PIL: A new algorithm for mobile robot localization
Philippe Lambert, Lotfi Jaïem, Lionel Lapierre, Didier Crestani

To cite this version:
Philippe Lambert, Lotfi Jaïem, Lionel Lapierre, Didier Crestani. PIL: A new algorithm for mobile robot localization. 2020. hal-02494512

HAL Id: hal-02494512
https://hal.archives-ouvertes.fr/hal-02494512
Preprint submitted on 28 Feb 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
PIL: A new algorithm for mobile robot localization

P. Lambert, L. Jaiem, L. Lapierre, and D. Crestani

University of Montpellier, CNRS, LIRMM
860 rue de Saint Priest, Montpellier, France
plambert,lapierre,crestani@lirmm.fr
IADYS, Roquefort-la-Bédoule, France
lotfi.jaiem@iadys.com

Abstract. This paper proposes a novel localization algorithm (Pseudo Image Localization: PIL) using a grid able to solve global localization and robot kidnapping issues. The proposed approach has 2 steps. The off-line step is a learning phase during which a simulated robot completely learns its environment at attitude $\theta = 0$. Several orientation-free level functions are defined. These functions are the different filters on which the on-line phase will rely to select the position candidates. During the on-line phase the robot smartly uses the massive data produced during the off-line phase to recover first its position and second its orientation. The paper proposes the theoretical development of this localization approach, some simulation examples and an experimental validation carried out with a Pioneer 3DX robot moving in a predefined environment.

Keywords: Localization · Grid localization · Mobile robotics · Pose tracking · Global Localization · Kidnapping · Deterministic

1 Introduction

1.1 The problem

Localization is a fundamental issue for achieving autonomous mobile robot navigation. Localization is the problem of determining the position and orientation of a robot $\xi_t = [x_t, y_t, \theta_t]$ at time $t$, given a map of the environment and sensors data [1–3]. Localization can be roughly divided into three main sub-problems:

- Position Tracking: Time continuous localization on a path on which the initial position $\xi_0 = [x_0, y_0, \theta_0]$ is known.
- Global Localization: Time continuous localization with unknown initial pose.
- Kidnapping: During its travel, the robot is kidnapped and brought to another place. This "academic" case aims to address the very realistic case of potential failures recovery.

* Supported by Labex Numev, Isite Muse and Feder funds.
In order to address these three sub-problems, the literature has proposed several approaches: Extended Kalman Filters, Monte Carlo Localization, Grid approaches and Miscellaneous approaches.

In a way, the three first approaches are different implementations of Bayes filtering. The Bayes filter technique provides a powerful statistical tool to understand and solve robot localization problems by recursively calculating the position belief distribution 

\[ Bel(\xi_t) = p(\xi_t | a, z, m) \]

from control data \(a\), measurement data \(z\) and a map \(m\). This belief function is the probability density of the position (3 random variables for a ground robot) to lie in the elementary hyper-cylinder \([\xi_t, \xi_t + d\xi_t]\).

The Extended Kalman Filter approach states that the robot movements and the measurements are Markov processes. The estimated position \(\hat{\xi}_t = [\hat{x}_t, \hat{y}_t, \hat{\theta}_t]\) results from the fusion process of the measures and the model. It is a straightforward powerful algorithm especially for the Position Tracking problem but inapplicable to the Global Localization problem in its earliest stage. This limitation is somehow overcome by the use of the multi-hypothesis Kalman filter proposed in [1, 4]. The Monte Carlo Localization is based on a particle filter that represents the posterior belief of the robot pose by a set of weighted samples (particles) distributed according to sensor measurements [5,9].

The disadvantage of this approach is that it bears heavy on-line computational burden when the number of particles is high. However, it can solve the Global Localization and Kidnapping problems.

Grid localization approximates the posterior using a histogram filter over a grid decomposition of the pose space [10, 11]. The disadvantage of this approach is that it also bears heavy on-line computational burden when the number of grid cells is high. However, it can also solve the Global Localization and Kidnapping problems.

