Active Appearance-Based Robot Localization Using Stereo Vision

J.M. Porta, J.J. Verbeek and B.J.A. Kröse
IAS Group, University of Amsterdam, Kruislaan 403, 1098SJ, Amsterdam, The Netherlands
{*porta,jverbeek,krose*}*@science.uva.nl*

Abstract. A vision-based robot localization system must be robust: able to keep track of the position of the robot at any time, even if illumination conditions change and, in the extreme case of a failure, able to efficiently recover the correct position of the robot. With this objective in mind, we enhance the existing appearance-based robot localization framework in two directions by exploiting the use of a stereo camera mounted on a pan-and-tilt device. First, we move from the classical passive appearance-based localization framework to an active one where the robot sometimes executes actions with the only purpose of gaining information about its location in the environment. Along this line, we introduce an entropy-based criterion for action selection that can be efficiently evaluated in our probabilistic localization system. The execution of the actions selected using this criterion allow the robot to quickly find out its position in case it gets lost. Secondly, we introduce the use of depth maps obtained with the stereo cameras. The information provided by depth maps is less sensitive to changes of illumination than that provided by plain images. The main drawback of depth maps is that they include missing values: points for which it is not possible to reliably determine depth information. The presence of missing values makes Principal Component Analysis (the standard method used to compress images in the appearance-based framework) unfeasible. We describe a novel Expectation-Maximization algorithm to determine the principal components of a data set including missing values and we apply it to depth maps. The experiments we present show that the combination of the active localization with the use of depth maps gives an efficient and robust appearance-based robot localization system.

Keywords: localization, appearance-based modeling, active vision, depth maps, stereo vision.

1 Introduction

One of the most basic abilities for a mobile robot is that of localization, i.e. to be able to determine its own position in the environment. To localize, the robot needs some kind of *representation* of the environment. In the literature on mobile robots, this representation of the environment comes basically in two flavors: *explicit* or *implicit*.

The *explicit* (or *geometric*) representations are based on maps of free spaces in the environment [20, 51, 28] (using, for instance grid maps or polygonal-based representations) or are based on maps with locations of distinct observable objects (i.e., landmarks) [44]. This approach relies on the assumption that geometric information (shape and position of obstacles, landmarks, etc.) can be extracted from the robot's sensor readings. However, the transformation from sensor readings to geometric information is, in general, complex and prone to errors, increasing the difficulty of the localization problem.

Figure 1: The robot Lino.

As a counterpart, the *implicit* (or *appearance-based*, when images are used as a sensory input) representation of the environment has attracted lot of attention recently because of its simplicity. This work derived from the seminal work on object recognition by Murase and Nayar [54]. In this paradigm, the environment is not modeled geometrically but as an appearance map that includes a collection of sensor readings obtained at known positions. The advantage of this representation is that the raw sensor readings obtained at a given moment can be directly compared with the observations in the appearance-based map. This approach has been applied to robot localization using range-based sensors [5, 13] and omnidirectional images [31, 39].

A comparison between the two families of localization methods using vision as sensory input can be found in [61], showing that appearance-based methods are more robust to noise, occlusions and changes in illumination (when a edge detector is used to pre-process the images) than geometric based-methods. For this reason, and for its efficiency and simplicity, we decided to use this localization paradigm in Lino, a service robot developed as a part of the European project "Ambience" [3]. In this project, the robot is the personification of an intelligent environment: a digital environment that serves people, that is aware of their presence and context, and that is responsive to their commands, needs, habits, gestures and emotions. The robot can get information from the intelligent environment to make it available to the user and, the other way around, the user can ask for services from the digital environment in a natural way by means of the robot. Lino is equipped with a 'real' head with three degrees of freedom (left/right, up/down and approach/withdraw). The head has dynamic mouth, eyes and eyebrows since this makes the interaction more attractive and also more natural.

The information about its location is fundamental for Lino. Modules such as navigation and environment awareness are based on information about the robot's position. If the location information is wrong, most of the Lino's modules must be stopped until the robot finds out its position again. Thus, to achieve a smooth operation of the robot, the localization module must be as robust as possible. In the extreme case the robot gets lost (i.e., completely uncertain about its location), the localization system must recover the position of the robot as fast as possible so that the services offered by the robot are interrupted as short as possible.

It is well known that appearance-based methods are quite brittle in dynamic environments since changes in the environment can make the matching of the current observation with the images in the appearance-based map difficult or even impossible. This is specially critical when localization is based on

global features extracted, for instance, from omnidirectional images [25, 31, 40]. A possible solution to this problem is to extract local features (i.e., corners, lines, uniform regions, etc.) [11, 42]. We can also obtain local features using a normal camera instead of an omnidirectional one. Due to the limited field of view of normal cameras, we can only get features in a reduced area in the environment and modifications in the environment would only be relevant if the camera is pointing toward them. If this is the case, we can rotate the cameras to get (local) features in other orientations hopefully not affected by the environment modifications. This is the solution we adopted in the Lino project. Moreover, in this paper, we present an entropy-based action evaluation criterion that we use to select the cameras movement (rotations and possibly displacements of the robot) that is likely to provided better information on the robot's position. We show that, using this criterion, the robot can efficiently recover it location when the localization system is in its initial stages or when a failure forces the robot to re-localize. The active localization problem has been addressed by some authors before [34, 35, 38] and different entropy-based criteria for action selection similar to the one we describe can be found in the literature for object recognition [1], environment modeling [64] or even for robot localization [6, 22, 16, 46]. The difference is that, in our case, the entropy-based evaluation can be computed more efficiently than in existing approaches.

The use of local features increases the robustness of the localization system in front of local changes in the environment. However, if global changes occur, the robustness of the localization system would decrease. One of the global changes that typically affect images (and, thus, the features derived from them) are changes in illumination. Thus, to achieve a robust system, we must deal appropriately with variable lighting conditions. One possible solution to this problem is to include in the appearance-based map images taken in different illumination settings. These sets of images can be obtained on-line or using rendering techniques on a geometric model of the environment. However, not all possible illumination setups can be devised when defining the appearance map and this clearly limits the usefulness of this approach. A more general solution is to pre-process the images to compensate for the effect of illumination on the extracted features. In this line, techniques such as histogram equalization or gradient filters have been used to obtain images that are, up to a given point, illumination-independent [32, 53]. In this paper, we introduce the use of depth maps for appearance-based localization. This is possible thanks to the use of the stereo camera we mounted on Lino's head. We show that the information provided by stereo depth maps is less sensitive to changes of illumination than that provided by plain images, even when typical illumination compensation techniques are used. The advantage of this setup is that, in contrast to standard laser range finders, which provide a 1-D depth map, using stereo vision a more informative 2-D depth map is determined. Depth information has been previously used in many application including mapping techniques based on map-matching [56], in legged robot navigation [7] and it is also common in the geometric-based localization approaches to increase the robustness of the landmark identification [10, 45, 55]. In this paper, we introduce a method to use the same kind of information also to increase the robustness in localization but, in our case, within the appearance-based approach.

