MOBILE ROBOT LOCALIZATION BASED ON KALMAN FILTERING

HUGO L. GOSMANN[a]

Depto. de Eng. Elétrica - ENE
Universidade de Brasília - UnB
C.P. 04591, 70.910-900 Brasília
hugo.gosmann@ieee.org

SJUR J. VESTLI

Institute of Robotics - IfR
Swiss Federal Institute of Technology - ETHZ
ETH-Center, CLA, CH-8092 Zurich
s.vestli@ieee.org

Abstract— This paper describes a method for mobile robot localization in a known environment. The technique combines position estimation from odometry with observations of the environment from a CCD camera. Fixed lamps in the environment provide landmarks. The position of these landmarks is known *a priori* by the robot. At each localization cycle an image of the environment is captured and processed by the vision system. The information obtained is then used by a Kalman filter to correct the position and orientation of the robot. The system was implemented using the SmartROB, a mobile robot platform developed at the ETHZ. Results from experiments in a real environment are presented.

Key Words— Autonomous mobile robots; Position estimation; Kalman filters.

1 Introduction and Motivation

(Leonard and Durrant-Whyte, 1992) summarized the general problem of mobile robotics by the following three questions: "Where am I?", "Where am I going?" and "How should I get there?".

The first question corresponds to *localization*, one of the major tasks of autonomous robot navigation. How can I work out where I am in a given environment, based on what I can see and what I have previously been told?

In a typical indoor environment with a flat floor, the task of localization becomes a matter of determining the Cartesian coordinates (x, y) and the orientation θ of the robot on a two dimensional plan. For a wheeled robot, odometry is one of the most important means of achieving this task.

However, with time, odometric localization unboundedly accumulates errors due to problems like surface roughness and undulation, wheel slippage and variations in load.

Although good approaches have already been investigated, all these problems make it rather difficult to perform really accurate navigation using only odometry. Therefore, some other kind of position updating method must be used. To reach its destination with reasonable accuracy, the robot requires external sensors and sensor fusion algorithms to relate knowledge about its environment to the information obtained from its sensors.

For a vision sensor, fixed objects in the known environment provide *landmarks* which are listed in a database. In general, landmarks have a fixed and known position, relative to which a robot can localize itself. The main task is then to recognize the landmarks reliably and to calculate the robot's position (Murata and Hirose, 1993).

The basic tool to approach navigation and sensor fusion is the Kalman filter. It combines all measurement data to get an optimal estimate of the system state in a statistical sense. Inputs to a Kalman filter are the system measurements. The *a priori* information are the system dynamics and the noise properties of the system and sensors. Outputs of the Kalman filter are the estimated system state and the innovation, *i.e.*, the difference between the predicted and observed measurement (Abidi and Gonzalez, 1992; Grewal and Andrews, 1993).

1.1 Specification of our Problem

In the recent years the ETHZ (Swiss Federal Institute of Technology), through the IfR (Institute of Robotics), has developed a mobile robot platform called SmartROB, which is a versatile, easy to use, mobile robot kit, suitable for the realization of a wide variety of tasks.

This work has been motivated by the interest of embedding a vision system in the SmartROB using a frame grabber developed at the ETHZ.

Our goal is to implement a localization system for the robot using vision and odometry. The chosen environment was part of the Laboratory room at the ETHZ. In this area, three fluorescent lamps were vertically placed in predetermined positions to be used as landmarks. The SmartROB, equipped with a CCD camera and the frame grabber, should be able to localize itself and navigate using information from odometry (encoders) and the vision system (position of landmarks extracted from images).

2 Localization

Localization is the process of determining the position of the robot with respect to a global reference frame (*i.e.* a coordinate system). It is a cyclic process that should continuously keep the robot on track.

[a]Master Degree Student at UnB and Professor at IESB - Instituto de Educação Superior de Brasília.

In our case, a combination of information from odometry and from the vision system is used to update the robot's position. The tool used to fuse this information is the Kalman filter, which is described in the following sections.

