Inteligencia Artificial, 23(65), 33-55
doi: 10.4114/intartif.vol23iss65pp33-55

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Cost-effective Indoor Localization for Autonomous Robots using Kinect and WiFi Sensors

Raulcézar Alves[1], Josué Silva de Morais[1], and Keiji Yamanaka[1]
[1]FEELT - Faculdade de Engenharia Elétrica
UFU - Universidade Federal de Uberlândia, Uberlândia, MG, Brazil.
raul@ufu.br, josue@ufu.br, keiji@ufu.br

Abstract: Indoor localization has been considered to be the most fundamental problem when it comes to providing a robot with autonomous capabilities. Although many algorithms and sensors have been proposed, none have proven to work perfectly under all situations. Also, in order to improve the localization quality, some approaches use expensive devices either mounted on the robots or attached to the environment that don't naturally belong to human environments. This paper presents a novel approach that combines the benefits of two localization techniques, WiFi and Kinect, into a single algorithm using low-cost sensors. It uses separate Particle Filters (PFs). The WiFi PF gives the global location of the robot using signals of Access Point devices from different parts of the environment while it bounds particles of the Kinect PF, which determines the robot's pose locally. Our algorithm also tackles the Initialization/Kidnapped Robot Problem by detecting divergence on WiFi signals, which starts a localization recovering process. Furthermore, new methods for WiFi mapping and localization are introduced.

Keywords: Autonomous Robots, Indoor Localization, Particle Filter, WiFi, Kinect.

1 Introduction

The ability to localize objects, people and devices within indoor environments has become increasingly important for many applications. With the emergence of technologies such as Global Positioning System (GPS), the performance of outdoor localization has become excellent. However, the signal from GPS satellites is too weak to come across most buildings, thus making GPS ineffective for indoor localization [1]. Therefore indoor localization became a focus of research and development during the past decades.

Indoor environments are particularly challenging for localization for several reasons, such as severe multipath from signal reflection from walls and furniture, Non Line of Sight (NLoS) conditions, high attenuation and signal scattering due to greater density of obstacles, fast temporal changes due to the presence of people and opening of doors, high demand for precision and accuracy, and others. However, indoor settings facilitate localization in many ways, like the presence of small coverage areas, low weather influences such as small temperature gradients and slow air circulation, fixed geometric constraints from planar surfaces and orthogonality of walls, infrastructures (such as electricity, internet access, and walls) suitable for target mounting, lower dynamics due to slower walking and driving speeds, etc.

When considering the drawbacks of this scenario, apparently there is no overall solution based on a single technology for indoor environments. Thus, we are still far away from achieving cheap provision of indoor positioning with an accuracy of 1 meter [2].

ISSN: 1137-3601 (print), 1988-3064 (on-line)

The works of [2] and [3] provide a technological perspective of indoor positioning systems, comprising a wide range of technologies and approaches. They show that low cost technologies, such as WiFi and Computer Vision, can achieve at least 1.5m and 1m of accuracy respectively. On the other hand, ZigBee and Ultra Wide Band (UWB) have much more precision, 25cm and 15cm, but the end user cost is high.

Some of these technologies and sensors are widely applied to indoor robot localization, but they are just one part of the problem. Usually, a mobile robot relies on observations made by its sensors, estimates motions given by odometry and pre-built maps, to reason upon its location in the environment.

There are many map representations in the literature, and the most common are: topological maps, feature maps, grid maps, and appearance based methods. Once there is a map of the environment, algorithms can infer the position of the robot on the map based on observations made by its sensors. In most cases, pose estimation algorithms use Probability Density Functions (PDFs) to handle the pose uncertainty given by motion and sensor noise.

In general, these algorithms can be grouped into: Dead-Reckoning, Topological, Gaussian PDF, Gaussian Sum PDF, Position Probability Grids, and Monte Carlo Methods. Two algorithms have become increasingly popular over the last few years in robotics, the Kalman Filter (KF) [14] and Particle Filter (PF) [15].

KF based techniques have proven to be robust and accurate for robot localization, however, it can not represent ambiguities and lacks the ability to (re-)localize the robot in the case of localization failures. For this reason, we believe that PF is more suitable for our work on indoor localization, as we tackle the "Initialization" and "Kidnapped Robot" problems.

This work aims to provide a localization algorithm with reasonable accuracy fed by low-cost devices, such as Microsoft Kinect and WiFi dongle. Microsoft Kinect is commonly used in robotics [16]. Unfortunately, indoor human environments consist of many movable objects like chairs and tables, and moving objects like humans, that can occlude features of the environment used for localization. On the contrary, WiFi based techniques are not so affected by changing objects. They are not costly either and are applicable anywhere there is dense WiFi network, which is the case of many human environments today [19]. However, some issues can impact the localization accuracy, such as: quality of Acess Point (AP) devices, quantity of APs, distribution of APs along the environment, noise on the radio frequency, and others.

One of our goals is to use the benefits of Kinect and WiFi localization in a single positioning algorithm. In such algorithm, the WiFi localization estimates the global location of the robot as APs are usually well spread along the environment. This global location determines a perimeter for the Kinect approach. Then, the estimate given by the Kinect localization is taken as the robot's current location, since it has a better local precision.

Both localization approaches use Particle Filter to maintain a set of hypotheses of the robot's location in real time. Regarding the Kinect localization, it was applied the Adaptive Monte Carlo Localization (AMCL) available on our robot. For the WiFi localization, we extended the work of [20] to indoor open areas. We also created new methods for WiFi mapping and localization, including the concept of irregular WiFi grid maps, different interpolation strategies for estimating WiFi signal continuously in the grid, and algorithms to replace particles and infer the robot's pose.

We carried out different experiments. In the first one, we compared the localization accuracy along a path within an indoor area using Kinect, WiFi and Kinect+WiFi. For the second, we tested the ability of our robot to recover its positioning autonomously in case it is lost (Initialization and Kidnapped Robot Problem). The last experiment evaluated the accuracy of WiFi localization with different amounts of APs.

A robotic platform, called Turtlebot-2 (Figure 1), was used for these experiments. This robot consists of an Yujin Kobuki base, wheel encoders, wheel drop sensors, an integrated gyroscope, bump sensors, cliff sensors, a 2200 mAh battery pack, a Microsoft Kinect sensor, an Intel NUC i7, a fast charger, a WiFi dongle, and a hardware mounting kit attaching everything together.

The remainder of the paper is organized as follows. A localization approach based on Kinect sensor is shown in Section 2 with related work, mapping technique, and a localization algorithm. In Section 3 is presented the proposed WiFi localization, which also includes related work, mapping technique, and a localization algorithm. Section 4 addresses how both Kinect and WiFi localization methods were used in a single algorithm. Experiments using all three approaches and their results are provided in Section 5. Finally, the conclusion is presented in Section 6.

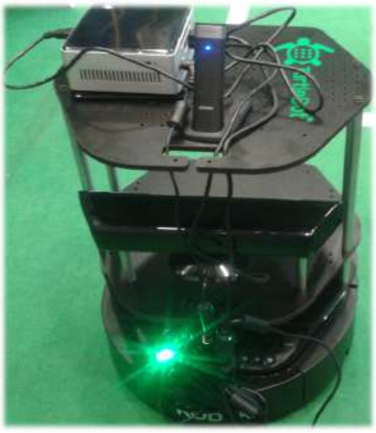


Figure 1: Turtlebot 2: Robotic Platform used for experiments with Kinect and WiFi devices.

2 Kinect Localization

Depth cameras are revolutionizing many fields in robotics. They are an alternative solution to laser range finders and stereo vision systems. But only recently, with the advent of Microsot Kinect, depth cameras started providing decent quality depth information of the surrounding environment at a highly competitive price. This device consists of an infrared laser emitter, an infrared camera and a RGB camera, with which it produces depth in addition to color of each pixel of a 640 x 480 frame, also known as RGB-D, at a frame rate of up to 30 fps.

2.1 Related Work

The robotics community has developed many techniques for indoor localization and mapping using the Kinect sensor. [21] introduces RGB-D Mapping, a framework that can generate dense 3D maps of indoor environments despite the limited depth precision and field of view provided by RGB-D cameras.

In [22], an algorithm is presented for filtering depth images into point clouds corresponding to local planes. Then, an observation model matches the plane filtered points to lines in the existing 2D maps for localization.

[23] introduce a reliable Kinect based navigation in large indoor environments. This approach includes methods for: (1) processing depth frames, (2) continuous incremental mapping to generate reliable local maps from extremely noisy readings, and (3) integrating local maps into a navigation pipeline.

A topological map-building-based method for localization and navigation is proposed in [24]. This method generates topological maps identifying corridors using a Bayesian classifier on depth curves provided by a Kinect sensor. Based on the map and a Markov localization method, the robot can then localize itself and navigate freely in the indoor environment.

Another method of robot navigation in corridors is proposed in [25], which combines the use of geometrical features of the environments and probabilistic occupancy information obtained from a Kinect sensor equipped on a mobile robot. The proposed method does not require a previously established map of the corridor environment and is suitable for application in unknown environments.

For the Kinect localization experiments presented in this work, the GMapping [26] algorithm was used to generate a grid map of our indoor environment. Following this, the robot's pose was estimated by the Adaptive Monte Carlo Localization (AMCL) [27] algorithm during the navigation. The standard implementation of both algorithms was used, they are available on the Robot Operating System (ROS) packages of Turtlebot-2. ROS is an open-source, meta-operating system for robots. It provides services expected from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple robots and computers.

2.2 Mapping

The mapping phase is very complex, because for localization a robot needs a map and for acquiring a map a robot needs to estimate its location continuously. This problem is often referred to the Simultaneous Localization and Mapping (SLAM) problem [28].

For the Kinect localization experiments presented in this work, the GMapping [26] algorithm was used to generate a occupancy grid map [9] of our indoor environment (Figure 2). When the robot is moving, it keeps updating the 2D map representation marking cells as occupied, free, or unknown.

GMapping is the most widely used laser-based SLAM algorithm in robotics, which is an implementation of the Rao-Blackwellized Particle Filter (RBPF) SLAM approach proposed by [26]. This algorithm introduces two improvements to the standard PF approach for the SLAM problem, an adaptive resampling technique to minimize the particle depletion, and an accurate distribution that takes into account the movement and also the most recent observations made by the robot.