The Miscellaneous approaches groups the algorithms that do not fall in the three first approaches. For instance, the interval approach [6] is a deterministic point of view that solves the localization problem using interval algebra.

1.2 Outline of the paper

This paper use and expande the methodology presented in [12] to adress localization in a novel approach called Pseudo Image Localization (PIL) for the localization of ground mobile robots using range sensors and raw data. It can be classified into the Miscellaneous approaches group, but has some common properties with grid algorithms. Used alone, it provides another way for solving the Global Localization and the Kidnapping problems.

Section 2 describes the general algorithm. The localization problem can be seen as a virtual image analysis. Distances are translated into a set of pseudo images representing orientation-free level functions. Section 3 presents simulated examples of kidnaping senari. In section 4 a real experiment is carried out with a Pioneer 3DX equipped with laser range finders. finally section 5 discusses the proposed approach and concludes this paper.
2 Level-Localization

2.1 The robot

A ground mobile robot moves in a structured a priori known environment. It is equipped with a full-range sensor giving \( N_s \) distances uniformly distributed all around the robot. Its pose is a vector \( \xi = [x \ y \ \theta]^T \). The Global Localization consists in determining an estimate \( \hat{\xi} = [\hat{x} \ \hat{y} \ \hat{\theta}]^T \) of its real pose without any odometry information. Figure 1-left shows a simulated robot equipped with \( N_s = 57 \) sensors moving in a \( 20m \times 20m \) environment. This simulation experiments have been carried out in order to validate the proposed algorithm.

2.2 The Learning phase (off-line)

The map is build accordingly to a given desired resolution. Considering the size of the map this give us a grid of dimensions \( L_{Max} \times C_{Max} \) to be able co cover the all space. For instance, if the desired resolution for a \( 20m \times 20m \) square environment is \( r = 20cm \), we will choose \( L_{Max} = C_{Max} = 100 \). In this grid some tiles may be invalid if their center is outside of the considered environment or inside an obstacle. A pseudo-images corresponds to what a unique robot sensor sees from all tiles of the map.

During the learning phase, a simulated robot moves from the center of all of the valid tiles, keeping its orientation at \( \theta = 0 \). At each of those positions, it builds \( N_s \) points (pixels) of \( N_s \) level pseudo-images of dimensions \( L_{Max} \times C_{Max} \). The images have been chosen to cover the entire simulated valid robot space and all of this results from the homogeneous transformation:

\[
\begin{bmatrix} L \\ C \\ 1 \end{bmatrix}^T = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^T
\]

where \( L \) and \( C \) are the line and column coordinates in the image space, \( x \) and \( y \) the coordinates of the robot in its "real" simulated space and \( M \) the homogeneous transformation matrix between these two sets of coordinates. The intensity of the the \((L,C)\) pixel in image \( k \) is chosen in order to reflect the perceived distance \( D_s(L,C,k) \) of sensor \( k \) at point \((x,y)\). In order to deal with normalized images, its value is normalized as follows:

\[
I(L,C,k) = 1 - \frac{D_s(L,C,k)}{R}
\]

where \( R \) is the range of the sensors.

Considering the previous example dimensions, the learned map will be a set of \( N_s \) intensity pseudo images of dimensions \( 100 \times 100 \). Figure 1-center and right show what the robot has seen for respectively its sensors \( k = 21 \) and \( k = 42 \).

Clear zones correspond to short distances and dark zones to large distances. White zones correspond to \( D = 0 \) (the robot is inside an obstacle; tile invalid) and black zones to blind zones (the robots is too far from the walls to see them).
2.3 The Transformation phase (off-line)

Once the map has been learned, in order to localize (i.e. to find $\hat{P} = [\hat{x} \quad \hat{y} \quad \hat{\theta}]^T$), the map consisting in $N_s$ level pseudo-images has to be processed in order guide the research process in the map. For this, $N_{OLF}$ independences Orientation-free Level Functions (pseudo-images $I_n$) are build off-line. Here are some examples of these images:

- **Energy pseudo-image** ($I_1$): the image is obtained by summing the $N_s$ images (and dividing the sum by $N_s$ in order to keep normalization). See figure 2-left.
- **Percentage $pq\%$ pseudo-image** ($I_2$): the image is obtained by counting the number of sensors having value $p \leq I(:,;k) \leq q$ (and dividing the count by $N_s$ in order to keep normalization). See figure 2-right.