2.1 Modeling Odometry

In the SmartROB, the odometry calculation is based on two optical encoders mounted on the front wheels. We assume that Δs_l is the distance traveled by the left wheel, and Δs_r is the distance traveled by the right wheel in the time cycle Δt.

Let the origin of the robot's coordinate system be the middle point of the front axle, as shown in Fig. 1. Then, the position and orientation changes of the robot are calculated using:

$$\Delta s = \frac{1}{2}(\Delta s_r + \Delta s_l), \quad \Delta\theta = \frac{1}{H}(\Delta s_r - \Delta s_l)$$

where H is the distance between the two front wheels. In other words, the robot's location changes by a translation forward through the distance Δs followed by a rotation counterclockwise through the angle $\Delta\theta$.

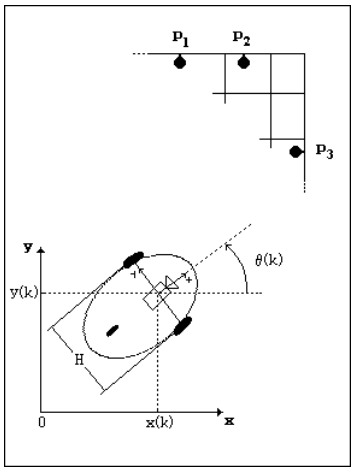

Figure 1. Global and Local Coordinate Systems

2.2 The Kalman Filter

We denote the position and orientation of the vehicle at time step k by the state vector $\mathbf{x}(k) = [x(k), y(k), \theta(k)]^T$ comprising a Cartesian location and an angle defined with respect to a global coordinate system, as shown in Fig. 1. At initialization, the robot starts at a known position (*e.g.* the origin), and has an *a priori* map of n_E landmarks, whose locations are specified by the set of known vectors $\{\mathbf{p}_i = (p_x, p_y) \mid 1 \leq i \leq n_E\}$.

At each time step, observations $\mathbf{z}_j(k+1)$ of these landmarks are taken. The Kalman filter is then used to associate measurements $\mathbf{z}_j(k+1)$ with the correct landmarks $\mathbf{p}_i$ to compute $\hat{\mathbf{x}}(k+1|k+1)$, the updated estimate of the vehicle's position. It consists of two models: a *plant* model and a *measurement* model.

The Plant Model. The plant model describes how the vehicle's position $\mathbf{x}(k)$ changes with time in response to a control input $\mathbf{u}(k)$ and a noise disturbance $\mathbf{v}(k)$, and has the form

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) + \mathbf{v}(k),$$

where $\mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$ is the non-linear state transition function and $\mathbf{v}(k)$ is a noise source assumed to be zero-mean Gaussian with covariance $\mathbf{Q}(k)$.

The control input $\mathbf{u}(k) = [\Delta s(k), \Delta\theta(k)]^T$ comes from the odometry model discussed in the previous section, and leads us to the following state transition function:

$$\mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) = \begin{bmatrix} x(k) + \Delta s(k)\cos\theta(k) \\ y(k) + \Delta s(k)\sin\theta(k) \\ \theta(k) + \Delta\theta(k) \end{bmatrix}$$

The Measurement Model. The robot is equipped with a CCD camera and a frame grabber. For our convenience the camera was installed in the origin of the robot's coordinate system. The information we have *a priori* is the position of our n_E landmarks with respect to the global coordinate system. Our vision system will provide us with angles corresponding to the extracted landmarks (*i.e.* observations). Then we can define the set of observations

$$Z(k) = \{\mathbf{z}_j(k) \mid 1 \leq j \leq n_O\}$$

where n_O is the number of *observed* landmarks. We have already seen that n_E is the total number of known landmarks, which will also be referred to as the number of *expected* landmarks.