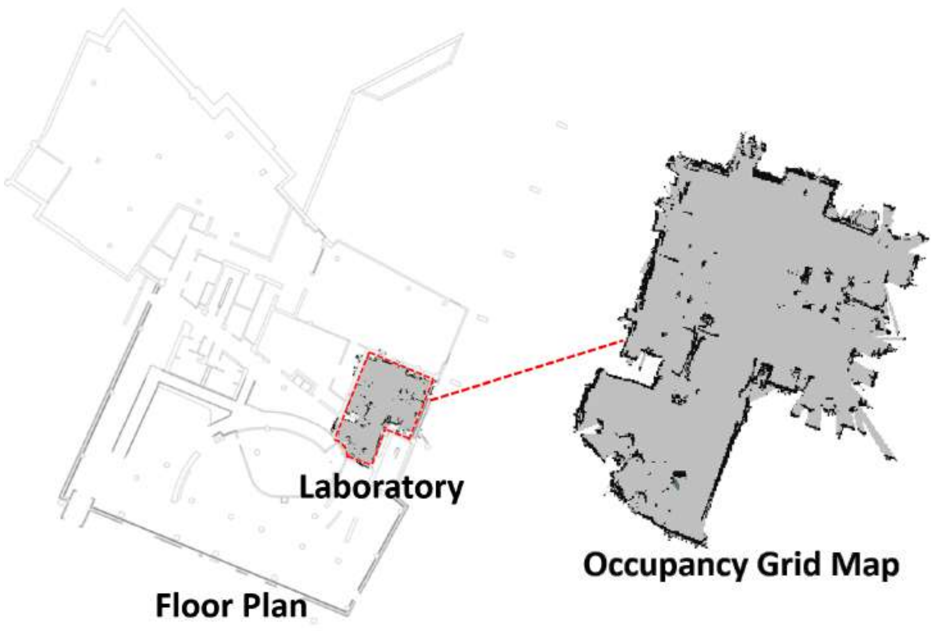


Figure 2: **Occupancy Grid Map of the laboratory.** White pixels of the occupancy grid map file represent unknown cells that could not be scanned, gray pixels are free cells where the robot can traverse, and black cells are occupied cells representing obstacles found during the mapping, such as walls, tables, chairs, cabinets and other objects.

2.3 Localization Algorithm

Once the map is built, robots can use it to estimate their positions in the environment using different localization methods. The Adaptive Monte Carlo Localization (AMCL), illustrated by Figure 3, is a PF based algorithm that implements KLD-sampling [27] to manage the number of particles during localization.

The AMCL process is composed of the following steps:

- *initialization*: we can either set the robot to some position on the map, in which case we are manually localizing it and concentrating the particles on this area based on a normal distribution, or we could very well make the robot start from no initial estimate of its position by spreading all particles along the map.
- *predict*: as the robot moves, the algorithm moves the particles according to the odometry data to predict the robot's position after a motion command. As these data can contain noise, the

odometry estimation in ROS, specially in Turtlebot-2, is computed by an Extended Kalman Filter (EKF) using wheel sensor, bump sensor, gyroscope, robot structure parameters and others.

- *update*: when the Kinect laser scans are received and transformed into distances and angles, they are compared with the data collected and saved previously. The weight of a particle is updated by the probability of observing these distances and angles given the particle's location. The particles with the readings closest to those of the robot will be given a higher weights and will be chosen as possible poses for the robot itself.
- *resampling*: only particles with higher weights will be included in the process of localization, this will make the number of possible particles less and less with time and the robot's pose will be guessed correctly. On the other hand, particles with lower weights should be replaced by others close to the existing ones with higher probability. Also, if the uncertainty is still high, few random uniformly distributed particles should be added to help the robot recover itself in cases where it has lost track of its position.
- *estimate*: at any given time, the particle with the highest probability is picked as the current pose of the robot.

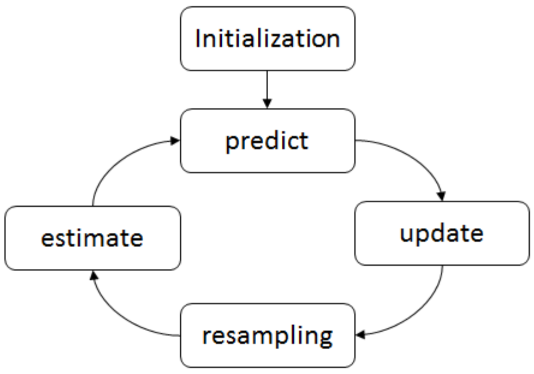


Figure 3: **AMCL.** In our experiments with Kinect localization, the standard implementation of AMCL was used on Turtlebot-2.

3 WiFi Localization

With the increase in Wireless Local Area Networks (WLAN) in human environments, considerable work has been carried out on using signal strength measurements from WiFi Access Points (APs) devices for indoor mapping and localization. Usually, these techniques use Radio-Frequency (RF) fingerprinting, where Received Signal Strength (RSS) measures in some RF band are collected using WiFi receivers, such as WiFi dongles, and compared to a map of previous collected signals.

3.1 Related Work

Several works on WiFi localization and mapping have been proposed. For instance, [30] introduce methods for creating a WiFi map from RF signals collected along robotic or pedestrian trajectories. The first method consists of naive fixed size grids, where WiFi samples collected by a freely moving robot fall into a specific spatial location cell an turn to a fingerprint. The next one is a clustering of RSS top-down into cells of increasingly smaller size using decision trees. Finally, in the last method is proposed an iterative bottom-up approach that aggregates smaller-size fingerprint cells into larger grid cells.

In [31], a localization method is presented for mobile robots based on RSS in indoor WLANs. An observation likelihood model is accomplished using kernel density estimation to characterize the dependence of location and RSS. Based on the measured RSS, the robot's location is dynamically estimated using an Adaptive Local Search Particle Filter (ALSPF).

The task of acquiring WiFi maps can be exhausting as most of them are created manually by driving robots through the environment and collecting signal strength measurements along with precise position information. Furthermore, it is necessary to repeat the mapping periodically to adapt to potential changes in the radio environment. [32] describe the design and working of a robotic platform capable of conducting repeated surveying of an indoor environment to update the signal maps as frequently as required. Also the robot serves as a testbed to collect data in controlled conditions, such as under uniform motion and with accurate positioning.

[20] propose a WiFi signal strength based localization algorithm that uses a parametric graph composed of discrete points for which WiFi signal strengths are collected. During navigation, a Particle Filter (PF) estimates the robot's location in five steps. (1) Predict: given new odometry data for the motion of the robot, the particles are moved using a motion model, where the robot's linear and angular motions are modeled as having Gaussian error. (2) Update: the weights of the particles are updated based on the latest WiFi signal strength observation using a probabilistic perceptual model that applies a linear interpolation to estimate signal strength values on unknown points between discrete known points on the graph. (3) Constrain: particles are constrained to the edges of the graph. (4) Resampling: particles are resampled based on Sensor Resetting Localization method. (5) Infer Location: K-Means clustering is performed on the particle set taking into account the weights of the particles, where the cluster with the largest weight represents the robot's location.

Our WiFi localization is based on that previous approach of [20]. The main difference is that their robots use graph representation for navigation and localization in the corridors of the buildings. Thus, hallways can be seen as edges and discrete known locations as vertices, which do not allow this approach to be used well within indoor open areas, such as atria, lobbies, laboratories, offices, etc. Instead, we introduce modifications on mapping and localization methods, so the robots can work not only in corridors, but also in open areas.

3.2 Mapping

A traditional way of building WiFi maps is by collecting repeated RSS measurements at predefined locations, represented as points in the form of (x, y) cartesian coordinates, in a 2D environment. In our approach, the robot collects RSS readings from all reachable Access Points (APs) at discrete points for a predetermined duration. Then, it computes mean and standard deviation values for each pair of discrete point and AP, which constitute the WiFi map.

Thus, the map $\mathbb{M}$ of the environment has the following structure $\mathbb{M} = \langle \mathbf{P}, \mathbf{A}, \mathbf{M}, \mathbf{D} \rangle$. The set **P** consists of all discrete points for which we collect RSS readings. **A** is the set of all accessible APs in the environment. **M** and **D** are matrices representing the RSS means and standard deviations. They are a combination of $\mathbf{P} \times \mathbf{A}$, i.e., the element $\mathbf{M}_{ij}$ of **M** is the mean RSS of AP a_j as measured from the discrete point p_i. Likewise, the element $\mathbf{D}_{ij}$ of **D** is the observed standard deviation of the RSS of AP a_j as measured from the discrete point p_i.

Tables 1 and 2 illustrate values for the matrices **M** and **D**. The mean RSS values stored in the matrix **M** are given in dBm, ranging from -20 (strong signal) to -90 (weak signal). Values in the matrix **D** are the standard deviation based on the mean value and the reading set. Therefore, a robot can estimate its location during the navigation by comparing its current RSS readings from some APs to a normal distribution based the matrices **M** and **D**.

P/A	a_1	a_2	...	a_n
p_1	-23	-34	...	-88
p_2	-27	-45	...	-61
...	...	...	...	...
p_m	-89	-63	...	-78

Table 1: Mean Matrix (**M**).

P/A	a_1	a_2	...	a_n
p_1	0.6	2.8	...	0.2
p_2	1.3	1.5	...	4.1
...	...	...	...	...
p_m	3.5	3.6	...	2.9

Table 2: Standard Deviation Matrix (**D**).

In this sense, WiFi maps consist of RSS values collected from some discrete points. Since robots are navigating in a continuous space, how can they estimate their locations at points where they did not collect any RSS values during the mapping? Usually, in this case, interpolation techniques are applied to estimate values for unknown points based on discrete known points.

There are many interpolation techniques for 2D spaces like linear, bilinear, nearest neighbor, and bicubic interpolation. However, accurate interpolation techniques have high computational cost, which is not desired in real time applications as robot localization, and low cost interpolations are not acceptable as they may increase the localization error. To deal with this trade-off in our work, depending on the location of a given point in the map, different types of interpolation are applied to estimate the RSS mean and standard deviation.

Our test environment was mapped in a grid fashion, where RSS values are collected at points in the corner of each cell. But depending on the shape of the environment, it is not possible to generate perfect grids with cells of same size and orientation. Thus, we introduced the concept of irregular grids in order that they be better suited to any environment.

In our experiments, the WiFi map of the laboratory was built based on the occupancy grid map generated with Kinect sensor (Section 2). Using the coordination system of this map, some discrete points were selected in order to form an irregular grid. Then, at each point, our Turtlebot collected RSS readings for 5 minutes, as depicted by Figure 4.