It is quite obvious that these images are independent of the robot orientation.
2.4 The \((x, y)\) Localization phase (on-line)

The real robot is placed in its environment (kidnapping at time \(t_0 = 0\)). The perceived distances \(D_r(\cdot)\) are transformed according to equation 2:

\[
I(L, C, k) = 1 - \frac{D_r(k)}{R}
\]

where now, \(L\) and \(C\) are unknown. The problem consists in inverting this equation (i.e. finding \(L\) and \(C\) given the set \(X(L, C) = \{I(L, C, k) \mid 1 \leq k \leq N_s\}\)).

The same transformations as before are applied to this set. For this, \(N_{OLF}\) orientation-free intensity values \(E_n\) are build on-line:

- **Energy**: \(E_1 = \frac{1}{N_s} \sum_{k=1}^{N_s} I(L, C, k)\)
- **Percentage pq\%**: \(E_2 = \frac{1}{N_s} \text{card}\{p \leq I(L, C, k) \leq q \mid 1 \leq k \leq N_s\}\)

Once these numbers computed, a resulting binary pseudo-image is produced by:

\[
I = \bigcap_{n=1}^{N_{OLF}} \alpha_n |I_n - E_n| < \sigma_n
\]

where \(\alpha_n\) is the weighting factor of each \(E_n\) and \(\sigma_n\) the arbitrary threshold corresponding to the accepted tolerance. The obtained pixels are the candidates \((L, C)\) that are translated back to the cartesian space:

\[
\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}^T = M^{-1} \begin{bmatrix} L \\ C \end{bmatrix}^T
\]

Figure 3-left shows the resulting image for the dual (Energy, pq\%) when the robot is situated at \(x = 1.8m, y = -0.8m\) in the reference frame.
2.5 The $\theta$ Localization phase (on-line)

For each of these candidates, we then extract the corresponding profile $D_s(L,C,:)$ (one point in each of the $N_s$ pseudo-images) and compare it with the real profile $D_r(\cdot)$ (see figure 3-right).

These two profiles are then correlated. The abscissa of the maximum of the correlation function gives the angular shift between the simulated robot that has learned the environment with $\theta = 0$ and $\hat{\theta}$ that localizes the robot. The chosen $\hat{\theta}$ is the one that maximizes the correlation $D_r(\cdot)$ vs $D_s(L,C,:)$.

At this step, an estimation of the localizer $\hat{P} = (\hat{x}, \hat{y}, \hat{\theta})^T$ has been obtained.

2.6 Summary

The PIL algorithm can be divided into two main parts (see figure 4): an off-line algorithm (Learning phase) and an on-line algorithm (Localization phase).

\begin{verbatim}
1: OFF-LINE: Learning phase
2: Environment description : set of polygons
3: Environment sweeping $\theta = 0$
4: for $x = x_{min}$ to $x_{max}$ do
5: for $y = y_{min}$ to $y_{max}$ do
6:   Move Robot to $(x,y)$
7:   Compute $[L,C,t]^T = M [x,y]^T$
8:   For each sensor $k$ compute $I(L,C,k)$ (eqn. 2)
9: end for
10: end for
11: Transformation phase
12: Compute the $N_{OLF}$ orientation-free pseudo-images $I_n$