The measurement model relates a sensor observation to the vehicle position and has the form:

$$\mathbf{z}_j(k) = \mathbf{h}(\mathbf{x}(k), \mathbf{p}) + \mathbf{w}_j(k),$$

The measurement function $\mathbf{h}(\mathbf{x}(k), \mathbf{p})$ expresses an observation $\mathbf{z}(k)$ from the sensor as a function of the vehicle position $\mathbf{x}(k)$. It has the form:

$$\mathbf{h}(\mathbf{x}(k), \mathbf{p}) = \arctan\left(\frac{\mathbf{p}_y - y(k)}{\mathbf{p}_x - x(k)}\right) - \theta(k).$$

Each observation is assumed corrupted by a zero-mean, Gaussian disturbance $\mathbf{w}_j(k)$ with covariance $\mathbf{R}_j(k)$.

2.3 The Localization Cycle

Given the *a posteriori* vehicle position estimate $\hat{\mathbf{x}}(k|k)$ and its covariance $\mathbf{P}(k|k)$ for time k, the current control input $\mathbf{u}(k)$, the current set of observations $Z(k+1)$ and the *a priori* map, compute the new *a posteriori* estimate $\hat{\mathbf{x}}(k+1|k+1)$ and its

covariance $\mathbf{P}(k+1|k+1)$. The algorithm consists of the following steps:

Vehicle Position Prediction. First, using the plant model and the knowledge of the control input $\mathbf{u}(k)$, we predict the robot's new location at time step $k+1$:

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{f}(\hat{\mathbf{x}}(k|k), \mathbf{u}(k)).$$

Next we compute $\mathbf{P}(k+1|k)$, the variance associated with this prediction:

$$\mathbf{P}(k+1|k) = \nabla\mathbf{f}\ \mathbf{P}(k|k)\ \nabla\mathbf{f}^T + \mathbf{Q}(k) \quad (1)$$

where $\nabla\mathbf{f}$ is the Jacobian of the state transition function $\mathbf{f}(\hat{\mathbf{x}}(k|k), \mathbf{u}(k))$ obtained by linearizing about the updated state estimate $\hat{\mathbf{x}}(k+1|k)$:

$$\nabla\mathbf{f} = \begin{bmatrix} 1 & 0 & -\Delta s(k)\sin(\theta(k)) \\ 0 & 1 & \Delta s(k)\cos(\theta(k)) \\ 0 & 0 & 1 \end{bmatrix}$$

Observation. The next step is to obtain the observation set $Z(k+1)$ from the vehicle's sensor system on the new vehicle location, that is, capture an image and apply an algorithm to identify the landmarks, converting them into angles.

Measurement Prediction. Now we use the predicted robot location $\hat{\mathbf{x}}(k+1|k)$ and the *a priori* map to generate predicted observations for each landmark $\mathbf{p}_i$:

$$\begin{aligned} \hat{\mathbf{z}}_i(k+1) &= \mathbf{h}_i(\hat{\mathbf{x}}(k+1|k), \mathbf{p}_i) \\ &= \arctan\left(\frac{\mathbf{p}_{iy} - y(k)}{\mathbf{p}_{ix} - x(k)}\right) - \theta(k), \quad (2) \end{aligned}$$

for $i = 1, \ldots, n_E$ to yield the set of predictions:

$$\hat{Z}(k+1) = \{\hat{\mathbf{z}}_i(k+1) \mid 1 \leq i \leq n_E\}$$

which contains n_E predicted landmarks.

The predicted state estimate $\hat{\mathbf{x}}(k+1|k)$ is used to compute the measurement Jacobian

$$\nabla\mathbf{h}_i = \begin{bmatrix} \frac{\mathbf{p}_{iy} - y(k)}{(\mathbf{p}_{ix} - x(k))^2 + (\mathbf{p}_{iy} - y(k))^2} \\ \frac{-\mathbf{p}_{ix} - x(k)}{(\mathbf{p}_{ix} - x(k))^2 + (\mathbf{p}_{iy} - y(k))^2} \\ -1 \end{bmatrix}^T \quad (3)$$

for each prediction.