The two techniques we introduce in this paper (the use of active vision and the use of disparity maps for appearance-based localization) are embedded in a probabilistic localization framework. We first we review this basic localization framework (Section 2). Next, in Section 3, we introduce the entropy-based active vision mechanism. After this, in Section 4 we describe how is it possible to use disparity maps for appearance-based localization and how this fits in the general localization framework. Section 5 describes some experiments that validate our two contributions and, finally, in Section 6, we summarize our work and we extract some conclusions out of it.

2 Appearance-based Robot Localization

In the following subsections, we introduce the three basic elements of the probabilistic localization system: the Markov localization model, the sensory model of the environment, and the auxiliary particle filter.

2.1 A Probabilistic Model for Robot Localization

Nowadays, the formalization of the robot localization problem in a probabilistic way has become a standard [68]. In our particular case, we assume that the orientation of the camera with respect to the robot is given with sufficient accuracy by the pan-tilt device and the absolute pose of the camera is considered as a stochastic (hidden) variable x. The localization method aims at improving the estimation of the pose x_t of the camera at time t taking into account the movements of the robot and its head $\{u_1, \ldots, u_t\}$ and the observations of the environment taken by the robot $\{y_1, \ldots, y_t\}$ up to that time. In our notation, the Markov process goes through the following sequence $x_0 \xrightarrow{u_1} (x_1, y_1) \xrightarrow{u_2} \ldots \xrightarrow{u_t} (x_t, y_t)$. So, we want to estimate the posterior $p(x_t|\{u_1, y_1, \ldots, u_t, y_t\})$. The Markov assumption states that this probability can be updated from the previous state probability $p(x_{t-1})$ taking into account only the last executed action u_t and the current observation y_t. Therefore, we only have to estimate $p(x_t|u_t, y_t)$. Applying Bayes we have that

$$p(x_t|u_t, y_t) \propto p(y_t|x_t)\, p(x_t|u_t), \tag{1}$$

where the probability $p(x_t|u_t)$ can be computed propagating from $p(x_{t-1}|u_{t-1}, y_{t-1})$

$$p(x_t|u_t) = \int p(x_t|u_t, x_{t-1})\, p(x_{t-1}|u_{t-1}, y_{t-1})\, dx_{t-1}. \tag{2}$$

Equations 1 and 2 define a recursive system to estimate the position of the camera/robot.

The probability $p(x_t|u_t, x_{t-1})$ for any couple of states and for any action is called the *action model* and it is assumed as known (i.e., inferred from odometry). On the other hand, $p(y_t|x_t)$ for any observation and state is the *sensor model* that has to be defined for each particular problem (see Section 2.2).

If the localization system can deal with a uniform initial distribution $p(x_0)$ then we have a system able to perform *global* localization. If the system requires $p(x_0)$ to be peaked on the correct robot's position, then we have a *position tracking* localization system that basically compensates the incremental error in the robot's odometry [68].

2.2 Sensor Model

An open problem in the just described framework is how to compute the sensor model $p(y|x)$. The sensor model is the environment representation used by the robot to localize itself. As mentioned before, due to its simplicity, we advocate for an appearance-based sensor model. Such a model must provide the probability of a given observation over the space of configurations of the robot according to the similarity of the current observation with those in the training set: the more similar the current observation with a given training point, the larger the probability of the robot to be close to that training point. A problem with images is their high dimensionality, resulting in large storage requirements and high computational demands. To alleviate this problem, Murase and Nayar [54] proposed to compress images z (with size D) to low-dimensional feature vectors y (with size d) using a linear projection

$$y \;=\; W\, z. \tag{3}$$

The $d \times D$ projection matrix W is obtained by principal component analysis (PCA) [33] on a supervised training set, $T = \{(x_i, z_i)|\, i \in [1, N]\}$, including images z_i obtained at known states x_i. PCA (also known as Karhunen-Loeve transform in the context of signal processing) is a commonly used data compression technique that aims at capturing the subset of dimensions that explain most of the variance of the data. Additionally, the PCA provide the optimal (in the least square sense) linear approximation to the given data set. PCA has been widely been used in robotics and computer vision for tasks such as face recognition [62], hand-print recognition [52], object modeling [54] and, more recently, for state compression in controllers based in partially observable Markov decision processes [60]. The numerically most stable

method to compute the PCA is using Singular Value Decomposition (SVD). SVD is a procedure that decompose a $D \times N$ ($D \geq N$) matrix Z as

$$Z = USV^\top,$$

with U the $D \times N$ orthonormal matrix of left singular vectors of Z, S a $N \times N$ diagonal matrix with the singular values, and V a $N \times N$ orthonormal matrix with the right singular vectors of Z. In our case, Z contains the images in the training set arranged in columns. A usual way to compute the SVD is by first determining V and S by diagonalizing $Z^\top Z$

$$Z^\top Z = VS^2V^\top$$

and, then, computing U as

$$U = ZVS^{-1}.$$

If the rows in Z are zero-mean, then $ZZ^\top$ is the covariance of the data and it can be rewrite as

$$ZZ^\top = (USV^\top)(USV^\top)^\top = USV^\top VSU^\top = US^2U^\top.$$

Thus, U and S^2 are a diagonalization of the covariance matrix: U include the eigenvectors and S^2 the eigenvalues, that are proportional to the variance in the dimension expanded by each associated vector in U. We want the projection matrix W to capture as much variance (or standard deviation) as possible with as less dimensions as possible and, thus, we have to include in the rows of W the singular vectors in U associated with the d largest singular values. The ratio

$$\sigma = \frac{\sum_{i=1}^{d} \sigma_i^2}{\sum_{i=1}^{N} \sigma_i^2},$$

with σ_i the singular values sorted in descending absolute value, is generally used as an indicator of the variance of the data captured by the d main singular vectors. In our applications, d is chosen so that v is about 0.75 which, in general, means a value of d below 15.

A classical method to approximate the sensor model $p(y|x)$ from a supervised training set is using kernel smoothing [57, 71, 40]. This method uses a multivariate kernel for each training point and, thus, each evaluation of the sensor model scales linearly with the size of the training set. This makes kernel smoothing inefficient for practical applications. Moreover, kernel smoothing only works well in low dimensions (e.g., less than five). If we compress the images via PCA to such a reduced set of features, we might lose lots of information useful for localization. For these reason, as proposed by Vlassis *et al.* in [70], we use a mixture to approximate the sensor model where $p(y|x)$ is computed using only the J points in the training set that are closer to the current observation (after the corresponding PCA dimensionality reduction). Thus, we have that

$$p(y|x) = \sum_{j=1}^{J} \lambda_j \, \phi(x|x_j), \tag{4}$$

with x_j the sorted set of nearest-neighbors (i.e., the set of training points x_i with a set of features y_i more similar to y, with $\|y - y_k\| \leq \|y - y_l\|$ if $k < l$), and ϕ a Gaussian with a standard deviation equal to half of the average distance between adjacent training points, independently computed for X, Y and the orientation. Finally, λ_j is a set of weights that favor nearest-neighbors closer to y that, in our implementation, is computed as

$$\lambda_j = \frac{2(J - j + 1)}{J(J + 1)}.$$

The determination of the sensor model for a given observation y can be performed with cost

$$O(Dd + J \, \log(N))$$

with D the number of pixels of image z, d the number of features extracted from each image, J the number of nearest neighbors used in the sensor model and N the size of the training set. The first factor of the cost $O(Dd)$ is due to the linear projection of the image to d features and the second factor $O(J \log(N))$ is the determination of the nearest-neighbors, provided the training set is organized in a KD-tree to speed up the identification of training points similar to the current observation. Once the sensor model parameters are determined, the evaluation for a given state scales with J, that is much smaller than the size of the training set N and that, as mentioned, would the cost in case we use a kernel smoothing based sensor model.