1: ON-LINE: Localization phase at time $t$
2: Map description : $N_{OLF}$ pseudo-images $I_n$
3: Profile: Distances $D_s(\cdot)$
4: Move robot to unknown location $(x, y, \theta)$
5: Calculate the $N_{OLF}$ numbers $E_n$
6: Calculate $I = \prod_{i=1}^{n} \alpha_i |I_n - E_n| < \sigma_n$
7: Extract the couples $(L, C)$ such that $I(L,C) = 1$
8: for each couple $(L, C)$ do
9:   Calculate $(\hat{x}, \hat{y})$
10: Calculate the correlation $C_n = D_r(\cdot)$ vs $D_s(L,C,:)$
11: Compute the candidates set $\{(x_n, y_n, \theta_n, C_n)\}$
12: end for
13: Calculate $\hat{\theta}$ that maximizes this $C_n$
\end{verbatim}

Fig. 4. PIL algorithm: Left: off-line phase. Right: on-line phase.

2.7 Analysis

Used alone, the previous algorithm presents some properties and also some drawbacks. The next list presents these Properties.

- Global localization: Robot odometry is not taken into account in the algorithm. Therefore, PIL is inherently a Global Localizer.
- Kidnapping: As robot odometry is not taken into account in the algorithm, a kidnapping should produce no effect. As we will see later, in order to accelerate the whole process, a windowing filter is added to the initial algorithm. This filter can produce a delay after kidnapping.
- Deterministic localization: For a given map the same mesurement will always lead to the same localization.
The next list presents these **Drawbacks** and proposes some possible solutions.

- In all cases, the Learning phase is quite time consuming. This is not really a drawback since a robot moving in a known environment can be provided with the map obtained off-line before its mission. However this acquisition time can be lowered if we lower the learning precision.

- In a large map many candidate tiles can satisfy 4 creating the binary pseudo image identifying potential location tiles. It is possible to filter all the candidates that are not in a spatial window around the localized robot. This window size is chosen according to the confidence the robot has on its localization. To answer the *Kidnapping* problem, the window cover the whole map.

- The main drawback of the proposed approach is the fact that it is very sensitive to environment modifications. As we will seen in the next paragraph, if an obstacle has not been learned and is in the perception field of the robot, it will considerably alter the localization process. However, if the environment is enough stable (not too many unlearned obstacles), the algorithm will be able to localize most of the time.

This algorithm under is development name (SL2) is presented in a video [8]. It shows in a simulated environment the different steps of the PIL algorithm and some cases of kidnapping. This problem will be addressed in the next section.

### 3 Kidnapping Simulations

The PIL algorithm (4) has been tested on a simulated mobile robot evolving on a building floor. The programs have been developed with MATLAB on a 3.4GHz Intel Core i7 Mac processor. The robot is equipped with a full range laser finder (57 sensors with inter-angular distance of $6.3^\circ$). It travels at about $v = 0.9 m/s$. The environment is a $30m \times 11m$ floor with an unbalanced structure. This environment is represented by a virtual image of $152 \times 56 = 8512$ pixels which means that the elementary pixel represents a $20cm \times 20cm$ tiles. The off-line algorithm processes only 2156 pixels it is to say one 25% of the complete image. The Energy lists necessitate $720kB$ of memory.

#### 3.1 Kidnapping to discriminating location

In this experiment (Figure 5), the robot travels during $35s$ from its initial position to point 3. At time $t = 30.5s$ (a position between point 2 and point 3) it is kidnapped and put back to its initial position. It recovers from this kidnapping in only one computation step. Figure 6 shows the error between the real position and the position of the localized robot. The mean of this error is about $13cm$. 
3.2 Kidnapping to an ambiguous location

In this experiment, the robot travels from its initial position to point 3. At time $t = 10s$ (a position between point 1 and point 2) it is kidnaped and put back to its initial position and recovers as quickly as previously. It is kidnaped again at time $t = 30s$ and put back to point 2. Because of the angular ambiguity in the corridor, the robot is mistaken on its orientation. Between $t = 30s$ and $t = 36s$, it believes it has an orientation $\theta_{LOC} = \pi$. It cannot instantaneously recover its orientation but only near the point 3 when the geometrical ambiguity disappears. Figure 7 shows the angular error during this travel.