Matching. The goal of the matching procedure is to produce an assigment from measurements $\mathbf{z}_j(k)$ to landmarks $\mathbf{p}_i$. For each prediction and observation we compute the innovation ν_{ij}.

$$\begin{aligned} \nu_{ij}(k+1) &= [\mathbf{z}_j(k+1) - \hat{\mathbf{z}}_i(k+1)] \quad (4) \\ &= [\mathbf{z}_j(k+1) - \mathbf{h}_i(\hat{\mathbf{x}}(k+1|k), \mathbf{p}_i)]. \end{aligned}$$

The innovation covariance is then calcultated by

$$\mathbf{S}_{ij}(k+1) = \nabla\mathbf{h}_i\mathbf{P}(k+1|k)\nabla\mathbf{h}_i^T + \mathbf{R}_i(k+1). \quad (5)$$

A *validation gate* is used to determine the correspondece between predictions and observations:

$$\nu_{ij}(k+1) \leq G. \quad (6)$$

This equation is used to test each sensor observation $\mathbf{z}_j(k+1)$ with each predicted measurement $\hat{\mathbf{z}}_i(k+1)$. When a single observation falls in the validation gate, we get a successfull match. Measurements which do not fall in this gate are ignored for localization. The same occurs if a measurement falls in the gate for more then one prediction, or vice-versa.

Estimation. The final step is to use successfully matched predictions and observations to compute $\hat{\mathbf{x}}(k+1|k+1)$, the updated vehicle position estimate. First we build a new vector $\mathbf{z}(k+1)$ containing all the matched observations for time $k+1$ and calculate the composite innovation $\nu(k+1)$. Then we build another vector $\nabla\mathbf{h}$ with all the validated predictions. Using the composite noise vector $\mathbf{R}(k+1)$ we compute the composite innovation covariance $\mathbf{S}(k+1)$ as in Eq. (5). We then calculate the Kalman filter gain

$$\mathbf{W}(k+1) = \mathbf{P}(k+1|k)\Delta\mathbf{h}^T\mathbf{S}^{-1}(k+1)$$

to compute the updated vehicle position estimate

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{W}(k+1)\nu(k+1)$$

with associated variance

$$\mathbf{P}(k+1|k+1) = \mathbf{P}(k+1|k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}^T(k+1).$$

2.4 System Implementation

We implemented our Kalman filter algorithm for localization using XOberon (a real-time operating system that runs in the SmartROB). Our system was consisted of two modules. The first one, called SRLocalization.Mod implements the Kalman filter algorithm. The second module, called ExtractLandmarks.Mod implements the vision procedures to capture images and identify the angles corresponding to the extracted landmarks.

The first *a priori* information defined in SRLocalization is the position of our landmarks with respect to our global coordinate system:

$$Map = \begin{cases} p_1 = (1.203m, 3.880m); \\ p_2 = (2.100m, 3.880m); \\ p_3 = (2.733m, 2.870m); \end{cases}$$

With respect to the Vehicle Position Prediction, $\hat{\mathbf{x}}(k+1|k)$ is obtained directly from the odometry calculation. This value is accessible in the SmartROB by means of software.

With $\hat{\mathbf{x}}(k+1|k)$ we can obtain $dx = \Delta s\cos\theta(k)$, $dy = \Delta s\sin\theta(k)$ and $\Delta\theta$ by just subtracting from $\hat{\mathbf{x}}(k|k)$. We can then update $\nabla\mathbf{f}$.

$\mathbf{Q}(k)$ for each time step k is defined by:

$$\mathbf{Q}(k) = \begin{bmatrix} K_{ss}dx & 0 & 0 \\ 0 & K_{ss}dy & 0 \\ 0 & 0 & K_{s\theta}\Delta s + K_{\theta\theta}\Delta\theta \end{bmatrix}$$

where K_{ss}, $K_{s\theta}$ and $K_{\theta\theta}$ are drifting coefficients presented in (Chenavier and Crowley, 1992). These coefficients were empirically set to be:

$$K_{ss} = 0.01; \quad K_{s\theta} = 0.005; \quad K_{\theta\theta} = 0.01. \quad (7)$$

We then calculate $\mathbf{P}(k+1|k)$ using Eq. (1).