2.3 The Auxiliary Particle Filter

The probabilistic framework presented before is a general formulation and, for each particular case, we have to devise how to represent and update the probability distribution on the state, $p(x_t|u_t, y_t)$. In the update, the main problem we have to confront is how to compute the integral of equation 2.

If we assume $p(x_t|u_t, y_t)$ to be a Gaussian, we can use a simple representation for the probabilities (its mean and its covariance matrix would be enough) and the probability update can be done using a Kalman filter [44]. Systems based on this type of representation are effective avoiding the error on the robot's location to grow without any bound. Additionally, the simplicity of the model used on these systems allows for formal demonstration on the convergence of the localization (and of the associated mapping process, if any). However, using a single Gaussian, it is not possible to track more than one hypothesis at the same time and, due to this, the Kalman-based localization systems are unable to deal with the global localization problem.

Multi-Gaussian probability representations [12, 2, 30, 37] can track more than one hypothesis simultaneously, but they still rely on restrictive assumptions on the size of the motion error and the shape of the robot's position uncertainty.

Probabilistic occupancy grids [5, 23, 66, 60] and particle filters [68, 70] can represent distributions with arbitrary shapes and, thus, they can be used to solve the global localization problem. In the probabilistic occupancy grids framework, the area where the robot is expected to move is discretized in small cells and the system maintains the probability for the robot to be in each one of these cells. This approach is quite expensive from a computational point of view. In the particle filter framework, the robot position is estimated using a set of discrete samples, each one with an associated weight to represent its importance. In this way, computational resources are focused on the areas of the configuration space where the robot is more likely to be. Additionally, the computational cost can be adjusted to the available resources by initially defining more or less particles or using techniques that on-line adapt the size of the sample set [41].

More formally, in a particle filter, the continuous posterior $p(x_{t-1}|u_{t-1}, y_{t-1})$ is approximated by a set of I random samples, called particles, that are positioned at points x^i_{t-1} and have weights π^i_{t-1}. Thus, the posterior is

$$p(x_{t-1}|u_{t-1}, y_{t-1}) = \sum_{i=1}^{I} \pi^i_{t-1}\, \delta(x_{t-1}|x^i_{t-1}),$$

where $\delta(x_{t-1}|x^i_{t-1})$ represents the delta function centered at x^i_{t-1}. Each one of the particles can be seen as a possible hypothesis on the cameras pose. Therefore, the estimation of the cameras/robot pose can be defined as a likely value associated with the set of particles (for instance, its weighted mean).
Using the above approach, the integration of equation 2 becomes discrete

$$p(x_t|u_t) = \sum_{i=1}^{I} \pi^i_{t-1}\, p(x_t|u_t, x^i_{t-1}), \tag{5}$$

and equation 1 reads to

$$p(x_t|u_t, y_t) \propto p(y_t|x_t) \sum_{i=1}^{I} \pi_{t-1}^i \, p(x_t|u_t, x_{t-1}^i).$$

The central issue in the particle filter approach is how to obtain a set of particles (that is, a new set of states x_t^i and weights π_t^i) to approximate $p(x_t|u_t, y_t)$ from the set of particles x_{t-1}^i, $i \in [1, I]$ approximating $p(x_{t-1}|u_{t-1}, y_{t-1})$. The usual Sampling Importance Resampling (SIR) approach [18, 29] sample particles using the motion model $p(x_t|u_t, x_{t-1}^i)$ then, it assigns a new weight to each one of these particles proportional to the likelihood $p(y_t|x_t)$ and, finally, it re-samples particles using these new weights in order to make all particles weights equal. The main problem of the SIR approach is that it requires lots of particles to converge when the likelihood $p(y|x)$ is too peaked or when there is a only a small overlap between the prior and the posterior likelihood. In our case, the sensor model defined in previous section is not peaked at all, on the contrary, the Gaussian mixture of equation 4 is rather smooth. Thus, in a perfectly static environment, the SIR technique would provide good results. However, we expect our environment to be not completely static: occlusions, small displacements of objects in the environment, people walking around, etc. would affect the observation model. In the worst case, this results in severe outliers and, in the best case, this produces partially wrong matches (i.e., training observations close to the current one but not as close as they would be in case of non-noisy observations). In this circumstances, the SIR technique would require of a large number of particles to achieve a robust localization, with the consequent increase in the execution time of the system.

Vlassis *et al.* [70] propose a more efficient but still robust alternative: to use an auxiliary particle filter. In the auxiliary particle filter [57] the sampling problem is solved in an elegant way by inserting the likelihood inside the mixture

$$p(x_t|u_t, y_t) \propto \sum_{i=1}^{I} \pi_{t-1}^i \, p(y_t|x_t) \, p(x_t|u_t, x_{t-1}^i).$$

Then $p(x_t|u_t, y_t)$ can be regarded as a mixture of the I transition components $p(x_t|u_t, x_{t-1}^i)$ with weights $\pi_{t-1}^i \, p(y_t|x_t)$. Therefore, sampling from $p(x_t|u_t, y_t)$ can be achieved just selecting one of the components j with probability $\pi_{t-1}^i \, p(y_t|x_t)$ and then sampling from the corresponding component $p(x_t|u_t, x_{t-1}^j)$. Sampling is performed in the intersection of the prior and the likelihood and, consequently, particles with larger prior and larger likelihood (even if this likelihood is small in absolute value) are more likely to be used to re-estimate the position of the robot.

Observe that the state x_t involved in $p(y_t|x_t)$ is unknown at the moment the sampling is performed (it is exactly the state we are trying to approximate). Instead of x_t, a likely value associated with the i-th transition component is used, for instance its mean μ_t^i

$$p(x_t|u_t, y_t) \propto \sum_{i=1}^{I} \pi_{t-1}^i \, p(y_t|\mu_t^i) \, p(x_t|u_t, x_{t-1}^i). \tag{6}$$

After the set of states for the new particles particles is obtained using the above procedure, we have to define their weights. This is done using

$$\pi_t^j \propto \frac{p(y_t|x_t^j)}{p(y_t|\mu_t^{i_j})},$$

where i_j is the transition component from which the particle j has been sampled.

