Robotics and Autonomous Systems 58 (2010) 295–305

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Omnivision-based KLD-Monte Carlo Localization☆

C. Gamallo [a], C.V. Regueiro [b], P. Quintía [b], M. Mucientes [a,*]

[a] Department of Electronics and Computer Science, University of Santiago de Compostela, E-15782, Spain
[b] Department of Electronic and Systems, University of A Coruña, E-15071, Spain

ARTICLE INFO

Article history:
Available online 20 November 2009

Keywords:
Monte Carlo Localization (MCL)
Omnidirectional camera
Feature-based map

ABSTRACT

Mobile robots operating in real and populated environments usually execute tasks that require accurate knowledge on their position. Monte Carlo Localization (MCL) algorithms have been successfully applied for laser range finders. However, vision-based approaches present several problems with occlusions, real-time operation, and environment modifications. In this article, an omnivision-based MCL algorithm that solves these drawbacks is presented. The algorithm works with a variable number of particles through the use of the Kullback–Leibler divergence (KLD). The measurement model is based on an omnidirectional camera with a fish-eye lens. This model uses a feature-based map of the environment and the feature extraction process makes it robust to occlusions and changes in the environment. Moreover, the algorithm is scalable and works in real-time. Results on tracking, global localization and kidnapped robot problem show the excellent performance of the localization system in a real environment. In addition, experiments under severe and continuous occlusions reflect the ability of the algorithm to localize the robot in crowded environments.

1. Introduction

Nearly all the tasks that an autonomous mobile robot has to carry out require knowledge on the position of the robot. Given a map of the environment, a localization system estimates the pose (position and angle) of the robot in the map based on a motion model of the robot and/or a measurement model for the sensors. A localization algorithm must be reliable, robust and executable in real-time.

Very different types of sensors have been used for localization. In particular, the three most widely used are laser [1,2], sonar [3,4] and cameras [5,6]. In recent years, different successful localization algorithms based on laser range finders have been proposed [7–9].

Moreover, the use of cameras for localization tasks has received increasing attention. The main advantages of these sensors are the quantity and quality of information that can be extracted from one acquisition. This is particularly interesting for localization, as different types of landmarks can be detected using information on shape, color, etc. Popular vision approaches are feature-based and pixel-based. Feature-based techniques [10,11] exploit typical properties of the environment or any distinctive and recognizable objects (landmarks). Pixel-based approaches [5,12] compute the correlation between images, in order to estimate the mobile robot pose.

In real environments, the presence of people in the surrounding of the robot is usual. Both static and moving people are detected by the robot sensors, generating measurements that have to be discarded. These measurements are useless for localization but, also, introduce noise in the localization algorithms, as sometimes measurements coming from people are confused with those from landmarks or other objects used for localization. This problem can be addressed by attempting to classify the measurements coming from people, and those coming from objects that are relevant to the localization task. For example, for laser range finders, several authors have classified measurements in order to distinguish between people and objects [13,14], while others have built maps in the presence of several moving people [15].

Nevertheless, when the environment is highly populated, the problem is not simply that of filtering measurements coming from people, but the inability to sense any landmark over long periods of time, thus rendering localization truly difficult, while the knowledge about the pose has a huge uncertainty. These situations could be typical, for example, for a tour-guide robot operating in a museum, and surrounded by a group of people who are interacting with the robot. The only area of the environment that can be seen regularly in these conditions is the ceiling.

In this article, a localization algorithm for a tour-guide robot operating in a very crowded environment is presented. Due to

☆ This work was supported in part by the Spanish Ministry of Science and Innovation under grant TIN2008-00040. Manuel Mucientes is supported by the *Ramón y Cajal* program of the Spanish Ministry of Science and Innovation.

* Corresponding author.
E-mail addresses: cristina.gamallo@usc.es (C. Gamallo), cvazquez@udc.es (C.V. Regueiro), pquintia@udc.es (P. Quintía), manuel.mucientes@usc.es (M. Mucientes).

0921-8890/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2009.08.007

the presence of groups of people, typical localization solutions based on range sensors, such as laser or sonar, or those based on the detection of conventional landmarks with cameras do not work properly. Moreover, the environment cannot be modified by introducing artificial landmarks to facilitate localization.