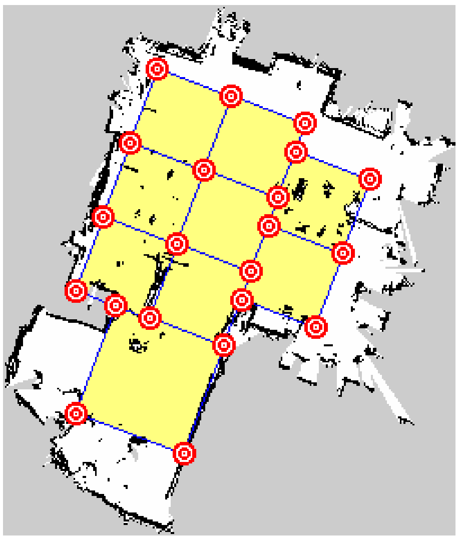


Figure 4: **Irregular Grid for WiFi Mapping.** Red targets are discrete points where RSS were collected, segments linking the points are shown in blue, and yellow squares are the interior of the cells

Therefore, for calculating the estimate RSS mean and standard deviation for any given point, the localization algorithm checks whether it lies on a discrete point, a segment, or inside of a cell, to apply the most appropriate interpolation technique.

3.3 Localization Algorithm

Given the WiFi map of the environment, the robot's location is estimated in real-time using a Particle Filter (PF). The algorithm maintains a set of particles $\mathbb{P}$ to represent the distribution of where the robot could be, i.e., each particle is a hypothesis of a possible location. A particle $p_i \in \mathbb{P}$ has the following properties $p_i = \langle l_i, \theta_i, w_i \rangle$:

- l_i: the cartesian location (x_i, y_i) of the particle on the map;
- θ_i: the orientation of the particle;
- w_i: the weight assigned to the particle.

In our tests, a passive "sniffing" technique was applied for gathering WLAN Received Signal Strength (RSS) to create WiFi observations, and also to collect data during the mapping phase. To do that, a

WiFi dongle (Linksys Dual-Band Wireless-N USB Adapter - AE3000) was mounted on our Turtlebot and used on monitoring mode to sniff out the WiFi network. Our sniffer *script* uses *tShark* commands to capture WiFi information, such as the IP address of an Access Point, its RSS value, and its Radio Frequency Channel. Three channels were chosen, 1, 6, and 11, as they do not overlap each other. In these channels, 124 APs were discovered, few of these belong to the laboratory, some of them were nearby, and the rest were spread along different floors of the building. An advantage of our approach is that, as we are not performing a triangulation to estimate the robot's position, it is not necessary to know the real location of the APs.

During the experiments it was noted that distant APs are more likely to have noise. Also, the time required to perform the localization algorithm using all APs was too long. To solve these issues the most relevant APs are filtered before they are used for localization. To do that, the APs set is reduced by selecting the AP with highest mean RSS for each discrete point. To avoid repetition, if an AP is already taken, the next available one is picked. As a result, the set was reduced to 21 APs.

The following sections describe in detail how each step of our WiFi localization process works.

3.3.1 Predict

The odometry data, denoted as $(\Delta_x^{odo}, \Delta_y^{odo}, \Delta_\theta^{odo})$, is read as incremental changes in the 2D robot's pose. Given the prior pose (x, y, θ) of each particle $p_i \in \mathbb{P}$, the *predict* step computes the new poses (x', y', θ') after the odometry reading using Equation 1. The vector $(\epsilon_x, \epsilon_y, \epsilon_\theta)$ refers to the Gaussian error added to the new pose.

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \left[\begin{pmatrix} cos(\theta + \frac{\Delta_\theta^{odo}}{2}) & -sin(\theta + \frac{\Delta_\theta^{odo}}{2}) & 0 \\ sin(\theta + \frac{\Delta_\theta^{odo}}{2}) & cos(\theta + \frac{\Delta_\theta^{odo}}{2}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \Delta_x^{odo} \\ \Delta_y^{odo} \\ \Delta_\theta^{odo} \end{pmatrix} \right] + \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_\theta \end{pmatrix} \tag{1}$$

3.3.2 Update

In order to update the weight of each particle $p_i \in \mathbb{P}$, whenever there is a new motion or not, a Perceptual Model is needed. The Perceptual Model is used to calculate the probability of making a particular signal strength measurement S at a location l, $P(S|l)$.

During the execution, the robot spends a time t listening to the APs of the environment to fill an observation set $S = [S_1, S_2, S_3, ..., S_{|\mathbf{A}|}]$. In our experiments, t was set to 1 second, which permits the collection of almost 30 RSS readings, from different APs or not. Thus, S is filled with the average RSS of each AP, where S_i is the average RSS measured from the AP i.

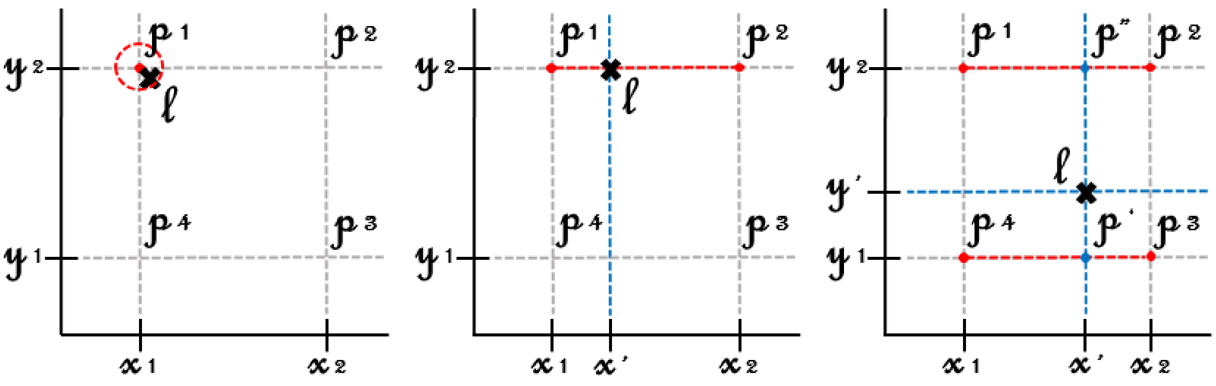
Once we have an observation S, the robot should be able to calculate the probability $P(S|l)$ from any location l in the environment, not only in the discrete points where RSS were collected during the off-line mapping. Therefore, using the discrete values of mean and standard deviation of each pair of discrete points and AP, a continuous Perceptual Model is proposed. It is composed of different strategies (*In Range*, *Linear Interpolation*, and *Bilinear Interpolation*) for estimating mean and standard deviation values for any location l on the map in order to calculate the probability $P(S|l)$ by the means of Gaussian Distributions.

Figure 5 illustrates an hypothetical example where points p_1, p_2, p_3, and p_4 constitute a cell in the WiFi grid map.

As described in Section 3.2, the map $\mathbb{M}$ has the matrices $\mathbf{M}$ and $\mathbf{D}$, which contain the WiFi signal strength means and standard deviations of every Access Point $a_k \in \mathbf{A}$ measured from each discrete point $p_i \in \mathbf{P}$. With this map, the WiFi signal strength mean and standard deviation are modeled as being piecewise linear along the map with Gaussian noise.

To better understand the estimation techniques, it is necessary to introduce the following formalism. Let $\mathbf{M}^l$ denote the vector of mean signal strengths of every Access Point as seen from location l. The component $\mathbf{M}_k^l$ of the vector $\mathbf{M}^l$ is the mean WiFi signal strength of Access Point $a_k \in \mathbf{A}$ as seen from location l. Similarly, let $\mathbf{D}^l$ denote the standard deviations of the signal strengths of each Access Point as seen from location l, with $\mathbf{D}_k^l$ being the standard deviation of the signal strength of Access Point a_k as seen from location l.

Given a location l and the WiFi grid map $\mathbb{M}$, the Perceptual Model works as follows.



(a) In Range. (b) Linear. (c) Bilinear.

Figure 5: **RSS estimation for any point of the map.** Estimating Mean and Standard Deviation Values for RSS on the WiFi Grid. The cross marks a location l where the mean and standard deviation are unknown. To estimate them, known values of the locations p_1, p_2, p_3, and p_4 can be used by the estimation strategies.

- *In Range Estimation*: first it checks whether l lies on any point $p_i \in \mathbf{P}$, or if it is near to some point in a certain range. Considering the scenario illustrated in Figure 5 (a), the location l is close to the point p_1 located in a corner of some cell of the WiFi grid. In such a situation, the Perceptual Model uses the mean and standard deviation of p_1 for the location l. Therefore, the mean signal strengths vector $\mathbf{M}^l = [\mathbf{M}^l_1, \mathbf{M}^l_2, ..., \mathbf{M}^l_{|\mathbf{A}|}]$, and the standard deviations vector $\mathbf{D}^l = [\mathbf{D}^l_1, \mathbf{D}^l_2, ..., \mathbf{D}^l_{|\mathbf{A}|}]$ at location l and Access Point k are given by:

$$\mathbf{M}^l_k \approx \mathbf{M}^1_k \quad (2)$$

$$\mathbf{D}^l_k \approx \mathbf{D}^1_k \quad (3)$$

- *Linear Interpolation Estimation*: when the location l is not close to any point, the Perceptual Model verifies if it belongs to a segment in some cell of the grid. Once the right segment is found, a Linear Interpolation is performed using the end points of the segment.

  Hence, following the example of Figure 5 (b), the mean signal strengths vector $\mathbf{M}^l$ and the standard deviations vector $\mathbf{D}^l$ at location $l = (x', y_2)$, between the points $p_1 = (x_1, y_2)$ and $p_2 = (x_2, y_2)$, and Access Point a_k are given by:

$$\mathbf{M}^l_k \approx \frac{[(x_2 - x') \times \mathbf{M}^1_k] + [(x' - x_1) \times \mathbf{M}^2_k]}{(x_2 - x_1)} \quad (4)$$

$$\mathbf{D}^l_k \approx \frac{[(x_2 - x') \times \mathbf{D}^1_k] + [(x' - x_1) \times \mathbf{D}^2_k]}{(x_2 - x_1)} \quad (5)$$

  The same idea is applied when the location l lies on other segments of the cell.

- *Bilinear Interpolation Estimation*: finally, if the location l is neither on a point nor on a segment, the Perceptual Model finds the most appropriate cell to perform a Bilinear Interpolation.

  The key idea is to perform Linear Interpolation first in one direction, and then again in the other direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location.