The whole filter update is linear with the number of particles and the number of nearest-neighbors used in the sensor model. With this, the total cost of each localization step is

$$O(Dd + J \, \log(N) + IJ)$$

with D the number of pixels of the images, d the number of features for each image, J the number of nearest neighbors used in the sensor model, N the size of the training set and I the number of particles. One of the most efficient localization systems described in the literature is FastSLAM introduced by Montemerlo *et al.* [50]. This system performs concurrent map building and localization, while in our approach the map is assumed as given. However, if we take into account the cost of the localization part (and the data association that is equivalent to our sensor model definition) but not the cost associated with the mapping, FastSLAM scales linearly with the number of particles and logarithmically with the size of the map, which is roughly the cost of our system.

Despite the robustness introduced by the auxiliary particle filter, we can not discard catastrophic errors in the localization: occlusions or temporary modifications of the environment can lead to wrong matches to define the sensor model for quite a long period in which odometry is the only source of information that can be used to keep track of the robot's position. The consequence is a possible divergence between the correct robot's position and that provided by our localization system. A similar situation occur if the robot is *kidnapped*. A kidnap is any displacement that violate the action model: wheel slippery produced while the robot is trying to advance but it is colliding with an obstacle, displacements of the robot performed by the user, etc. Our localization system must include a mechanism to detect and correct those error situations. Many approaches to this problem can be found in the literature. For instance, [47, 43] propose to sample a portion of the particles directly form the posterior likelihood, [67] introduce the Mixture-CML where normal sampling is used in conjunctions with a dual sampling where particles are drawn from the sensor model and then odometry is used to asses its compliance, and [65] describe a particle filter in which the state is extended with information about coherence in the sensor model along time to cope in a unified way with non-Markovian noise in the environment and with kidnap situations. In the system described in this paper, we use a simple but effective strategy consisting in re-sampling all particles when a mismatch between the prior and the likelihood is detected for a while. The agreement between the prior and the likelihood is measured as the sum of the likelihood for the particles as in equation 6

$$\sum_{i=1}^{I} \pi_{t-1}^{i}\, p(y_t|\mu_t^i)\, p(x_t|u_t, x_{t-1}^i),$$

If this sum is below a user defined threshold (10^{-6} in our implementation), the observation is considered an outlier and it is not used in the particle update (thus, only the motion model is used). If this situation is repeated for many consecutive time slices (10 in our tests), the particles are re-sampled from scratch using the sensor model of the first available observation. To ensure a fast re-identification of the correct robot's position the active vision procedure described in next section is triggered.

The existence of a error recovery mechanism and the fact that we never sample uniformly over the whole configuration space allow us to reduce the number of particles, increasing the performance of the system. However, a trade off must be reached since the use of too few particles would result in a constant activation of the active re-localization mechanism and in a continuous interruption of the normal behavior of the robot.

3 Active Localization

In general, the localization model outlined in the previous section is implemented as a *background* process: the localization module keeps track of the position of the robot using the observations obtained along the robot's path decided by a navigation module. However, when the robot is uncertain about its position it makes little sense to continue with the navigation task and it is more advisable to first reduce the uncertainty in the robot position and, then continue with the navigation. Therefore, in some cases, the robot's actions have to be determined by localization criteria and not to navigation related ones. The question in that case is how to determine the appropriateness of actions as far as localization is concerned.

Next, we describe a general criterion to compute this appropriateness for each action and how to implement this criterion in our localization framework.

3.1 Entropy-based Action Selection

We assume that, at a given moment, the robot can execute a discrete set of actions $\{u_1, \ldots, u_n\}$. The usefulness of one of these actions u, as far as localization is concerned, can be determined examining the probability $p(x_{t+1}|u, y_{t+1})$. An ideal action would allow the robot to find out its position without any doubt, that is, would produce a probability $p(x_{t+1}|u, y_{t+1})$ with a single peak centered at the correct position of the camera. Such a probability distribution would have a very low entropy $H(u, y_{t+1})$ defined as

$$H(u, y_{t+1}) = -\int p(x_{t+1}|u, y_{t+1}) \ \log p(x_{t+1}|u, y_{t+1}) \, dx_{t+1}.$$

To compute the entropy of a given action u, we integrate over all possible observations

$$H(u) = \int H(u, y_{t+1}) \, p(y_{t+1}|u) \, dy_{t+1}.$$

Following the reasoning presented at [22], the posterior involved in $H(u, y_{t+1})$ can be written as

$$p(x_{t+1}|u, y_{t+1}) = \frac{p(y_{t+1}|x_{t+1}) \, p(x_{t+1}|u)}{p(y_{t+1}|u)},$$

and, consequently, the entropy for a given action becomes

$$H(u) = -\int\int p(y_{t+1}|x_{t+1}) \, p(x_{t+1}|u) \ \log \frac{p(y_{t+1}|x_{t+1}) \, p(x_{t+1}|u)}{p(y_{t+1}|u)} \, dx_{t+1} \, dy_{t+1}. \tag{7}$$

At any moment, the best action u to be executed next (as far as localization is concerned) is the one with lower entropy $H(u)$ since, the lower $H(u)$, the more informative the action u is likely to be.

3.2 Implementation

The localization framework presented in Section 2 allows us to efficiently implement the action-selection formalism just described. The basic idea is to exploit the particle filter and the appearance-based training set to approximate the double integral of equation 7. We have to discretize the distribution on the camera poses $p(x_{t+1}|u)$ after executing action u and the distribution on the observations the robot will made when it reaches its new placement $p(y_{t+1}|x_{t+1})$. Additionally, we have to approximate the probability for each observation y after action u, $p(y_{t+1}|u)$.

First, we discretize the probability $p(x_{t+1}|u)$. Using equation 5 we have that

$$p(x_{t+1}|u) = \sum_{i=1}^{I} \pi_t^i \, p(x_{t+1}|u, x_t^i).$$

In the absence of any other information (i.e., new observations) the probability on the pose of the camera $p(x_{t+1}|u)$ after the execution of action u can be approximated sampling on the action model $p(x_t|u_t, x_{t-1})$ for each one of the particles x_{t-1}^i approximating the current pose of the camera. We denote the state of particle resulting from the sampling process on each $p(x_t|u_t, x_{t-1}^i)$ as $x_t^i(u)$. Using this new set of particles, we have that

$$p(x_{t+1}|u) = \sum_{i=1}^{I} \pi_t^i \, \delta(x_{t+1}|x_t^i(u)).$$

Often, the above resampling is just a shift and a blur of the particles representing $p(x_{t+1}|u, x_t^i)$. In the particular case in which we only move the head of the robot, the re-sampling process becomes a simple shift of the particles x_t^i since head movements do not add error on the position of the camera. Another particular case is that of actions that do not affect the pose of the cameras/robot (for instance, movements around the tilt degree of freedom of the head). In this case we have that

$$x_t^i(u) = x_t^i,$$

and thus, particles at time t, x_t^i, can be used directly to approximate $p(x_{t+1}|u)$, without any modification.