Our approach is based on a map of the lights (landmarks) placed on the ceiling of the environment. These landmarks are easy to detect, repetitive and usually visible for long trajectories. On the other hand, most buildings have these kinds of landmarks, so there is no need for prior adaptation of the environment in order to use the proposed localization method. The main problem is their individual identification, as they are usually identical, making the data association more difficult. Measurements from the environment are acquired with an omnidirectional camera fitted with a fish-eye lens. This sensor provides an extremely wide field of vision, covering half the space of the environment and, therefore, it can get a high amount of information in one acquisition.

In recent years, probabilistic techniques have been successfully applied in different fields of robotics, such as mapping, localization, tracking, or planning [16]. The application of particle filters for localization, named Monte Carlo Localization (MCL), has proved to be a very popular approach for solving both position tracking and global localization problems. The proposed localization algorithm is based on MCL, combined with the injection of random particles to recover from failures (kidnapped robot problem), and with an efficient implementation due to the adaptive sample set size through KLD-sampling.

In summary, the main points of our proposal are: first, the localization algorithm, KLD-Augmented-MCL, can recover from failures, works in real-time and is robust to severe and continuous occlusions. In second place, data of the landmarks in the environment are obtained with an omnidirectional camera with a fish-eye lens and an algorithm for the detection of the lights on the ceiling. Finally, the system has been tested in real and complex conditions, showing a good performance.

The article is organized as follows. Section 2 analyzes other contributions for localization, in particular for vision-based approaches. Section 3 describes the localization algorithm (KLD-Augmented-MCL), while Section 4 presents the measurement model. Finally, Section 5 shows the results for different experiments in real conditions, and Section 6 points out the conclusions.

2. Related work

There has been extensive research in the literature to solve the localization problem using vision. Most of the algorithms are based on probabilistic approaches. For example, in [17], a localization system based on the MCL algorithm is presented. This algorithm is a Bayesian filtering method that uses a sampling-based density representation. The robot was equipped with a monocular camera pointing to the ceiling, and the map was built as a mosaic of 250 images captured and globally aligned. The pose of the robot was estimated using information of the brightness of the images, and tests of tracking and global localization were performed. A similar approach has also been followed in [1].

In [18], a localization system using a particle filter and a monocular camera was presented. The Scale-Invariant Feature Transform (SIFT) signature of the images in a database was used for comparison with the present image. The majority of vision-based approaches rely on the existence of a database of images that are compared with the present image. This has several disadvantages. In first place, the images are recorded for a route, and if the route is changed, or the robot has a very different orientation in a position, the error in the localization increases. In second place, the localization error increases greatly owing to partial occlusions, for example in the presence of people. Finally, this kind of algorithm is highly sensitive to modifications in the environment (e.g. a door was closed and now is open, a new object is placed or removed from the environment, etc.).

Stereo camera systems have also been used for localization, as in [19] with an MCL algorithm. Images were recorded on a database and compared with the present one using a histogram of local features. The system was tested on tracking, global localization and kidnapping problems.

Omnidirectional cameras are particularly interesting for localization algorithms due to their high field of view (FOV). The first work combining localization and omnivision was published in [20]. In [21], authors proposed a hierarchical (three-step) algorithm that used line descriptors, a global filter (based on color descriptors) and a pyramid matching kernel. In [22], the similarities between images were measured using the average color value of each sector of the image, while in [23] the measurement model used the gray level curve sectors, and a sector matching algorithm based on the Dempster–Shafer fusion of different criteria.

Many omnivision approaches are based on the MCL algorithm using different techniques for the estimation of the weights of the particles. For example, in [12], the measurement model used the correlation of the present image with a set of images that had been recorded for a specific route. That comparison was carried out with a modified version of the SIFT algorithm. The system can cope with partial occlusions, but increasing the localization error (4 m for a 50% occlusion). Another approach was presented in [24], where a Fourier transform of omnidirectional images was used to weight the samples. In [6], the chromatic transitions for scan matching were employed to estimate the conditional probability of a measurement. Finally, in [25], the system used a graph-based representation of the operation area. The nodes of the graph were labeled with both visual feature vectors (average color value) extracted from omnidirectional images, and odometric data of the robot at the moment of node insertion.