  Given the means and standard deviations of the points $p_1 = (x_1, y_2)$, $p_2 = (x_2, y_2)$, $p_3 = (x_2, y_1)$, and $p_4 = (x_1, y_1)$ of our example, depicted by Figure 5 (c), the Perceptual Model first performs a Linear Interpolation in the x-direction for each Access Points $a_k \in \mathbf{A}$ using inner points p' and p'' extracted from the location l. This yields:

$$\mathbf{M}_k^{p'} \approx \frac{[(x_2 - x') \times \mathbf{M}_k^4] + [(x' - x_1) \times \mathbf{M}_k^3]}{(x_2 - x_1)} \tag{6}$$

$$\mathbf{M}_k^{p''} \approx \frac{[(x_2 - x') \times \mathbf{M}_k^1] + [(x' - x_1) \times \mathbf{M}_k^2]}{(x_2 - x_1)} \tag{7}$$

$$\mathbf{D}_k^{p'} \approx \frac{[(x_2 - x') \times \mathbf{D}_k^4] + [(x' - x_1) \times \mathbf{D}_k^3]}{(x_2 - x_1)} \tag{8}$$

$$\mathbf{D}_k^{p''} \approx \frac{[(x_2 - x') \times \mathbf{D}_k^1] + [(x' - x_1) \times \mathbf{D}_k^2]}{(x_2 - x_1)} \tag{9}$$

Then, the Perceptual Model proceed by interpolating in the y-direction to obtain the desired estimate:

$$\mathbf{M}_k^{l} \approx \frac{[(y_2 - y') \times \mathbf{M}_k^{p'}] + [(y' - y_1) \times \mathbf{M}_k^{p''}]}{(y_2 - y_1)} \tag{10}$$

$$\mathbf{D}_k^{l} \approx \frac{[(y_2 - y') \times \mathbf{D}_k^{p'}] + [(y' - y_1) \times \mathbf{D}_k^{p''}]}{(y_2 - y_1)} \tag{11}$$

With these estimation techniques, we can model the WiFi signal strength distribution of an Access Point at any location as a Gaussian Distribution using the means and standard deviations.

Suppose there are M locations representing positions of particles on the map. To compute the weight for all particles at locations $l_1, l_2, l_3, ..., l_{|M|}$ given a new observation $S = [S_1, S_2, ..., S_{|\mathbf{A}|}]$, it is necessary to calculate the joint probabilities of all Access Points $a_k \in \mathbf{A}$, as illustrated by Figure 6.

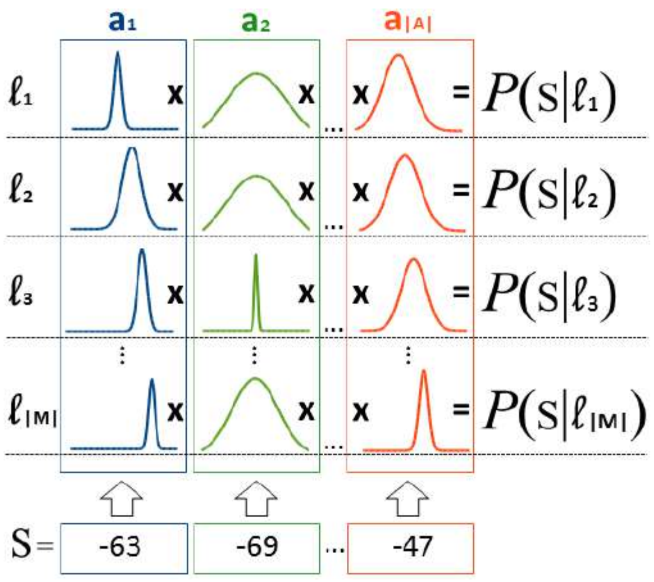


Figure 6: **Perceptual Model.** Join probabilities to update the weight of the particles. Each line represents the probability product of a gaussian set modeled by the mean and standard deviation on each access point. Given the current value of each access point in S, $P(S|l)$ calculates the probability of hearing S at each location l where a particle is found.

However, probabilities are often small numbers, and in this case, we have to multiply all of them together, which gives an even smaller number. Besides, it is possible to get into difficulty with the precision of floating point values.

Another problem is that the computation can be very slow, which should be avoided in real-time applications. To overcome these issues, we work in the log probability space, i.e., we take the logarithm of the probabilities.

Thus, the probability of making the observation S from a location l, ignoring the unobserved Access Points (i.e. $S_i : S_i = 0$) is given by Equation 12.

In order to evaluate how distinguished the points are on our WiFi map of the laboratory, we used their mean RSS values as observations. The behaviour of the estimated probability given by our Perceptual Model $P = (S|l)$ (Equation 12) along the map is depicted by Figure 7.

$$P(S|l) = \prod_{i=1,S_i\neq 0}^{i=|A|} \left(\frac{1}{\sqrt{2\pi}\mathbf{D}_i^l} \times \exp^{-\frac{(S_i-\mathbf{M}_i^l)^2}{2\mathbf{D}_i^{l\,2}}} \right) = \left(\frac{1}{\sqrt{2\pi}\mathbf{D}_1^l} \times \exp^{-\frac{(S_1-\mathbf{M}_1^l)^2}{2\mathbf{D}_1^{l\,2}}} \right) \times ... \times \left(\frac{1}{\sqrt{2\pi}\mathbf{D}_{|A|}^l} \times \exp^{-\frac{(S_{|A|}-\mathbf{M}_{|A|}^l)^2}{2\mathbf{D}_{|A|}^{l\,2}}} \right)$$

$$\log\left(P(S|l)\right) = \left(\log\left(\frac{1}{\sqrt{2\pi}\mathbf{D}_1^l} \right) - \frac{(S_1-\mathbf{M}_1^l)^2}{2\mathbf{D}_1^{l\,2}} \right) + ... + \left(\log\left(\frac{1}{\sqrt{2\pi}\mathbf{D}_{|A|}^l} \right) - \frac{(S_{|A|}-\mathbf{M}_{|A|}^l)^2}{2\mathbf{D}_{|A|}^{l\,2}} \right) = \sum_{i=1,S_i\neq 0}^{i=|A|} log\left(\frac{1}{\sqrt{2\pi}\mathbf{D}_i^l}\right) - \frac{(S_i-\mathbf{M}_i^l)^2}{2\mathbf{D}_i^{l\,2}} \tag{12}$$

3.3.3 Resampling

With the estimate of the Perceptual Model $P(S|l)$ and the Motion Model $P(l_t|l_{t-1}, u_{t-1})$ of the Particle Filter, the localization *belief* of the robot can be recursively updated. But in the meantime, low weight particles must be replaced to ensure the convergence of the set.

Algorithm 1 resample($\mathbb{P}$)

```
1: for $i \leftarrow 1..|\mathbb{P}|$ do
2:    if $w_i < min_w$ then
3:       $w_i \leftarrow 0$
4:    end if
5:    $sum_w \leftarrow sum_w + w_i$
6: end for
7: for $i \leftarrow 1..|\mathbb{P}|$ do
8:    $r \leftarrow$ random.uniform($sum_w$)
9:    $cum_w \leftarrow 0$
10:   for $j \leftarrow 1..|\mathbb{P}|$ do
11:      $k \leftarrow j$
12:      $cum_w \leftarrow cum_w + w_j$
13:      if $r \leq cum_w$ then
14:         break
15:      end if
16:   end for
17:   $p_i \leftarrow p_k$
18: end for
```

The *resampling* step consists of drawing new particles after the *update* step. As result, part of particles will be duplicated, and some rejected - the higher the weight, the greater the chance that the particle will be drawn several times. Resampling can be compared with a probabilistic implementation of Darwin's theory, which relates to adaptation by natural selection. In this case, the "adjustment" is the particle weight. The same behavior can be seen in Genetic Algorithms (GAs) where the fitness function plays this role.

Our implementation of a Particle Filter uses a resampling algorithm (Algorithm 1) based on the Multinomial Resampling method [33], also called Simple Random Resampling. Instead of resampling only N particles of the particle set $\mathbb{P}$, it is able to replace the entire set. In addition to this, particles

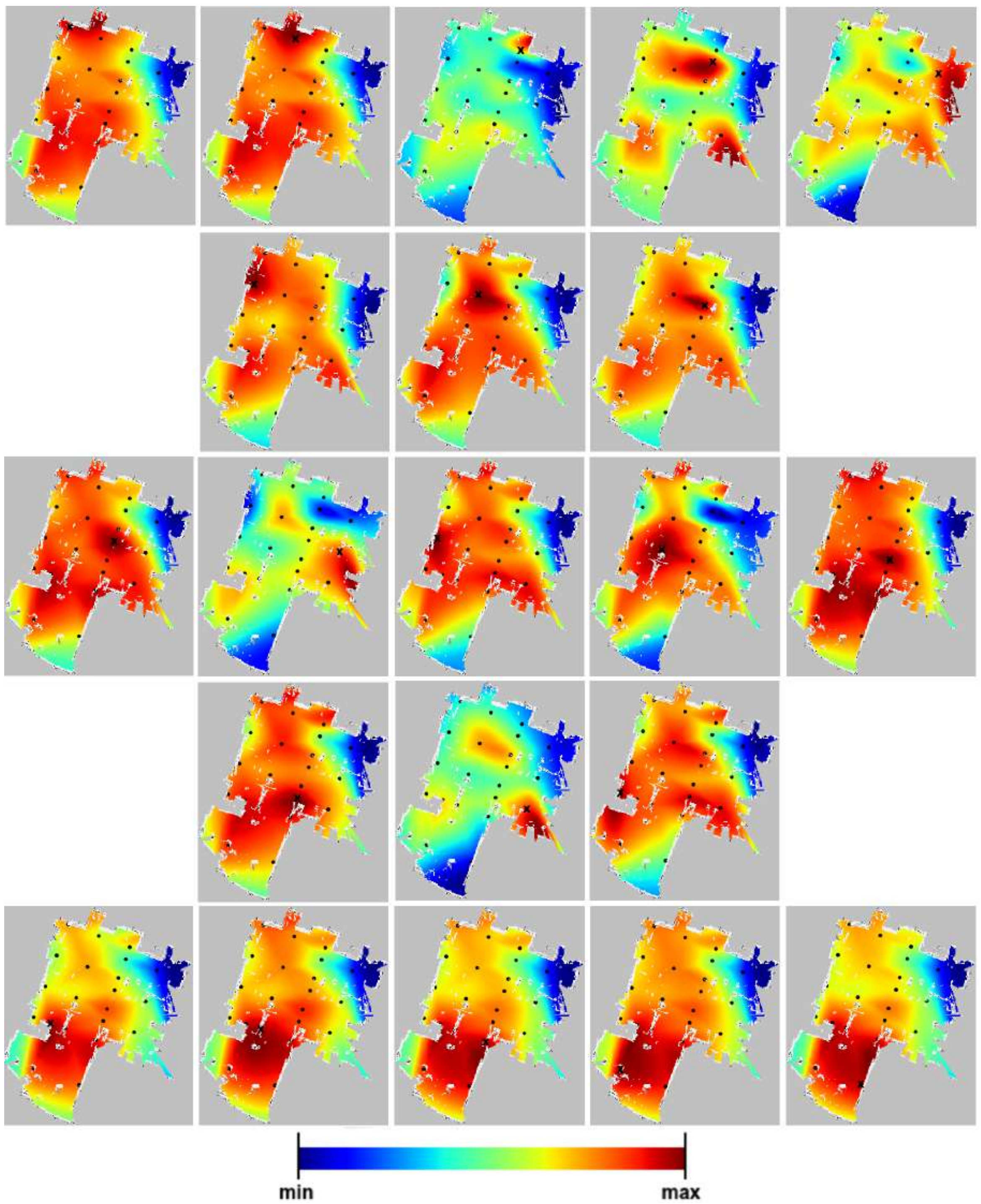


Figure 7: **WiFi Heat Map.** The discrete point where the robot observed the signals is marked with a cross, and the other discrete points are marked with regular dots. The probability values were calculated for all discrete points $p_i \in \mathbf{P}$ based on the observation of the point marked with a cross, and the values for the rest of the map were estimated by Bicubic Interpolation, which is more accurate than other interpolation techniques but takes more time to process. The values are color-coded, varying from the minimum to the maximum values found.

with weight less than a threshold min_w are forced to be resampled without chance of being duplicated. Using an uniform distribution based on the particle weights, the algorithm randomly chooses a particle to replace another. This process is similar to the Roulette Wheel on GA.