Using the above we can re-write equation 7 as

$$H(u) \approx -\int \sum_{i=0}^{I} \pi_t^i \, p(y_{t+1}|x_t^i(u)) \, \log \frac{\pi_t^i \, p(y_{t+1}|x_t^i(u))}{p(y_{t+1}|u)} \, dy_{t+1},$$

where the integral over dx_{t+1} vanished. Now, we have to discretize the integration over the observations.

The set of states $x_t^i(u)$ is a sample on the possible poses of the camera after executing action u. The set of features observed at each one of these poses can be inferred using the training set: the observation for each position $x_t^i(u)$ would be similar to the observation y obtained in the training point x that is as close as possible to $x_t^i(u)$. We take the set of views (Y_u) obtained in this way from all states $x_t^i(u)$ as a representative sample on the likely observations after executing action u.

If the training set is sampled on a uniform grid over the space of configuration of the robot, finding the closest training point to a given state $x_t^i(u)$ is straightforward and can be done in constant time. If this is not the case, a KD-three structure can be used to perform this search in logarithmic time in the number of training points.

With the above, we achieve a discretization on the possible observations. Now, for each one of the observations $y \in Y_u$ we have to define $p(y|x_t^i(u))$. This can be done, as in Section 2.2, using a nearest-neighbor approach. So,

$$p(y|x_t^i(u)) = \sum_{j=1}^{J} \lambda_j \, \phi(x_t^i(u)|x_j), \tag{8}$$

for x_j the J training points with observations more similar to y.

Observe that, we only compute equation 8 for sets of features y stored in the training set. Consequently, the process of finding the nearest-neighbors to y, x_j, and the corresponding weights, λ_j, can be pre-computed for all the observations in the training set, saving a large amount of time in the on-line execution of the entropy evaluation algorithm.

Finally, to complete the approximation of equation 7 we define the probability on observation y after action u ($y \in Y_u$) as

$$p(y|u) = \sum_{k=1}^{K} \pi_t^{i_k}, \tag{9}$$

with $\{i_1, \ldots, i_k\}$ the set of particles that advocate for observation y. In a situation where particles are spread all over the configuration space of the robot, each particle is likely to propose a different observation y. However, in case where particles concentrate in few clusters many particles can propose the same observation to be included into Y_u and the weights of all these coincident particles are used to define $p(y|u)$ for that observation.

With the above approximations, the entropy-based evaluation of an action u becomes

$$H(u) \approx -\sum_{y \in Y_u} \sum_{i=1}^{I} \left[\pi_t^i \, \sum_{j=1}^{J} \lambda_j \, \phi(x_t^i(u)|x_j) \, \log \frac{\pi_t^i \, \sum_{j=1}^{J} \lambda_j \, \phi(x_t^i(u)|x_j)}{\sum_{k=1}^{K} \pi_t^{i_k}} \right]. \tag{10}$$

Input:
A set of candidate actions U.
The current set of particles $\{(\pi_t^i, x_t^i) \mid i \in [1, I]\}$.
The training set $T = \{(x^i, y^i) \mid i \in [1, N]\}$.
Output:
The most informative action $u*$

```
1:  h* ← ∞
2:  For each u ∈ U
3:      Y_u ← ∅
4:      For each particle (π_t^i, x_t^i)
5:          (x, y) ∈ T with minimum ‖x − x_t^i(u)‖
6:          If y ∈ Y_u then
7:              p(y|u) ← p(y|u) + π_t^i
8:          else
9:              p(y|u) ← π_t^i
10:             Y_u ← Y_u ∪ {y}
11:         Endif
12:     Endfor
13:     h ← 0
14:     For each y ∈ Y_u
15:         For each particle (π_t^i, x_t^i)
16:             g ← π_t^i Σ_{j=1}^{J} λ_j φ(x_t^i(u)|x_j)
17:             h ← h − g log(g/p(y|u))
18:         Endfor
19:     Endfor
20:     If h < h* then
21:         h* ← h
22:         u* ← u
23:     Endif
24: Endfor
25: Propose u* for execution
```

Table 1: Algorithm for entropy-based action selection.

The algorithm to evaluate this equation is shown in Table 1. In this algorithm, we identify the most informative action for localization purposes, u^*, out of a set of candidate actions U. We compute the entropy h for each action (lines $14-19$, equation 10) and we select the one with the lowest entropy, h^* (lines $20-23$). To compute h we need to guess the possible observations after the action execution (the set Y_u that is initialized in line 3 and updated in line 10). For each possible observation $y \in Y_u$, we compute the probability of actually observing y (lines 7 and 9, equation 9).

The cost of this algorithm is $O(U\ I^2\ J)$ with U the number of actions considered, I the number of particles, and J the number of nearest-neighbors used to compute the sensor model. To speed up this procedure, we can replace the point $x_t^i(u)$ by its closest point in the training set. In this way, equation 8 can be fully pre-computed and the cost reduces to $O(U\,I^2)$. In any case, the use of the particles to approximate the entropy provides a lower cost than other approaches that discretize the whole configuration space of the robot [22] or the whole field of view of the cameras [64].

The only point that remains to decide is when to use the action selection procedure just described. The particle filter allow us to devise a simple criterion for that since particles not only estimate the robot's

position, but also provide an estimation on the localization error: the variance of the particle centers. Thus, when this variance grows above a given threshold, we consider that the robot is lost, we stop the robot services based on the robot's position and we trigger the action selection procedure to find out the action that should be executed next.

4 Localization using Sparse Disparity Maps

Beside the active vision strategy, the other enhancement to the classical appearance-based localization that can be made thanks to the use of a pan-and-tilt with a stereo camera is to use disparity (i.e., depth) maps for localization. Next, we introduce how disparity maps are defined, how to extract features from them, and how to use these features in our localization framework.

4.1 Disparity maps

We can determine a depth map matching points in images taken by a pair of calibrated cameras mounted on the robot. Given a single image I, the three-dimensional location of any visible object point Q must lie on the line that passes through the center of projection of the camera c and the image of the object point p (see Figure 2). The determination of the intersection of two such lines generated from two independent images is called triangulation and provides the 3-D position of Q w.r.t the cameras. For a review of scene reconstruction from images taken from different views we refer to [21].

To apply triangulation, for each pixel p in one of the images I we have to search for a correspondent point in the other image I'. Usually, the correspondence is done by comparing areas around pixel p with areas around each candidate pixel p'. Epipolar geometry allow us to reduce the search space to a small segment on I'. The pixels p and p' with more similar surroundings are assumed to correspond to different projections of the same point Q in the scene. If the images planes for the two cameras are co-planar, the distance r from the scene point Q to the cameras can be computed as

$$r = \frac{b\, f}{d - d'},$$

where b is the baseline (distance between the two viewpoints), f is the focal length of the cameras, d is the horizontal distance from the projected point p to the center of one of the images, and d' is the same for the other image. The difference $d - d'$ is called disparity. With constant baseline b and focal length f, the disparity is inversely proportional to the depth of point Q. For this reason, instead of working with depth maps it is enough to use disparity maps.