The main differences of our approach with these proposals are:

- We use a feature-based map for localization, as the proposed feature extraction process is able to obtain the landmarks (lights) of the environment. The main advantage of this approach is that the system does not need a database of images and, therefore, the localization algorithm is not limited to certain routes, it is robust to changes in the environment, and also to occlusions, as only a few landmarks of the map are needed for localization at each time instant.
- The proposed algorithm is scalable with the size of the environment, in contrast with those approaches relying on a database of images. Moreover, only a few approaches that use image databases can work in real-time (limiting the size of the database), while our proposal works in real-time with no type of constraint. In addition, the adaptive number of particles contributes to real-time operation.
- Our approach is robust under severe and continuous occlusions, which makes it especially suited to crowded environments.

3. KLD-Augmented-Monte Carlo Localization algorithm

The localization algorithm is based on MCL, i.e. the Probability Density Function (PDF) of the pose of the robot is represented by a set of particles. Each particle is a sample of the PDF, and codifies a possible pose of the robot. Particles are distributed according to the PDF, i.e. the regions of the PDF with a higher probability will have a higher concentration of particles. The basic MCL algorithm (also known as particle filter localization) was proposed in [26,27], and solves both the local and global localization problems. However, the most complex localization problem, the kidnapped robot problem, cannot be solved with the basic MCL.

1: static w_{slow}, w_{fast}, N_{kld}
2: $\chi_t = \emptyset$, $\overline{\chi}_{t-1} = \emptyset$, $N_{t-1} = |\chi_{t-1}|$, $N_r = 0$, $N_b = 0$, $w_{avg} = 0$
3: **for** $i = 1$ to N_{t-1} **do**
4: **if** $rand() < \max\left\{0,\ 1 - w_{fast}/w_{slow}\right\}$ **then**
5: add random pose to $\overline{\chi}_{t-1}$
6: $N_r = N_r + 1$
7: **end if**
8: **end for**
9: **for all** $b \in G$ **do**
10: $b = 0$
11: **end for**
12: $\overline{\chi}_{t-1} = sampler(\chi_{t-1},\ \max\{0,\ N_{kld} - N_r\}) \cup \overline{\chi}_{t-1}$
13: $N_t = |\overline{\chi}_{t-1}|$
14: **for** $i = 1$ to N_t **do**
15: $x_t^i = sampleMotionModel\left(u_t,\ \overline{x}_{t-1}^i\right)$
16: $w_t^i = measurementModel\left(x_t^i,\ F,\ M\right)$
17: $\chi_t = \chi_t \cup \left\langle x_t^i,\ w_t^i\right\rangle$
18: $w_{avg} = w_{avg} + \frac{w_t^i}{N_t}$
19: **if** $bin\left(x_t^i\right) = 0$ **then**
20: $bin\left(x_t^i\right) = 1$
21: $N_b = N_b + 1$
22: **end if**
23: **end for**
24: $w_{slow} = w_{slow} + \alpha_{slow}\left(w_{avg} - w_{slow}\right)$
25: $w_{fast} = w_{fast} + \alpha_{fast}\left(w_{avg} - w_{fast}\right)$
26: **if** $N_b > 1$ **then**
27: $N_{kld} = \frac{N_b - 1}{2\epsilon}\left\{1 - \frac{2}{9(N_b-1)} + \sqrt{\frac{2}{9(N_b-1)}}z_{1-\delta}\right\}^3$
28: **else**
29: $N_{kld} = 1$
30: **end if**
31: return χ_t

Fig. 1. KLD-Augmented-MCL (χ_{t-1}, u_t, F, M).

In a kidnapping situation, the robot is placed in another location while the localization system still believes that the position of the robot is known. Kidnapped robot experiments reflect the ability of the localization algorithm to recover from failures. One approach for solving this problem with the MCL algorithm is the addition of random particles to the particles set. The Augmented-MCL algorithm [8] adds random particles to the set based on the probabilities of sensor measurements. Thus, the lower the average sensor measurements probabilities, the higher the number of random particles added to the set.

In all these MCL algorithms, the number of particles is fixed and must usually be large, in order to represent the PDF in the initial stages of a global localization. However, when the algorithm is simply tracking the position of the robot, the number of particles could be much lower. Therefore, adapting the particles set size improves the efficiency of MCL algorithms. One approach to adapt the number of particles is Kullback–Leibler divergence (KLD) sampling [9]. This is based on the KLD, which measures the difference between two PDFs. KLD-sampling determines the number of samples such that, with probability $1 - \delta$, the error between the true posterior and the sample-based approximation is less than ϵ.