The next step is the Observation. Now we call the module ExtractLandmarks which performs all the image manipulation. For the time being, what we need to know is that it will return a set $Z(k+1)$ of n_O observed angles $\mathbf{z}_j(k+1)$ corresponding to the extracted landmarks.

Now comes the Measurement Prediction step. First, with $\hat{\mathbf{x}}(k+1|k)$ and the map, we calculate the n_E expected angles using Eq. (2) and also $\nabla\mathbf{h}_i$ for $1 \leq i \leq n_E$ from Eq. (3).

In the Matching step we calculate all the innovations ν_{ij} using Eq. (4) and apply Eq. (6) to match the right angles. The validation gate G was empirically set to $G = 0.02rad$.

In the Estimation step, we update the position. If no angles are matched in the previous step then the updated vehicle position $\hat{\mathbf{x}}(k+1|k+1)$ is set to be $\hat{\mathbf{x}}(k+1|k)$ and $\mathbf{P}(k+1|k+1) = \mathbf{P}(k+1|k)$. This means that the position was updated only using information from odometry.

To calculate $\mathbf{S}_i(k+1)$ we use Eq. (5) with

$$\mathbf{R}_i(k+1) = [r_{ii}]$$

where r_{ii} is the covariance associated with the landmark i.

Then for a matched angle we can calculate the Kalman filter by

$$\mathbf{W}_i(k+1) = \mathbf{P}(k+1|k)\Delta\mathbf{h}_i^T \frac{1}{\mathbf{S}_i(k+1)}$$

and also

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{W}_i(k+1)\nu_i(k+1) \quad (8)$$

In case of more than one matched angle this procedure is repeated for each one of them separately.

Now we just have to update the robot's position. To do that we just change the attributes of the object odometry for new values.

We start the algorithm with a $\mathbf{P}(0|0)$, whose elements correspond to the initial uncertainty in the robot's position.

3 Vision System

3.1 Identifying Landmarks

The first task of our system is to capture an image from the environment. Fig. 2 shows an example of a raw image captured by the camera.

Now that we have the image, we identify in pixel level where the lamps are located. Lamps are sources of light and provide regions of high brightness that characterize them. All we have to do is detect these regions and find the peaks of brightness, using a predefined *pixel threshold*.

Figure 2. Raw Image from Camera

The algorithm we developed condenses the image information in a single vector with dimension equal to the number of columns of the image. We initialize this vector with zeros. Then for each column of our image, we test the level of brightness of each element. If it is greater than the threshold we add 1 to the corresponding element in our vector. In the end of this process our vector contains all the information we need to detect the peaks of brightness.

Now we "walk" through this vector and analyze each set of 5 subsequent elements, summing their values. If the result is greater than a *sum threshold* then we define the middle element to be an extracted landmark as in Fig. 3.

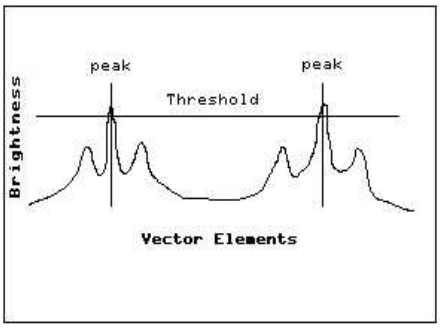

Figure 3. Extracted Peaks in Vectorized Image

The process of tuning the threshold values depends on many factors such as the reflection of light on objects, the amount of artificial or natural light in the environment and image distortion.

Ideally, we would want to have some dynamic procedure that could perform this task in real-time taking into account the instantaneous conditions of the environment. This is not a trivial task, so we have decided to define these values once in the beginning of the process. To be able to do that we assumed that the environmental conditions wouldn't change too much.