The stereo algorithm we use [36] applies many filters in the process to determine the disparity map both to speed up the process and to ensure the quality of the results. For instance, if the area around pixel p is not textured enough it would be very difficult to find a single corresponding point p' (we are likely to find many points p' with almost the same probability of being the corresponding point of p). For this reason, pixels on low textured areas are not even considered in the matching process. The result of this and other filtering processes is to produce a sparse disparity map: a disparity map where many pixels do not have a disparity value (see Figure 3). This makes the use of standard PCA to determine the projection matrix (see Section 2.2) unfeasible and we have to use more elaborated techniques as the Expectation-Maximization (EM) algorithm introduced in the following sections (for a general overview of the EM algorithm see Appendix I and the references therein).

4.2 Principal Component Analysis of Data Sets with Missing Values via EM

Given a $D \times N$ matrix Z, representing a set of disparity maps $Z = \{z_1, \ldots, z_N\}$, our aim is to find a projection that maps Z onto its d-dimensional principal components space ($d \ll D$). For each disparity

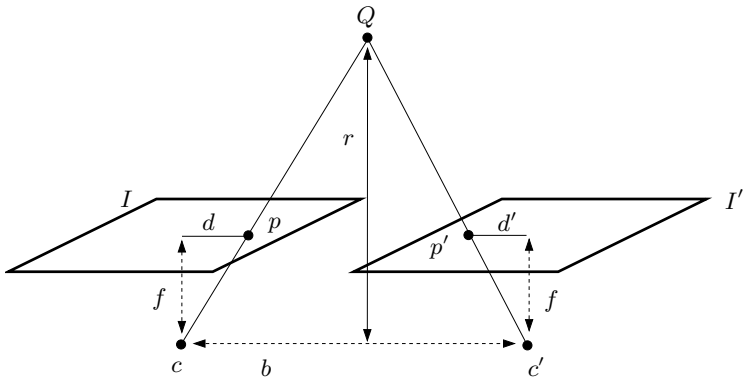

Figure 2: Elements in disparity computation.

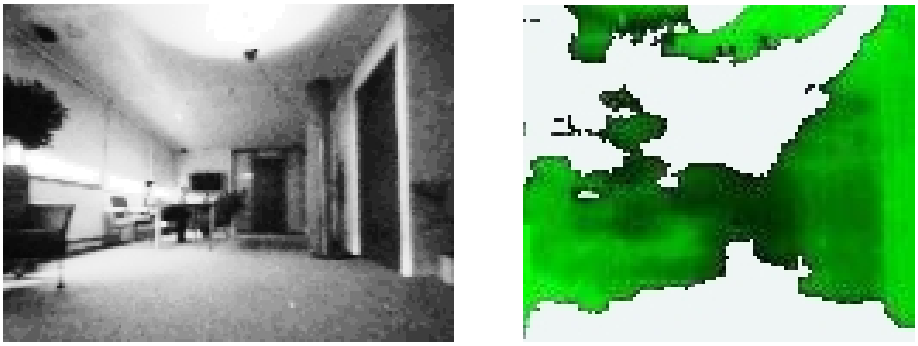

Figure 3: Plain image (left) and the corresponding disparity map (right). In the disparity map, light gray areas are missing values and dark areas correspond to points that are far from the robot.

map z_i, we have a subset of elements that are observed $z_{i,o}$ and a set of missing values $z_{i,h}$. Thus, we want to define the probability

$$p(y_i|z_{i,o}),$$

with y_i the set of principal component features derived from the observed values in disparity map z_i. We do so by first defining $p(y)$ as a Gaussian with zero mean and identity covariance, and $p(z|y) = \mathcal{N}(z; C^\top y, \sigma^2 I)$. The projection density $p(y_i|z_{i,o})$ then follows a Gaussian distribution $\mathcal{N}(y_i; \mu_{y_i}, \Sigma_{y_i})$, where $\Sigma_{y_i}^{-1} = I + C_o C_o^\top/\sigma^2$ and $\mu_{y_i} = \Sigma_{y_i} C_o z_o/\sigma^2$. Our model is described with two parameters, C and σ, that are the ones we want to estimate via EM. Here, C is the $d \times D$ projection matrix from the space of disparity maps to the space of principal components and C_o contains only the columns of C that correspond the observed elements in a given map z_i. Observe that C_o is different for each disparity map and hence so is the covariance matrix Σ_{y_i}.

To find the maximum likelihood projection matrix C via EM, we have to define the E and M steps of the algorithm (see Appendix I). This can be done paying attention only to the observed values z_o, taking the projected features as the hidden variables, and not making inference about the missing pixels. However, the EM algorithm can be made more efficient by estimating y and z_h at the same time.
In the E step the KL divergence to be minimized is

$$KL = \int q(y, z_h) \frac{q(y, z_h)}{p(z_h, y|z_o; C, \sigma)}.$$

In order to obtain an efficient E step we factor q over y and z_h. Thus,

$$KL = \int q(y) \left[\log \frac{q(y)}{p(y|z_o; C, \sigma)} + \int q(z_h) \log \frac{q(z_h)}{p(z_h|y; C, \sigma)} \right].$$

Using free-form optimization, we have

$$q(z_h) \propto \exp \int q(y) \log p(z_h|y; C, \sigma),$$

$$q(y) \propto p(y|z_o; C, \sigma) \ \exp \int q(z_h) \log p(z_h|y; C, \sigma).$$

Both of these densities are Gaussian. The covariance of the $q(z_h)$ is given by $\sigma^2 I$ and the mean for the missing values for disparity map i is given by $z_{i,h} = C_h^\top y_i$. The covariance and the means (collected as columns in Y) of the $q(y)$ for the disparity maps are given respectively by

$$\Sigma_y = [I + \sigma^{-2} C C^\top]^{-1},$$
$$Y = \sigma^{-2} \Sigma_y C Z,$$

where Z is the data matrix with missing values filled in with the new $z_{i,h}$.

Observe that, the covariance matrix Σ_y is common for all data points when computing the set of projections Y. Therefore, we only have to perform one matrix inversion and not one per data point.

With respect to the M step, the relevant part of the likelihood function to be optimized is

$$\Psi_M = \int_{y, z_h} q(y, z_h) \ \log p(z_o, z_h, y; C, \sigma).$$

Analytical manipulation leads to the effective objective to be maximized

$$\Psi_M = ND \log \sigma^2 + \sigma^{-2} \left[\sum_{i=1}^{N} \|z_i - C^\top y_i\|^2 + Tr\{C^\top \Sigma_y C\} \right] + \sigma^{-2} D_h \sigma^2_{\text{old}},$$

with D_h the total amount of missing values in Z and σ_{old} the previous value for σ. From the above, we get the simple updates

$$C = Z Y^\top \left(N \Sigma_y + Y Y^\top \right)^{-1},$$
$$\sigma^2 = \frac{1}{ND} \left[N Tr\{C^\top \Sigma_y C\} + \sum_{i=1}^{N} \|z_i - C^\top y_i\|^2 + D_h \sigma^2_{\text{old}} \right].$$

The E and M steps have to be iterated while there is a large (relative) change in the lower bound of the log-likelihood function that, after the updates, reads to

$$\Psi(C, \sigma) = -\frac{N}{2} \left(D \log \sigma^2 + \text{Tr}\{\Sigma_y\} - \log |\Sigma_y| \right) - \frac{1}{2} \text{Tr}\{Y Y^\top\} + \frac{1}{2} D_h \log \sigma^2_{\text{old}}. \tag{11}$$

A reasonable initialization of C would be that containing d randomly selected disparity maps (with zeros in the missing values) and σ^2 equal to the initial reconstruction error.