The localization algorithm presented in this article (Fig. 1) is based on the Augmented-MCL algorithm, combined with KLD-sampling. The input parameters of the algorithm are the previous PDF, represented by the particles set (χ_{t-1}), the motion command (u_t), the set of detected features in the present time instant (F), and the map (M). The steps of the algorithm can be classified into three groups:

- *MCL part*: is the core of the algorithm (lines 12–17). It samples the previous particle set, updates the particles using the motion model and, finally, estimates the weight of each particle through the measurement model.
- *Augmented part*: corresponds to the insertion of random particles when the present measurements do not match with the expected ones (lines 3–8, 18, 24 and 25). This allows the algorithm to recover from localization failures, for example in the kidnapping problem.
- *KLD part*: calculates the number of particles that are necessary to appropriately represent the PDF of the pose of the robot (lines 9–11, 19–22, 26–29).

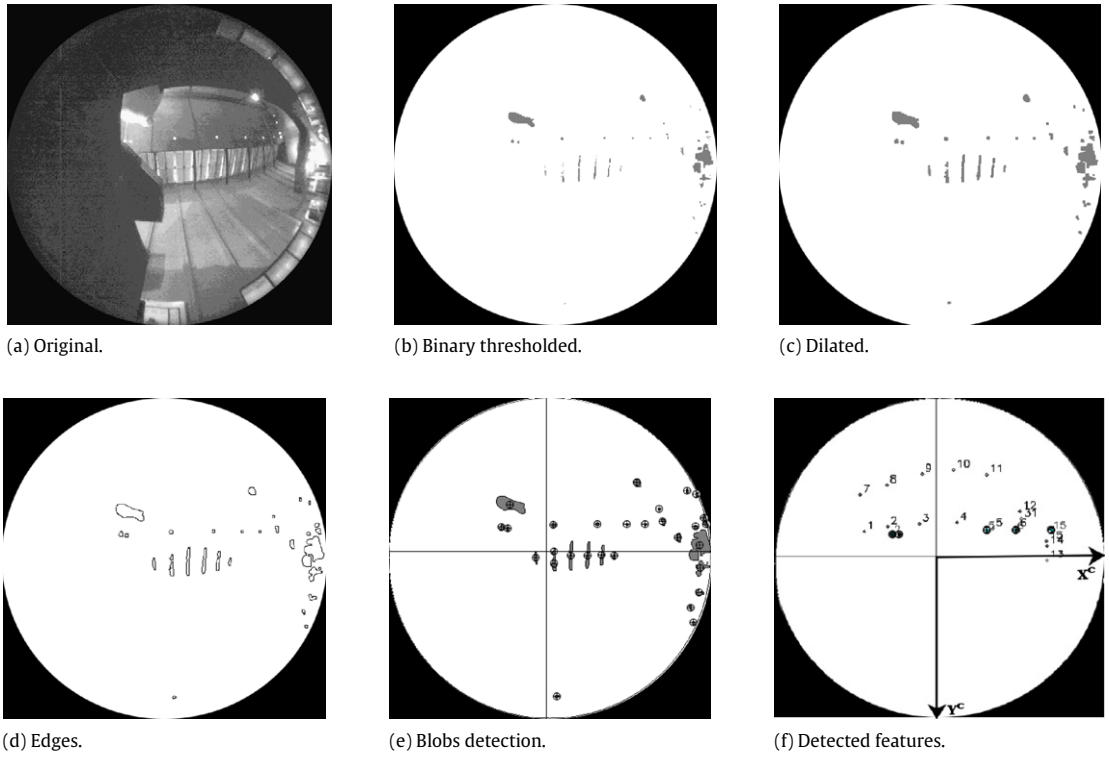
Going into the details, at the beginning of the algorithm (lines 3–8), random particles are added to the particle set: a random number in [0, 1] (*rand* ()) is generated N_{t-1} times (size of χ_{t-1}). Therefore, a random particle is added with probability $\max\{0, 1 - w_{fast}/w_{slow}\}$ to $\overline{\chi}_{t-1}$, which represents the particle set χ_{t-1} after sampling. w_{fast} and w_{slow} are the short- and long-term averages of the measurement likelihoods (weights of the particles). If w_{fast} and w_{slow} are very similar, or $w_{fast} > w_{slow}$, then no random particles will be added. This means that the measurements are the expected ones for the poses represented by the particles. On the other hand, if $w_{fast} < w_{slow}$ the PDF does not correspond with the measurements, and random particles must be added. The lower the quotient between the short and long-term, the higher the number of random particles. For instance, in the kidnapped robot problem w_{fast} decreases drastically, while w_{slow} decreases smoothly and, therefore, the number of randomly generated particles is high in the first iterations after the kidnapping.