3.3.4 Infer Location

So far, the robot has only hypotheses about its location, represented by the particle set $\mathbb{P}$. To perform tasks like navigating to some point on the map, the robot needs to specify its current location as a single position (x,y,θ). Therefore, to infer the current position of the robot, the sum of the particle weights is calculated for each cell of the WiFi grid map. Then, using the cell with highest weight, its center of mass is computed based on its particles (Equations 13 14). The orientation θ is given by odometry data, which uses Kinect, gyroscope, wheel encoders, and others devices to correct the robot direction.

$$x = \frac{\sum_{i=1}^{i=|\mathbb{P}_{bestCell}|}(x_i \times w_i)}{\sum_{i=1}^{i=|\mathbb{P}_{bestCell}|} w_i} \tag{13}$$

$$y = \frac{\sum_{i=1}^{i=|\mathbb{P}_{bestCell}|}(y_i \times w_i)}{\sum_{i=1}^{i=|\mathbb{P}_{bestCell}|} w_i} \tag{14}$$

4 Kinect & WiFi Localization

Kinect localization is the most popular technique in robotics today. However, within indoor human environments, objects can move or be moved, which changes the internal map of the robot. Also, there might be many similar parts on the map generating ambiguous observations for the localization algorithm, as depicted by Figure 8. For these reasons, many robotic platforms that use Kinect localization needs human intervention to set the initial pose of the robot once it is turned on or when it is completely lost in the environment.

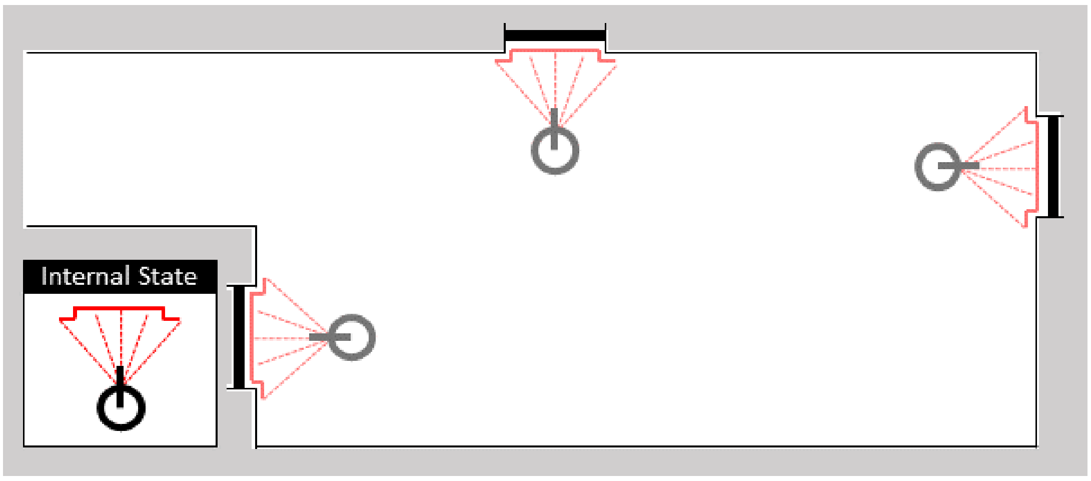


Figure 8: **Kinect localization.** The robot captures a very distinct observation, which looks like the shape of a door one meter ahead. Despite the precision, there are three options in different parts of the map where this observation fits.

WiFi localization is also a suitable approach for indoor human environments, as it is applicable anywhere there is a dense WiFi network. It is not costly either, since many robotic platforms have onboard computers with WiFi data card or WiFi dongle capable of capturing WiFi signals. But the minimum error of WiFi Localization can be greater than that of Kinect due to noise in the working Radio Frequency of the Access Points. Therefore, the estimates of the robot's position are never concentrated in a single spot of the map, as illustrated by Figure 9.

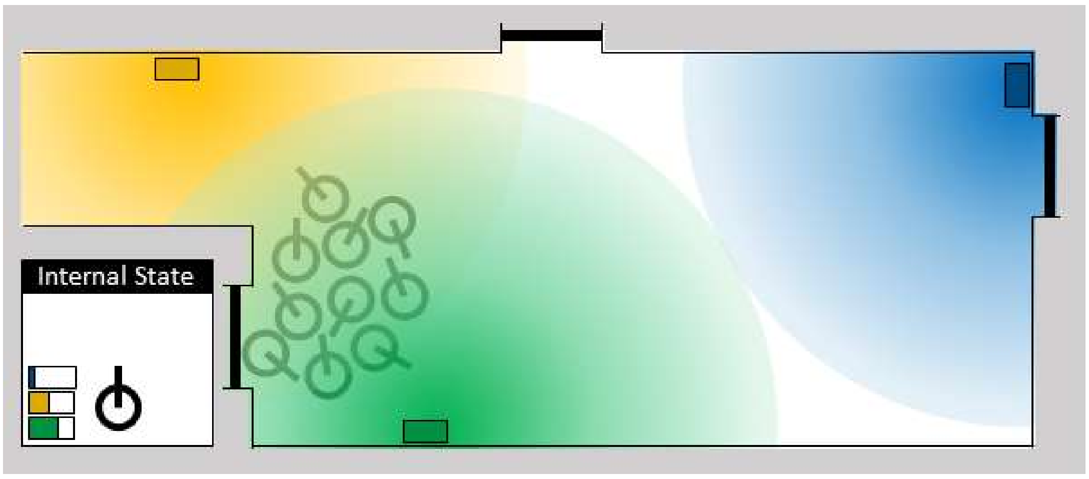


Figure 9: **WiFi localization.** The robot can listen to three APs (green, blue, and yellow) positioned at distant places along the environment. This time, the robot is not confused by observations from different parts of the environment. The main problem is that, the WiFi signals are not very distinguished locally.

Our approach aims at combining the benefits of both localization techniques. The general idea is to use the WiFi approach for global localization. Then, the Kinect approach estimates the current position (x, y, θ) of the robot considering the region determined by WiFi localization (Figure 10). Since Kinect hypotheses are bounded by those of WiFi, our algorithm uses thresholds to check whether the robot is lost, and recovers its position autonomously without human intervention using the global estimate given by the WiFi localization.

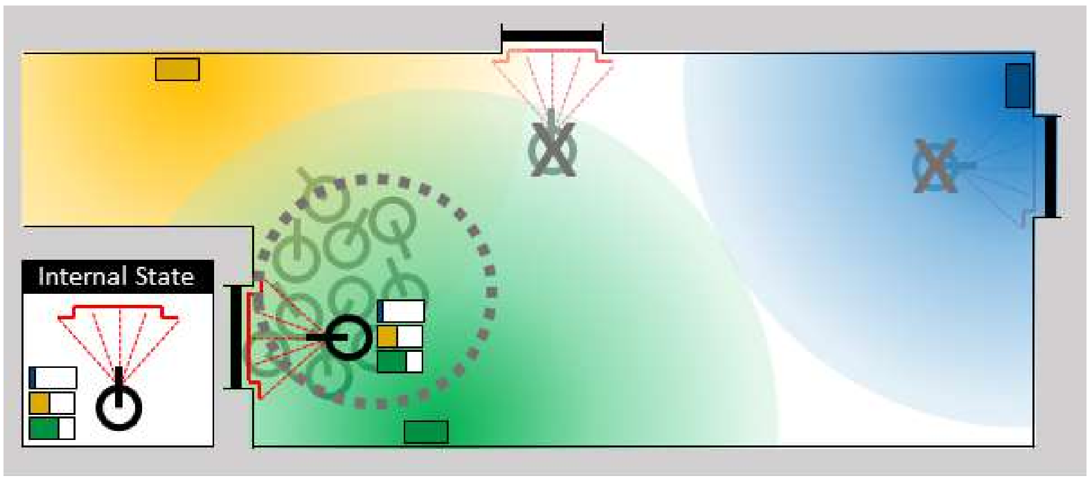


Figure 10: **Kinect & WiFi Localization.** The robot rejects hypotheses of Kinect localization that do not belong to the area delimited by WiFi localization, reducing the uncertainty about its position. The remaining hypotheses help to estimate the correct position of the robot in that area.

The proposed localization algorithm uses two Particle Filters (PFs), one for each device. The reason why we use two separate PFs instead of merging their observations in a single PF, as in [34], is due to the fact that WiFi particles should not be close to each other in order to compensate for the WiFi noise. On the other hand, Kinect particles should be able to converge almost to the same spot as Kinect observations are not noisy locally.

In addition to PF, another technique widely applied to robot localization is the Kalman Filter (KF). In this case, the localization is restrict to a Gaussian density assumption, not discrete as PF, which raises difficulties to the process of self-recovering the robot's position. Therefore, we have not considered traditional methods other than PF.

One of the most relevant novelties introduced by this approach is to separate global and local estimation, so one can support the other. Usually, localization algorithms try to merge data observation to provide a single estimation, specially when KF is used. But what we have noticed is that the uncertainty behaves differently on each type of observation, such as Kinect point clouds and WiFi signals, so they should be apart and have their own estimation.