This fact brings some drawbacks. In Fig. 3,

for example, an inadequate choice of threshold could lead us to false observations, caused by the reflection of light on the wall that generates two regions of high brightness around the lamp.

Because of these problems we need a good Matching procedure to be able to distinguish correct from false landmarks.

Finished the image processing procedures, we still need to convert our eventual extracted landmarks from positions in a vector into angles with respect to the robot's coordinate system. This process is done after a calibration of the camera, where we build a conversion table that associates an angle to each position in our vector.

3.2 The Matching Procedure

In our algorithm, after the Measurement Prediction and Observation steps, we have a set $\hat{Z}(k+1)$ of n_E predicted angles and also a set $Z(k+1)$ of n_O observed landmarks.

The number of elements in these two sets (n_E and n_O) is not necessarily the same, *e.g.* the case we described before where reflection areas on the wall could be identified as landmarks. The following question arises: How do we distinguish correct observations from false ones?

Many authors have already discussed this problem and solutions point to the use of filters that separate false observations from correct ones (Abidi and Gonzalez, 1992; Leonard and Durrant-Whyte, 1992; Grewal and Andrews, 1993). We chose a simple approach based on just applying Eq. (6).

The choice of G depends on the accuracy of our odometry and also our vision system, and tells us how far, at most, an extracted landmark can be from a predicted one. If the odometry is precise enough we can make G small, otherwise we have to relax it. This assumption is quite reasonable because if we set G to a small value we are assuming that an observation has to be close enough to a prediction to be matched. If our odometry is not so precise, we can loose important information just because of this strict choice of G. On the other hand, if we relax G we allow the observations to be in a wider range around our prediction, consequently we have a greater chance of detecting a false observation. Clearly there's always a trade-off between these two things.

4 Experiment Results

4.1 Evaluating our Odometry

The first step toward testing our system is to evaluate the accuracy in our odometry. We used the UMBmark (University of Michigan Benchmark) test proposed in (Borenstein et al., 1996) to illustrate qualitatively the characteristics of the odometry in the SmartROB.

Our measurements showed that for a 10-meter run the error in the position was approximately 10 centimeters (1% error).

Our intention in running this test was just to have some feeling about our odometry, not to improve its characteristics. The reason for that is the fact that our ultimate goal is to make a comparison between the use of odometry alone and its combination with vision. If our odometry were too precise, it would be more difficult to illustrate the improvements added by a vision system or any other sensor.

4.2 Localization using only Odometry

When we perform localization using only odometry, at each time step we estimate the robot's position based on information from the encoders and than associate a correspondent uncertainty to it. With this approach each estimated position is surrounded by a characteristic "error ellipse" which indicates a region of uncertainty for the robot's actual position. Typically, these ellipses grow with travel distance, due to odometry errors. To construct them we used $\mathbf{P}(k|k)$.

Fig. 4 shows the result obtained with the SmartROB when only odometry is used for localization. The crosses represent the position given by the SmartROB's odometry. The balls indicate the real position of the SmartROB. The theory is confirmed and the robot looses precision with travel distance. In the long-term it will surely get lost. Another sensor is then required to overcome this problem.

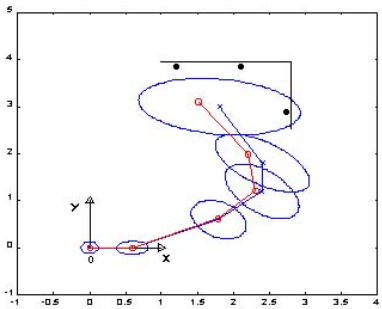

Figure 4. Localization using only Odometry

4.3 Localization using Odometry and Vision

In the second run of experiments our vision system was used. To better illustrate the improvements achieved in localization we defined a sequence of steps for each run. The robot starts at the origin with an initial uncertainty defined by $\mathbf{P}(0|0)$. In each subsequent step we move the robot to a new position and than run the localization cycle to update the position. Fig. 5 shows the result.