In order to derive a statistical bound for the number of particles from the KLD, the state space has to be divided in bins. Lines 9–11 reset the information contained in each bin b, storing a value of 0 which indicates that no particles belong to that bin. Next (line 12), the algorithm resamples the initial particle distribution (χ_{t-1}), obtaining $N_{kld} - N_r$ new particles, where N_r is the number of particles randomly generated in the present iteration, and N_{kld} is the statistical bound estimated from the KLD for the particles in the previous iteration. *sampler* () function can be implemented with any sampling algorithm, for example the low variance sampler [16].

The core of the algorithm (lines 14–23) repeats, for each particle in $\overline{\chi}_{t-1}$, the following steps: first, taking into account the previous pose ($\overline{x}_{t-1}^i$) and the motion command (u_t), a random new pose (x_t^i) is drawn according to the motion model. Then, given the new pose, the detected features (F) and the map (M), the measurement model gives the likelihood of the detected features for that pose and map. This likelihood is the weight of the particle (w_t^i). The particle is added to the new set (χ_t), and the average weight is updated (w_{avg}). Finally (lines 19–22), if the bin of the particle is empty, it is set to non-empty and the number of non-empty bins is increased.

Once all the particles have been added to the set, the short- and long-term average weights are updated, using their difference with the present average weight and a parameter (α_{fast}, α_{slow}). These parameters can be considered as decay rates, and they must fulfill that $0 \leq \alpha_{slow} \ll \alpha_{fast}$.

Finally, the statistical bound for the number of particles is calculated (lines 26–29) using the number of non-empty bins (N_b) and the statistical error bounds ϵ and δ. $z_{1-\delta}$ represent the upper $1 - \delta$ quantile of the standard normal distribution. The estimation of the number of particles that are necessary to represent the PDF of the pose of the robot is based on KLD (N_{kld}), and is proportional to the number of non-empty bins (N_b). A high value for N_b means that the particles are distributed over the state space, i.e., there is a high uncertainty in the pose of the robot. Thus, in order to represent that situation and to keep track of all the plausible poses, a higher number of particles is needed. On the other hand, when N_b is low, the particles are concentrated around a few regions of the state space and, therefore, the PDF can also be represented with a lower number of particles.



(a) Original. (b) Binary thresholded. (c) Dilated.

(d) Edges. (e) Blobs detection. (f) Detected features.

Fig. 2. Feature extraction from an omnidirectional image.

One of the factors that can increase uncertainty during localization is the existence of symmetries in the environment. With a fixed number of particles, the set is divided among the different localization hypothesis. The concentration of particles in each cloud could be low and, therefore, the localization can fail due to the elimination of those particles that represent the right pose. On the contrary, through the use of KLD-sampling, the number of particles dynamically increases as the uncertainty in the pose of the robot is high, i.e., there are several good candidate poses due to the symmetries in the environment. Once the robot moves and the ambiguities are solved, uncertainty is reduced and, consequently, the number of particles that are necessary is also lower.

4. Measurement model

The sensor model that has been used is based on feature extraction from the images obtained by an omnidirectional camera with a fish-eye lens. Thus, the localization algorithm relies on a map (M) composed of a set of landmarks. These landmarks are the lights placed on the ceiling of the environment.

4.1. Features extraction

Due to the special characteristics of the landmarks, the feature extraction process can be improved by incorporating a bandpass infrared filter to the camera. The process for features detection consists of five steps: acquisition, preprocessing, segmentation, recognition and features extraction. The output of the system is an array of features for each candidate landmark. In the preprocessing phase, the image (Fig. 2(a)) is transformed to facilitate the processing in the next stages. The techniques that have been used are binary thresholding (Fig. 2(b)) and morphological filtering (dilation) (Fig. 2(c)).