Each iteration of the EM steps scales with order $O(dDN)$, assuming $d < N$. Observe that the EM is applied off-line on a training set and, thus, this cost has not to be taken into account in the on-line localization phase. Our approach to finding the maximum likelihood C and σ has the advantage that the guess of the missing values comes essentially *for free* while in other approaches they were obtained

extending the E step with a optimization process in the space of images for every image separately [59]. This is an important saving especially when working with data sets such as sparse disparity map where each point z contains a large proportion of missing values.

A Matlab version of the EM algorithm just described can be downloaded from [69].

Once C and σ are determined, we can on-line compute the set of features y corresponding to a new disparity map z as

$$y = (\sigma^2 I + C_o C_o^\top)^{-1} C_o z_o. \tag{12}$$

The matrix $(\sigma^2 I + C_o C_o^\top)^{-1} C_o$ maps from (observed) disparity values to features and plays the same role as matrix W in equation 3. For vanishing sigma, $(\sigma^2 I + C_o C_o^\top)^{-1} C_o$ is the pseudo inverse of C_o.

The most expensive step in the evaluation of equation 12 process is the product $C_o C_o^\top$ that is $O(Dd^2)$. Because of this, the on-line determination of the features is only feasible for relative small d (and moderated D's).

4.3 Sensor Fusion

Once we have a way to obtain features from disparity maps, it remains the question of how to combine this information with that obtained from intensity to define a unified sensor model.

In ideal conditions (no sensor noise, static environment) the two types of information are redundant, or correlated, since the intensity image and the disparity map are taken in the same position and at the same time. Thus, in this ideal case we have that

$$p(y_i, y_d|x) \approx p(y_i|x) \approx p(y_d|x).$$

If the correlation between the two sources of information is high, only the information provided by one of the sensor modalities, for instance intensity, need to be used in the particle update. In this case, the information provided by disparity just reinforces the information given by the intensity images. On the other hand, if the correlation is low, this indicate that there is a mismatch between the two sources of information. In this situation, it would be hard to decide which one of the sources is more reliable. In the case of changes in illumination, intensity is likely to be an outlier and, in general it's better to use disparity. If the environment has been locally modified (some objects moved, people standing in front of the robot, etc.) intensity tends to be more reliable than disparity. In the worst case, none of the sensor modalities would be correct.

Fortunately we do not need to identify in which situation we are since the particle filter does it for us. As mentioned, the particle filter can deal with outliers and it only uses the sensory hypothesis that are more consistent over time. So, instead of *a priori* trying to select the correct sensory hypothesis, we just take all of them into account in the particle filter update and the most coherent ones are automatically used. Thus, we define the global sensor as a *linear option pool* [26, 9]

$$p(y_i, y_d|x) = w\, p(y_i|x) + (1 - w)\, p(y_d|x) \tag{13}$$

that, using equation 4, reads to

$$p(y_d, y_i|x) = w \sum_{j=1}^{J} \lambda_j\, \phi(x|x_j) + (1 - w) \sum_{k=1}^{K} \lambda_k\, \phi(x|x_k),$$

with $w \in [0, 1]$ and x_j and x_k the closest training points to the features of the current intensity and disparity images respectively. The weight w can be used to balance the importance of the information obtained with each type of sensor. If both information sources are assumed to be equally reliable, we can set $w = 1/2$. The linear pool combination of two probability distributions is a widely used technique.

Figure 4: An image taken in a testing position (left) and the image taken at the closest training point (right).

It is one of the simplest methods to combine different judgments but, as suggested by some empirical studies [8], simpler methods perform better than more complex ones in practice.

The expression on equation 13 for the sensor model is used both in the update of the particles (see equation 6 in Section 2.3) and in the active vision strategy described in Section 3 (equation 8).

Observe that, using equation 13, if the two sensors are highly correlated, we have that $p(y_d|x) \approx p(y_i|x)$ and, as desired, $p(y_i, y_d|x) \approx p(y_i|x)$. If the sensor models are uncorrelated and only $p(y_i|x)$ or $p(y_d|x)$ is correct, the particle filter will work without problem: the outliers are not used and a constant scale factor in the correct part of the likelihood do not affect the update procedure detailed in section 2.3. This works even if the information provided by intensity and that from disparity are partially correct, but none of them is totally right. If both intensity and disparity are outliers, the robot's pose is updated using only odometry and evidence is accumulated on a possible failure of the localization system (caused, for instance, by a kidnap). The failure is more likely when both sensor readings are outliers but they are coherent between them.

5 Experiments and Results

In this section, we describe the experiments we performed to validate our contributions in active localization and in localization using disparity maps. We also report on the global performance of our localization system including the two enhancements we present in this paper.

5.1 Experiments on Active Localization

The objective of the first experiment was to investigate the relation between the localization error and the action selection criteria introduced in Section 3. We tested the localization system in an office environment. We mapped a room of 800×250 cm. with obstacles taking images in the free space every 75 cm. and every 15 degrees. This makes a total amount of about 400 training images. In the experiments, we compress the images using PCA keeping 5 features that preserve up to 70% of the variance of the original images (see Section 2.2), we use 10 nearest-neighbors to approximate the sensor model $p(y|x)$, and we define the initial distribution $p(x_0)$ as uniformly over the configuration space of the robot.

We tested the system placing the robot at positions not included in the training set, rotating the camera as measuring the error and $\|c - a\|$ with c the correct position and a the position estimated by the particle filter (the weighted sum of the particle centers, see section 2.3).

An example of the difference between images at a testing point and images at the closest training point is shown in Figure 4: although the main elements of the scene are similar (the wall with the poster, etc.),

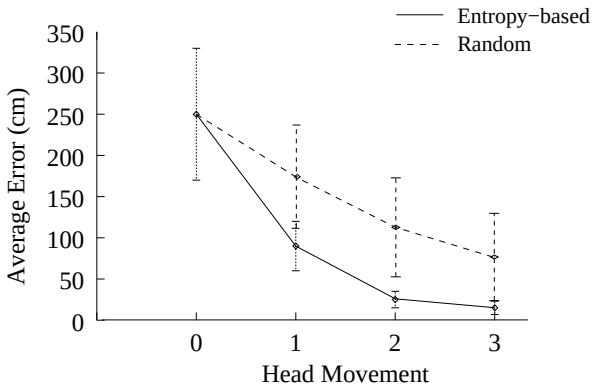

Figure 5: Evolution of the average error (and the standard deviation) w.r.t. the correct position as we get new images.

items on the desk changed from the moment the training set was collected to the moment the testing was performed. In this $X - Y$ point, the differences are not so large for other camera orientations and this is why rotating the camera helps to improve the localization of the robot.