4 Experimental results

4.1 Context Description

The experimental context can be described by the following points:

- The mobile platform is a unicycle type Pioneer 3DX (Figure 8-Top-left) equipped with several sensors. Among them two URG-04 LX Hokuyo lasers allow for full horizontal scanning of the environment. Their data are merged to cover the $360^\circ$ around the robot and are used for PIL localization.
The robot mission is a 187 meters long autonomous patrolling in the laboratory to inspect the state (open/close) of two valves (V1-V2) (Figure 8-Bottom).

A PIL map have been made from the corridor environment with an tile size of 20cmx20cm and Ns=57 giving us a resolution of 0.11 rad.

We choose to observe PIL algorithm performances between the starting point DS and V1 (F area).

4.2 Simulation performance

In order to estimate the efficiency of the PIL approach in our environment we virtually put the robot simulation in a grid cell (pixel) of the grid (pseudo-image), with a random but known pose. A window of 4 m square is used to limit the research area. Then we look at the PIL algorithm results to see if it selects a good grid-cell or not. This test is repeated 63 times for each pseudo-image pixel generating a statistical analysis of the PIL algorithm efficiency. The Figure(8-Top-Right shows the results of this study. In red, we can find the areas where the PIL algorithm succeed with more than 80% to find the correct robot place. The blue color identifies the areas where the success is lower than 50%. It is clear that the results are better when the environment does not present a high level of symmetry which is a recurring known problem for indoor localization.

4.3 Experimental result

The robot velocity in the area F is set to $V = 0.2m/s$. If a position is computed by the PIL localization algorithm, the robot position is updated according to these results, each 2s. Figure 9-Top shows the real robot and localization positions. The localization data are associated with time. The experimentation video record and time localization are then synchronized to deduce the real robot positions. The real robot angle is deduced by analyzing the sensors signature according to the robot position and attitude.
5 Discussion and conclusion

The PIL algorithm proposed allows to solve the global localization and the robot kidnapping issues. It is based on an off-line massive accumulation of perceived distances all around a simulated planar mobile robot. These data are organized into level functions that are orientation-free. The smart filtering of these data during the on-line phase allows a fast detection of possible localization candi-
Fig. 9. Top: Real robot position versus the localization results. Bottom: Angular and position localization errors according to real robot pose.

dates. These candidates are then compared to the real measurement provided by the real robot. This correlation-based process eliminates all but one of these candidates and provides the robot orientation. The efficiency of this approach have been estimated in simulation and experimentaly using Pioneer 3DX robot in an indoor environemnt. At this point, we can emphasize important remarks and foresee the improvements that could be brought to this algorithm:

- The proposed approach is direct, e.g. it does not rely on the data fusion with proprioceptive information (odometers). This is an advantage (even if the robot is lost, due to unknown obstacles, it will be able to recover as soon as it will move in a known and learned area), and a disadvantage in the sense that a fusion process would certainly help the global filtering (better windowing for instance).
- The proposed approach is not, a priori, compatible with the SLAM (Simultaneous Localization an Mapping) approach. But it could become compatible if associated with a SLAM algorithm (FastSLAM, GraphSLAM,...) as a pre-caching technique [7] that would accelerate the localization process.
Only a few orientation-free level functions have been proposed here (Energy, Percentage) and some improvements should be brought by seeking the ‘good’ functions that would lead to a better filtering of the position candidates. These functions must own the fundamental property of orientation-freeness and should own other properties that have not been discussed here. For instance, robustness with respect to unknown obstacles.

The heart of the proposed method is the explicit analogy between maps and images. What has not been developed here, but that could bring a substantial improvement, is the study and the pre-processing of the transformed pseudo-images. For instance, histogram equalizations could be used to improve contrast, pseudo-color transformation to augment the level of seen details, derivation (image gradients) to localize pathways,...

References

8. https://www.youtube.com/watch?v=oAZPaPP606I.