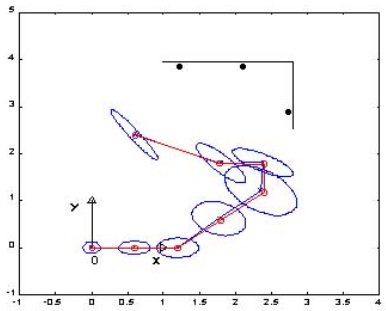

Figure 5. Localization using Odometry and Vision

In the first five steps of this path the SmartROB was intentionally placed with such orientation that no landmarks were seen. The result is that only the odometry is used to update the robot's position, therefore the associated uncertainty starts growing.

When the robot first sees a landmark in the sixth step we notice that the position is corrected and the uncertainty is reduced. The same occurs in the next two steps. It means that every time the robot sees a landmark we have more confidence in our position. This confidence is determined by the state covariance matrix $\mathbf{P}(k|k)$ that is used to construct the error ellipses. Our task is to keep $\mathbf{P}(k|k)$ as small as possible. In a real situation the localization algorithm has to be executed continuously and the idea is that the robot be able to see at least one landmark at a time to keep his position accurate.

In a system where we have a precise odometry and also a precise vision system, the uncertainty associated to the actual position should be reduced to very small values. This allows very precise navigation. In our case we can notice that when the robot sees a landmark the uncertainty does not grow and is even a little bit reduced, but not completely.

Many reasons led us to these results. The first one was related to the backslash in the fixation of our camera, that introduces inaccuracies in the measurements. It means that we have to reduce our confidence with respect to our vision system. Besides that, our Kalman filter used a simple Matching procedure. As we discussed previously, when we relax the value of G we have a greater chance of detecting false landmarks. If we just use Eq. (6) without any other criterion of selection, when more then one observation fall in the validation gate we loose precious information.

Besides all these problems in our implementation we were able to navigate the SmartROB quite precisely keeping bounded the uncertainty around the position. The results were satisfactory if we take into account the simplifications we made. It follows that it's possible to perform accurate navigation of mobile robots with a simple approach.

5 Conclusion and Future Work

In this work we have developed a localization algorithm for mobile robot navigation. This algorithm was based on odometry and vision. Fluorescent lamps were used as landmarks. A Kalman filter was used to update the robot's position.

The results showed that when we use only odometry for localization, the uncertainty correspondent to the robot's position grows with distance travel. On the other hand when we combine odometry with vision we can keep the uncertainty bounded, allowing precise navigation. Due to some problems already described our results were not optimal, however we were able to achieve our ultimate goals.

Improvements in our system could be addressed in the following directions: 1) Use a more stable fixation for the camera to increase precision; 2) Develop a more sophisticated matching procedure in the localization cycle; 3) Develop algorithms to dynamically calculate the threshold values; and 4) Optimize the code.

Suggestions of new approaches include the use of landmarks that naturally occur in the environment, such as edges of a table, corners of the room, doors, etc. The main idea is that we don't need to change the environment just to be able to have precise localization.

References

Abidi, M. A. and Gonzalez, R. C. (1992). *Data Fusion in Robotics and Machine Intelligence*, Academic Press.

Borenstein, J., Everett, H. R. and Feng, L. (1996). *Navigating Mobile Robots*, Wellesley, Massachusetts.

Chenavier, F. and Crowley, J. L. (1992). Position estimation for a mobile robot using vision and odometry, *IEEE International Conference on Robotics and Automation*, Nice, France, pp. 2588–2593.

Grewal, M. S. and Andrews, A. P. (1993). *Kalman Filtering: Theory and Practice*, Prentice-Hall.

Leonard, J. J. and Durrant-Whyte, H. F. (1992). *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers, London.

Murata, S. and Hirose, T. (1993). Onboard locating system using real-time image processing for a self-navigating vehicle, *IEEE Transactions on Industrial Electronics* **40**(1): 145–154.