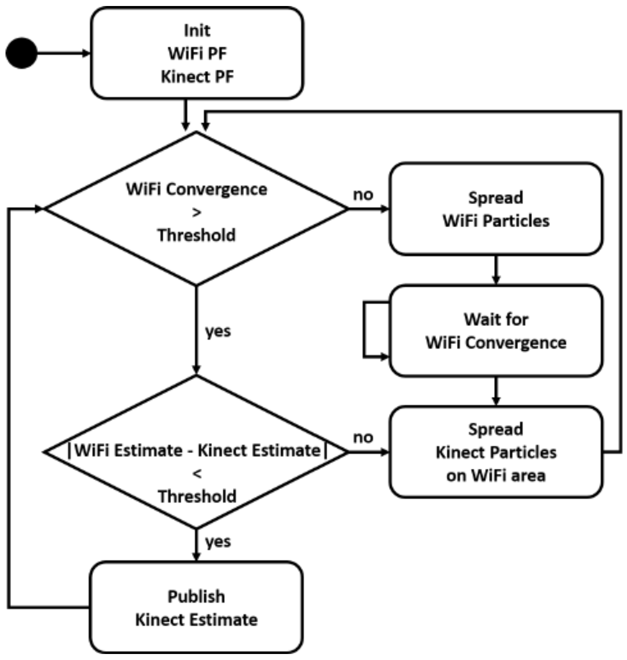

Init
WiFi PF
Kinect PF
WiFi Convergence > Threshold
no
Spread WiFi Particles
yes
Wait for WiFi Convergence
|WiFi Estimate - Kinect Estimate| < Threshold
no
Spread Kinect Particles on WiFi area
yes
Publish Kinect Estimate

Figure 11: **Flowchart.** Localization and Recovering Process.

Flowchart of Figure 11 summarizes the localization algorithm. After the initialization of both Particle Filters, the main loop starts by checking the convergence of WiFi Particles. If it is not greater than a threshold, the WiFi particles are spread along the environment in order to recover the global estimate. Once WiFi particles reach a certain convergence, Kinect particles are spread into the same area of WiFi particles to retrieve the local estimate. However, in case of WiFi particles are close to each other, the estimates of WiFi and Kinect are compared. If they are distant, Kinect particles are spread into the same area of WiFi particles, otherwise, the Kinect estimate is returned as the current position of the robot.

5 Experiments and Results

The localization experiments were carried out using a Turtlebot-2 in our laboratory. These experiments aim to evaluate the localization approaches addressed in this paper, namely: Kinect localization, WiFi localization, and Kinect & WiFi localization.

Setting up an experiment on a real environment is challenging. In our case, we had to ensure that our environment would not be changed during the mapping phase, otherwise, it could impact the localization accuracy. This means that physical objects must remain at the same position for Kinect mapping, and the WiFi management should keep channels, networks and IP addresses the same for WiFi mapping. After that, it is expected that the environment changes a little, as people start using it, but localization algorithms are able to handle this dynamics.

Although we choose to not run our first experiments on a public area, in order to avoid mapping issues and keep the same conditions for several trials, our laboratory represents a regular working area of approximately $47m^2$ with tables, chairs, cabinets and equipment used by ten people daily.

5.1 Localization error during navigation

The first experiment compares the localization accuracy of the approaches during the navigation on a fixed path crossing the laboratory, as shown by Figure 12.

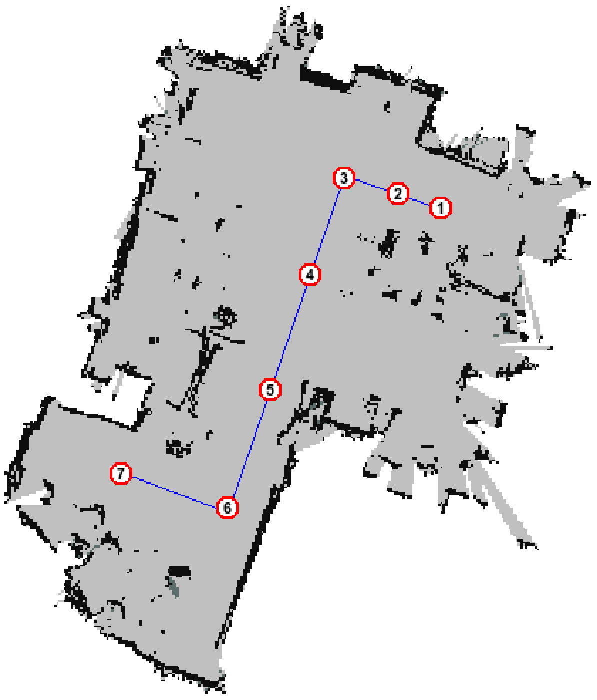


Figure 12: **Ground Truth Path.** Seven checkpoints positioned along the path were used to measure the error between the actual and the estimated location given by each approach.

Figure 13 shows a trial of our experiment. The left-hand column of the figure illustrates the original path performed by the robot. The next column represents the execution with Kinect localization. The third column shows the execution using WiFi localization. Finally, the last column depicts the execution using the Kinect & WiFi approach.

The first frame, on top, shows the initialization of the experiment where all particles are spread across the map. The second frame depicts the beginning of each approach and their first convergences. The reason why Kinect particles stay almost static in the same places is because the robot is not moving yet, and the Kinect is receiving the same observation over and over again. On the other hand, the WiFi algorithm keeps receiving different observations, which enables the updating of particle weights. Then, the robot makes a turn on its axis and starts the path. The next frame shows the robot in the middle of the path. There, the robot is completely lost in the Kinect approach. This happens when the robot is not well localized before starting the navigation. To overcome this issue, many Kinect approaches need human intervention to set the initial pose of the robot. At this point, the Kinect & WiFi localization is slightly better than the WiFi localization. The last frame, at the bottom, illustrates the end of the path, where the estimate of Kinect & WiFi localization is close to the actual location of the robot. Although the WiFi localization seems to yield the same result, its estimate is a little further from the actual location, and it could worsen if the path were longer.

The robot performed the path 10 times for each approach. Figure 14 shows a plot of the error in the localization at each checkpoint of the path. Figure 15 emphasizes the results by showing only the mean error of all approaches together. As depicted by the red line in the charts, on both figures, the average error for Kinect & WiFi localization was less than 50cm almost all the time.

Our work focuses on robot localization without any human intervention. The reason why the Kinect approach had poor results is because the initial position was unknown by the robot during the tests. Although the use of Kinect in conjunction with AMCL is very common in robotics, either the robot must

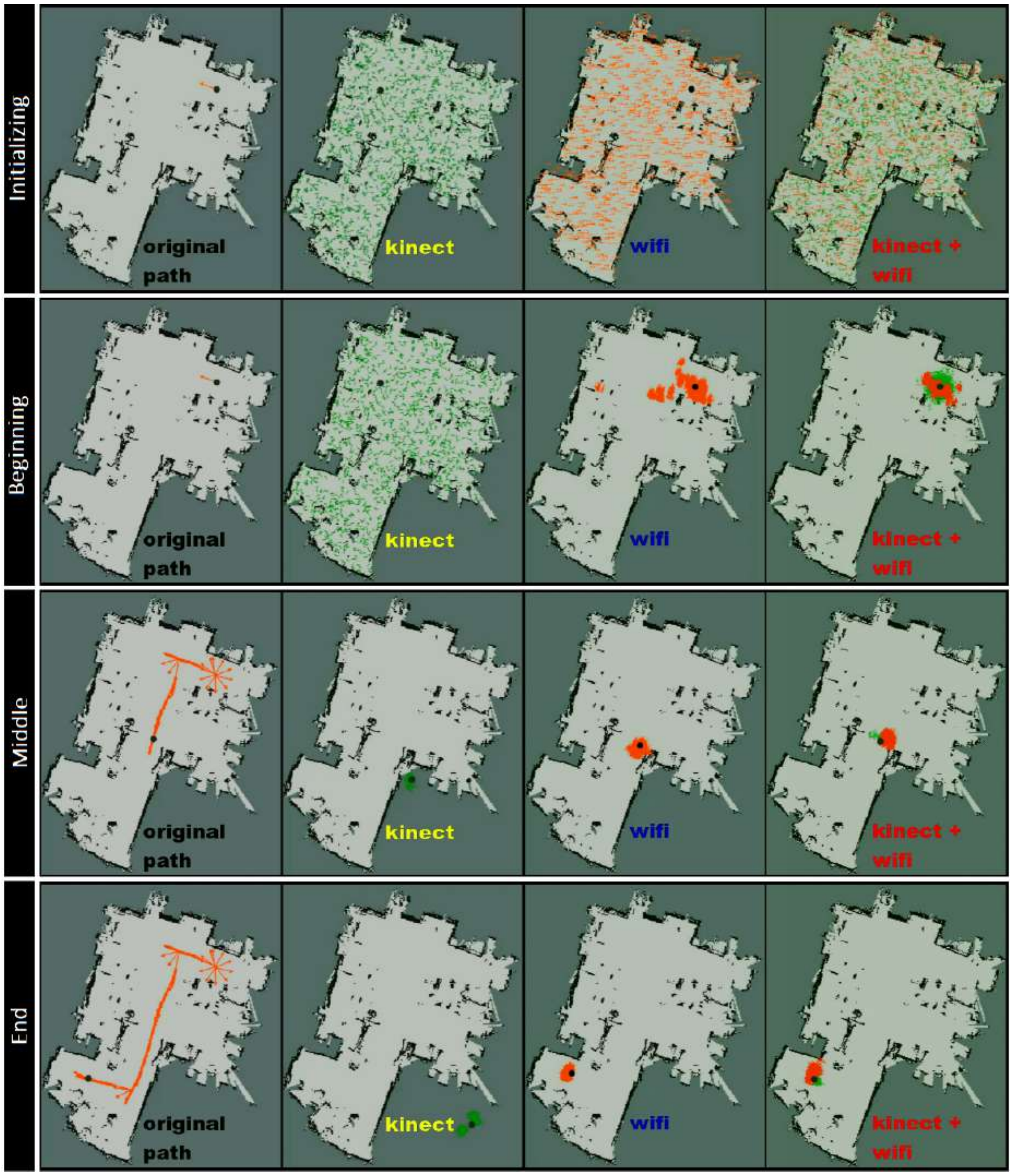


Figure 13: **Trial example on the path.** In this picture, black dots represent the robot's estimate position given by each technique, small green arrows are Kinect particles, small red arrows are WiFi particles and big red arrows on the left pictures compose the real path executed by the robot.

be turned on at some predefined location on the map or its initial location must be set by a human so the particles can start all together avoiding a wrong convergence right in the beginning. The Kinect & WiFi outperformed the other approaches as our WiFi global localization plays this role. It automatically initializes and reset Kinect particles regarding the WiFi positioning every time the robot is turned on or lost.