As segmentation techniques, the system uses a Canny filter and contour extraction (Fig. 2(d)). The next step is to extract the characteristics of each region:

- *Perimeter*: number of pixels in the perimeter.
- *Centroid*: coordinates of the center of gravity.
- *Radius*: centroid distance to the center of the image.
- *Azimuth*: orientation of an object in the image with respect to the x-axis.

If a light is pointing directly to the camera, then the acquired image will be saturated (Fig. 3(a)). In such cases, a big blob can be detected and the image has to be processed again using a higher threshold (Fig. 3(b)). This situation can be very frequent when lights are quite close to the camera.

4.2. Camera model

The camera model describes how a 3-dimensional scene is transformed into a 2-dimensional image. The standard model is the *Pin-Hole*, which projects the scene on a flat retina, but it is limited to cameras with FOV $\ll 180°$. The other cameras require a model based on a spherical retina. In our system, we have used a projection model developed by Pajdla and Bakstein [28] that indicates how a point (B) in a 3-dimensional Cartesian reference system can be transformed to the coordinates of a pixel in a 2-dimensional image. The model requires the calculation of two angles. On one hand, θ (Fig. 4) is the angle formed between the optical axis of the camera and the beam. This beam is the line from the origin of coordinates of the camera to point B. On the other hand, φ is the angle between the x-axis and the projection of the beam on the x–y plane. Next, the distance r (Fig. 4) from the image center (u_0, v_0) to the coordinates of point B in the image (u_B, v_B) is estimated as:

$$r = a * \tan\frac{\theta}{b} + c * \sin\frac{\theta}{d}, \tag{1}$$

where $a, b, c,$ and d are parameters of the model. This function makes it possible to calculate the coordinates of the point in the image (u_B, v_B) depending on the azimuth (φ) and the elevation (θ) (Fig. 4):

(a) Saturated image. (b) Postprocessed image.

Fig. 3. Postprocessing phase.

Fig. 4. Projection of a 3-dimensional point B on the image reference system using the omnidirectional camera model.

$$\begin{aligned} u_B &= u_0 + r * \cos\varphi \\ v_B &= \beta * (v_0 + r * \sin\varphi) \end{aligned} \tag{2}$$

where β is the ratio between the width and the height of a pixel. These equations (Eqs. (1) and (2)) define the omnidirectional camera model.

4.3. Landmark measurements

The map that has been used for localization consists of a list of landmarks, and each landmark is defined by its 3-dimensional coordinates in a Cartesian reference system. As described in Fig. 1 (line 16), given a pose of the robot (x_t), the set of features (F) that have been extracted from an image, and the set of landmarks (M, the map of the environment), the measurement model has to calculate the conditional probability that given the pose and the map, F is the set of detected features at time t. To calculate this probability, we need to have the elements of sets M and F both in the same coordinate system. Therefore, and using the camera model, all the landmarks $B_i \in M$ have to be transformed.

Fig. 5. Maximum likelihood data association.

A landmark projection is the calculation of the pixel coordinates in the image (u_{B_i}, v_{B_i}) for landmark B_i, given its coordinates in the world ($\mathbf{B}_i^W$) and the coordinates of the camera ($\mathbf{C}^W$) in the same reference system. First, the landmark coordinates must be transformed from world coordinates ($\mathbf{B}_i^W$) to the camera reference system ($\mathbf{B}_i^C$) with the rotation matrix R^C:

$$\mathbf{B}_i^C = R^C \cdot \mathbf{B}_m^W - \mathbf{C}^W. \tag{3}$$

From $\mathbf{B}_i^C$, we obtain the elevation (θ) and the azimuth (φ) angles by applying the traditional Euclidean transformations (Fig. 4). Finally, to obtain the landmark projection $(u_{B_i^C}, v_{B_i^C})$, Eqs. (1) and (2) are applied. The image composed of all the landmark projections is called the map projected image. The measurement model returns the value of the conditional probability $P(F \mid x, M)$. Assuming conditional independence between features, it can be calculated as:

$$P(F \mid x, M) = \prod_j P\ F_j \mid x, M \tag{4}$$

and, therefore, the probability for each feature F_j can be estimated independently.

In order to calculate the conditional probability, it is necessary to decide the correspondence between each feature F_j and each landmark B_i. This has been solved with the maximum likelihood data association algorithm (Fig. 5). The calculation of $P(F \mid x, M)$