. IEEE International Conference on Robotics and Automation (ICRA-99), Detroit, MI (1999).

[3] Zhang, L., Zapata, R., Lépinay, P. Self-Adaptive Monte Carlo Localization for Mobile Robots Using Range Sensors. IEEE/RSJ International Conference on Intelligent Robots and Systems (2009).

[4] Gleason, C.P.; Perez, L.C.; Goddard, S.; , "On the Ranging Connectivity in the Cricket Localization System," IEEE International Conference on Electro/information Technology, (2006), pp.619-624. doi: 10.1109/EIT. 2006.252217

[5] Festo. Didactic. http://www.festo-didactic.com http://wiki.openrobotino.org

[6] Kapse, A.; Dongbing Gu; Zhen Hu; "Using cricket sensor nodes for Pioneer robot localization," International Conference on Mechatronics and Automation ICMA (2009), pp. 2008-2013, doi: 10.1109/ICMA.2009. 5246393

[7] Nakajima, S.; Ikejiri, M.; Toriu, T.; , "Monte Carlo Localization Robust against Successive Outliers," Fourth International Conference on Innovative Computing, Information and Control ICICIC (2009), pp.1515-1518. doi: 10.1109/ICICIC.2009.268

[8] Weidong Wang; Qingxin Zhu; "Varying the Sample Number for Monte Carlo Localization in Mobile Sensor Networks", Second International Multi-Symposiums on Computer and Computational Sciences IMSCCS (2007), pp.490-495. doi: 10.1109/IMSCCS.2007.49

[9] Tiancheng Li; Shudong Sun; Jun Duan; "Monte Carlo localization for mobile robot using adaptive particle merging and splitting technique," IEEE International Conference on Information and Automation ICIA (2010), pp.1913-1918, doi: 10.1109/ICINFA.2010. 5512017

[10] HOKUYO photo sensors http://www.hokuyo-aut.jp

[11] ROS http://www.ros.org/wiki/

[12] Wen, Y., Using log-transform to avoid underflow problem incomputing posterior probabilities (2007). http://web.mit.edu/ wenyang/www/log_transform_for_underflow.p df

[13] Smith, R.; Self, M.; Cheeseman, P. “Estimating uncertain spatial relationships in robotics”. IEEE International Conference on In Robotics and Automation (1987).