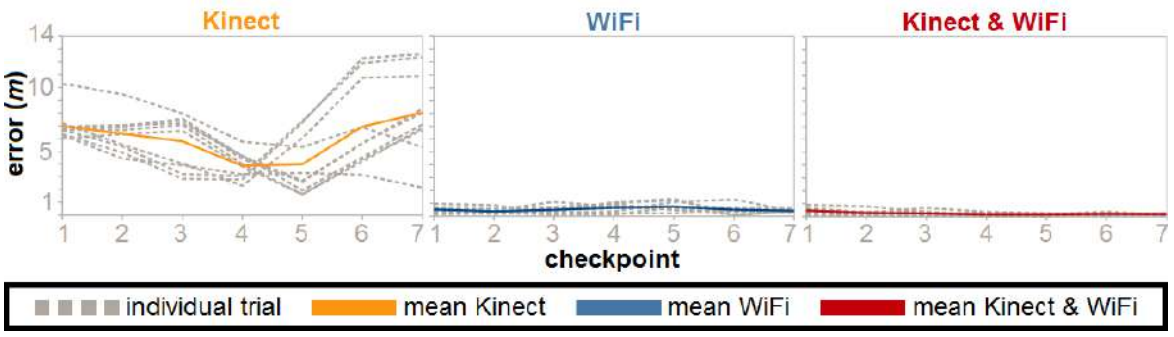


Figure 14: **Localization error along the path.** The dashed lines represent the error for individual trials and the colored ones represent the mean error.

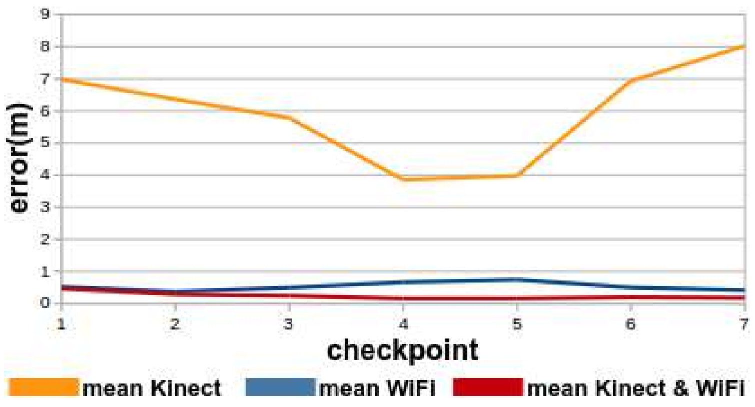


Figure 15: **Mean error.** Colored lines highlight the mean error of all approaches.

5.2 Initialization and the Kidnapped Robot Problem

The Kinect & WiFi localization algorithm (Figure 11) enables the robot to automatically set its pose in case it is lost. In robotics, this problem is known as the Kidnapped Robot Problem, which commonly refers to a situation where an autonomous robot in operation is carried to an arbitrary location. The Kidnapped Robot Problem creates significant issues with the robot's localization system, and only a subset of localization algorithms can successfully deal with the uncertainty created; it is commonly used to test a robot's ability to recover from catastrophic localization failures.

In the next experiment, this ability was tested in our robot by performing a sequence of kidnappings. Figure 16 illustrates this process. First, the robot was positioned at some place in the laboratory and its location was properly set. Then, the robot was taken to somewhere else. Despite receiving no new odometry data, the robot senses that its position has changed by perceiving different WiFi observations. In such case, the weights of its WiFi particles start decreasing, and when they reach a threshold, the algorithm randomly initializes the particles again across the map. When the WiFi particles converge, the algorithm uses the estimated location given by the WiFi Particle Filter to reset the Kinect particles. Then, the Kinect Particle Filter uses Kinect observations to update particles at right places and recover the robot's position.

In order to evaluate the uncertainty concerning localization and how long this process takes, 50 kidnappings were performed. The chart in Figure 17 shows the mean localization error for 5 seconds of recovering. In 2 seconds, the error is less than 1 meter. To keep reducing the error after recovering, the robot needs to navigate a little so the Kinect particles can converge.

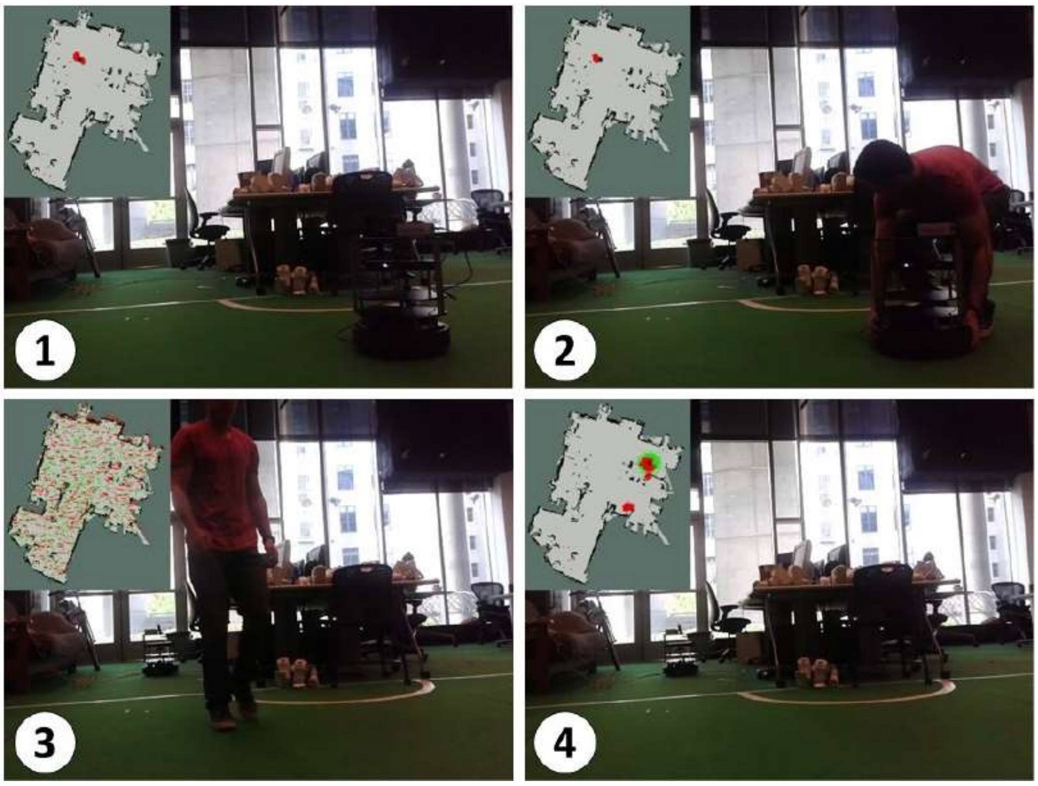


Figure 16: **Robot Kidnapping Experiment.** The red arrows represent WiFi particles, green arrows represent Kinect particles, and the black dot is the current estimate of the robot's position. The sequence of pictures 1-4 shows the robot being tacked away from its initial known location to other unknown location where it was able to recover its position automatically.

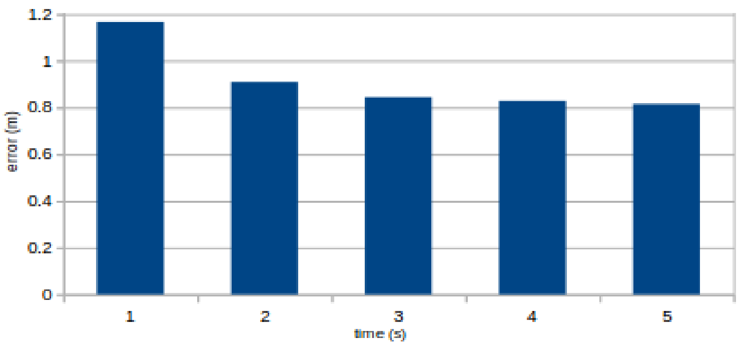


Figure 17: **Recovery Time.** The horizontal axis of the chart displays the time in seconds and the vertical axis shows the average error in meters.

5.3 WiFi localization accuracy vs. number of APs

While the robot is navigating, it is reasonable to expect that it would drop WiFi signal strength readings from some Access Points. Therefore, it is difficult to have a perfect observation that contains values for all APs. In this last experiment, the impact of dropped signals on the WiFi Localization accuracy is investigated. During navigation along another path the number of APs that could be used in the localization was limited. Thus, every time an observation was required by the WiFi localization algorithm, we first randomly removed some APs from the observation vector.

Figure 18 shows the behavior of the particles using different amounts of APs at the beginning of the path, a few seconds after the initialization. By looking at the picture, it is noteworthy that the convergence of the particles is better when more APs are used.

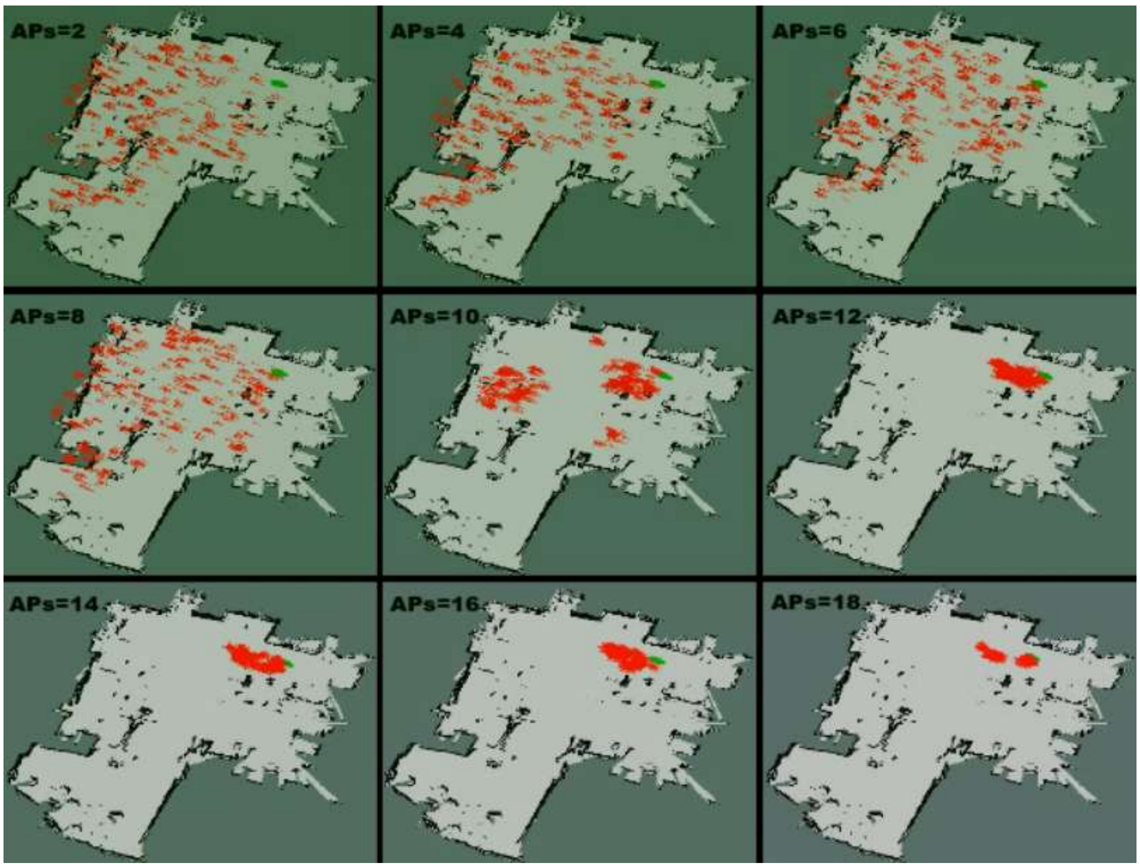


Figure 18: **Accuracy of WiFi localization.** This figure illustrates a trial of the experiment using 2, 4, 6, 8, 10, 12, 14, 16, and 18 APs. The test starts by randomly initializing WiFi particles across the map. The red arrows represent WiFi particles and the green arrows represent the accurate estimate of the robot's position.

The chart in Figure 19 presents the results of this test. Although our WiFi map is composed by 21 Access Points, it is not possible to listen to all of them everywhere all the time, specially in 1 second of observation. But even with 12 Access Points, the mean error in localization is close to 1 meter.

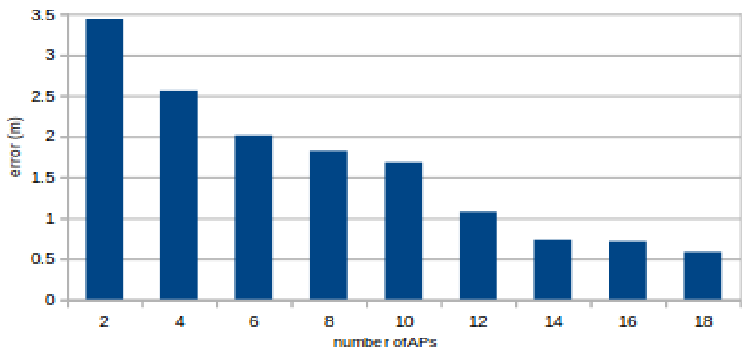


Figure 19: **Localization error vs. number of AP.** The horizontal axis of the chart displays the number of Access Points used in the localization and the vertical axis shows the average error along the path.

6 Conclusion

In this paper, the indoor localization for autonomous robots using low-cost devices has been investigated. Two devices were used, a Kinect camera and a WiFi dongle.

An approach was presented that uses a Kinect device to map and localize a robot in an indoor area. To map the area, the GMapping algorithm was used to treat depth images captured from the Kinect and build an occupancy grid. Then, with this grid map, the AMCL algorithm estimates the robot's position in real time by comparing its observations against the map.

A novel WiFi localization approach was introduced. We proposed the concept of irregular grids to collect WiFi signal strengths at discrete points in the area to generate a WiFi map. Then, our localization algorithm, based on the Particle Filter framework, estimates the robot's position continuously along the map by the means of different interpolation techniques and the center of mass of the most promising cell of the irregular grid.

Considering that the Kinect approach is more accurate locally and the WiFi approach gives a global location of the robot on the map, we further introduced a localization algorithm that combines both approaches using separate Particle Filters. This algorithm also enables the robot to autonomously recover its position in case it gets lost.

Different experiments were carried out. The first experiment aimed at analyzing the localization error at some checkpoints along a fixed path during navigation in order to compare all three approaches: Kinect localization, WiFi localization, and Kinect & WiFi Localization. In the next experiment, the localization accuracy and recovery time were measured when the robot was "kidnapped". For the final experiment, the WiFi localization error was taken using different amounts of Access Points. These experiments showed that the proposed algorithms are very much appropriate for localization in indoor human environments, yielding acceptable errors on the estimates.

During the tests, we noticed a positioning error of 50cm-60cm, which is great considering the cost of our devices (Microsoft Kinect v1 and USB WiFi dongle). This result could be even better depending on the environment. For example, the indoor positioning within small and narrow areas, such as corridors and rooms, has a better precision, as walls and corners yield more featured observations to the Kinect localization, different from large indoor open areas, like a lobby or a laboratory. We have not found any related work using low cost devices in large indoor open areas like we did. Either way, more experiments can be carried out in different types of environment in order to show more about the contributions of our work.

We believe that other devices could be tested, such as LiDAR laser scanner or Kinect v2, in order to reduce the error. Furthermore, new methods for WiFi localization using Angle of Arrival and Time of Arrival should be evaluated, as they seem to perform better than the traditional Received Signal Strength approach.

We had some problems with WiFi localization regarding the network management. Usually, the settings of some APs are switched for security and performance purposes, which changes the WiFi map of the environment and increases the error. Also, furniture that are moved frequently in indoor areas might change the laser map used by the Kinect localization. Therefore, researches on how to fix maps will be extremely important to the indoor localization field.

References

[1] Z. Farid, R. Nordin, M. Ismail, Recent advances in wireless indoor localization techniques and system, Journal of Computer Networks and Communications 2013.

[2] R. Mautz, Indoor positioning technologies, Ph.D. thesis, ETH Zurich (2012). `doi:10.3929/ethz-a-007313554`.

[3] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, J. Fangmeyer, Evolution of indoor positioning technologies: A survey, Journal of Sensors 2017.

[4] S. P. Engelson, D. V. McDermott, Error correction in mobile robot map learning, in: Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on, IEEE, 1992, pp. 2555–2560.

[5] D. Kortenkamp, T. Weymouth, Topological mapping for mobile robots using a combination of sonar and vision sensing, in: AAAI, Vol. 94, 1994, pp. 979–984.

[6] B. Kuipers, Y.-T. Byun, A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, Robotics and autonomous systems 8 (1) (1991) 47–63.

[7] J. Leonard, H. Durrant-Whyte, I. J. Cox, Dynamic map building for autonomous mobile robot, in: Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on, IEEE, 1990, pp. 89–96.

[8] H. F. Durrant-Whyte, Sensor models and multisensor integration, The international journal of robotics research 7 (6) (1988) 97–113.

[9] H. P. Moravec, Sensor fusion in certainty grids for mobile robots, AI magazine 9 (2) (1988) 61.

[10] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, S. Thrun, The mobile robot rhino, Ai Magazine 16 (2) (1995) 31.

[11] S. Thrun, A. Bücken, Learning maps for indoor mobile robot navigation., Tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE (1996).

[12] G. Weiß, E. v. Puttkamer, A map based on laserscans without geometric interpretation, in: Intelligent Autonomous Systems, Vol. 4, IOS Press, 1995, pp. 403–407.

[13] J.-S. Gutmann, C. Schlegel, Amos: Comparison of scan matching approaches for self-localization in indoor environments, in: Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on, IEEE, 1996, pp. 61–67.

[14] P. S. Maybeck, Stochastic models, estimation, and control, Vol. 3, Academic press, 1982.

[15] F. Dellaert, D. Fox, W. Burgard, S. Thrun, Monte carlo localization for mobile robots, in: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, Vol. 2, IEEE, 1999, pp. 1322–1328.

[16] J. Biswas, M. Veloso , Depth camera based indoor mobile robot localization and navigation, in: Proceedings of IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 1697–1702.

[17] J. Biswas, M. Veloso , Episodic non-markov localization: Reasoning about short-term and long-term features, in: International Conference on Robotics and Automation (ICRA) 2014, 2014.

[18] S. Lee, J.-B. Song, Mobile robot localization using infrared light reflecting landmarks, in: Control, Automation and Systems, 2007. ICCAS'07. International Conference on, IEEE, 2007, pp. 674–677.

[19] J. Biswas, M. Veloso, Multi-sensor mobile robot localization for diverse environments, in: Robot Soccer World Cup, Springer, 2013, pp. 468–479.

[20] J. Biswas, M. Veloso, Wifi localization and navigation for autonomous indoor mobile robots, in: 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 4379–4384. `doi:10.1109/ROBOT.2010.5509842`.

[21] P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments, in: In the 12th International Symposium on Experimental Robotics (ISER, Citeseer, 2010.

[22] J. Biswas, M. Veloso, Depth camera based indoor mobile robot localization and navigation, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 2012, pp. 1697–1702.

[23] M. Jalobeanu, G. Shirakyan, G. Parent, H. Kikkeri, B. Peasley, A. Feniello, Reliable kinect-based navigation in large indoor environments, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 495–502.

[24] H. Cheng, H. Chen, Y. Liu, Topological indoor localization and navigation for autonomous mobile robot, IEEE Transactions on Automation Science and Engineering 12 (2) (2015) 729–738.

[25] K. Qian, Z. Chen, X. Ma, B. Zhou, Mobile robot navigation in unknown corridors using line and dense features of point clouds, in: Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE, IEEE, 2015, pp. 001831–001836.

[26] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with rao-blackwellized particle filters, IEEE transactions on Robotics 23 (1) (2007) 34–46.

[27] D. Fox, Kld-sampling: Adaptive particle filters, in: Advances in neural information processing systems, 2001, pp. 713–720.

[28] S. Thrun, et al., Robotic mapping: A survey, Exploring artificial intelligence in the new millennium 1 (2002) 1–35.

[29] C. Stachniss, W. Burgard, Particle filters for robot navigation, Foundations and Trends in Robotics 3 (4) (2014) 211–282.

[30] P. W. Mirowski, T. K. Ho, P. Whiting, et al., Building optimal radio-frequency signal maps., in: ICPR, 2014, pp. 978–983.

[31] B.-F. Wu, C.-L. Jen, Particle-filter-based radio localization for mobile robots in the environments with low-density wlan aps, IEEE Transactions on Industrial Electronics 61 (12) (2014) 6860–6870.

[32] R. Palaniappan, P. Mirowski, T. K. Ho, H. Steck, P. Whiting, M. MacDonald, Autonomous rf surveying robot for indoor localization and tracking, in: International conference on indoor positioning and indoor navigation, 2011.

[33] R. Douc, Comparison of resampling schemes for particle filtering, in: In 4th International Symposium on Image and Signal Processing and Analysis (ISPA, 2005, pp. 64–69.

[34] A. Canedo-Rodríguez, V. Álvarez Santos, C. Regueiro, R. Iglesias, S. Barro, J. Presedo, Particle filter robot localisation through robust fusion of laser, wifi, compass, and a network of external cameras, Inf. Fusion 27 (C) (2016) 170–188. `doi:10.1016/j.inffus.2015.03.006`.
URL `http://dx.doi.org/10.1016/j.inffus.2015.03.006`