We considered 22 different actions (i.e., different orientations for the camera) and we used up to 150 particles to approximate $p(x_t|u_t, y_t)$. With these figures, the entropy-based action evaluation for all actions takes less than 0.2 seconds in a Pentium 4 at 1.8 GHz.

Figure 5 shows the decrease on the average positioning error as new actions are issued compared with the error decrease when actions are selected at random. The results shown correspond to the average (and the standard deviation) over ten runs placing the camera in different testing positions. We can see that the entropy-based action selection reduces the error in localization faster than the random-based action selection. However, the difference between the two strategies is rather dependent on the environment: in highly aliased environments the entropy-based strategy performs much better than the random one ,but in non-ambiguous environments they perform almost the same. If we consider the estimation a to be correct if the closest training point to a is the same as the closest training point to the correct position c, then the success ratio in localization after 3 camera movements is over 95%.

5.2 Experiments on Localization using Disparity Maps

To test the sensitivity of the features obtained from disparity maps to changes in illumination we acquired three sets of images in a 900×500 cm. environment, but with three different lighting conditions: using tube lights, using bulb lights and using natural light (opening the curtains of the windows placed all along one wall of the lab). For each illumination setup, we collected images every 50 cm. (both along X and Y) and every 10 degrees. This makes a total amount of about 4000 images per illumination setup. We used the set of images obtained with tube lights to determine a projection matrix W with 20 projections vectors (that retain more than 80% of the variation of the training set). The two other sets of images (the one obtained with bulb lights and the one with natural light) were used to assess the effect of the illumination conditions on the features obtained using W. These two tests sets provide changes both in the global intensity of the images and in the distribution of light sources in the scene (that is the situation encountered in real applications).

Each disparity map has $D = 160 \times 120 = 19200$ pixels but about 20% of the pixels on the disparity maps are undefined. We use $N = 100$ randomly sampled images to compute the $d = 20$ principal components of the training set applying the algorithm we describe in Section 4.2.

The principal component of the set of disparity maps are computed in 15 iterations of our EM-based

Image Process	Sensitivity to		
	Translations	**Rotations**	**Light Changes**
Plain Images	0.42 (0.26)	0.95 (0.54)	1.57 (0.52)
Hist. Equalization	0.63 (0.27)	1.40 (0.43)	1.10 (0.31)
Gradient Filter	0.75 (0.28)	1.26 (0.42)	0.91 (0.21)
Disparity Maps	0.85 (0.45)	1.37 (0.57)	0.56 (0.31)

Table 2: Mean (and standard deviation in parenthesis) of the relative change of the features using different image processing techniques and in different illumination conditions.

algorithm with a convergence threshold (relative change on two consecutive maximums of functions Ψ, equation 11) of 0.0001. The execution of this process takes 150 seconds in a Pentium 4 at 1.8 GHz. The on-line computation of the features for a given disparity image involves inverting a 20×20 matrix (see Section 4.2), but it takes less than 0.1 seconds. Therefore, the use of disparity maps for real-time appearance-based localization is completely feasible.

We now need to determine which modality (disparity of intensity) is best for localization. The ideal input for localization would be one with maximal sensitivity w.r.t. translations and rotations, but with a minimum sensitivity to changes in illumination. We measure the sensitivity of features to these three different factors as

$$s_{a,b} = \frac{\|y_a - y_b\|}{\|y_a\|},$$

with y_a and y_b the set of features corresponding to images a and b respectively. To assess the sensitivity to translations on the features, we evaluated the average of $s_{a,b}$ for each couple of images (a,b) taken with the same orientation and the same lighting conditions but at adjacent positions (50 cm). Considering only difference in features for positions that are close each other gives us an estimation of the minimum change (i.e., the minimum sensitivity) in features due to translations. For rotations, we computed the average of $s_{a,b}$ for each couple of images (a, b) taken at the same point and with the same illumination but with adjacent orientations (10 degrees). Finally, to measure sensitivity to illumination conditions, we computed the average of $s_{a,b}$ with a an image take with the training illumination (i.e., using tube lights) and b the image taken at the same position and orientation but with a different lighting setup.

We computed these measures for the features obtained from plain images, images processed with two usual techniques for dealing with illumination related problems: histogram equalization [27, 53] and a gradient-based filter (the Sobel filter [32, 63]) and, finally, for the features obtained from disparity maps.

Table 2 shows the results we obtained for the experiment just described. We can see that plain images provide the features that are less sensitive to translations/rotations and more sensitive to illuminations changes. In the case of features obtained from disparity maps, the sensitivity to changes in illumination is the lowest one. Actually, this is the only case in which the sensitivity to illumination conditions is lower than that due to translations. This means that, in principle, disparity is the best of the three image processing techniques we compare, as far as independence of illumination is concerned. However, we observe that the standard deviation of measures corresponding to disparity maps is, in general, larger than that using other techniques. This is because sometimes the features obtained from disparity maps are quite affected by noise. Due to this sporadic noisy readings features obtained from disparity maps are usually, but not always, the best ones. For this reason, in our localization system disparity maps are not used as the only source of information, but as a complemented to intensity images.

To assess the contribution of using disparity maps in definition of the sensor model, we moved the robot along a pre-defined path in the three different illumination conditions mentioned above: tube lights, bulb lights and natural light. At regular distances along the test path, we took an image and we computed the corresponding sensor model using the J training points with a set of features more similar to those

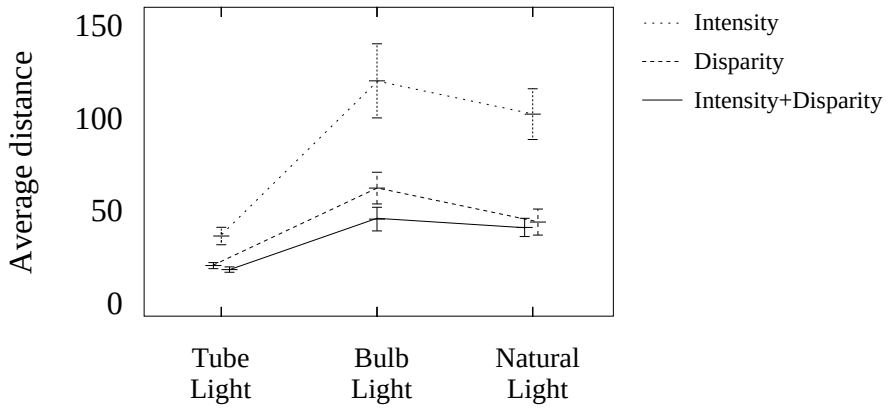

Figure 6: Error in the sensor model in three different illumination conditions, using only intensity images (dotted line), only disparity maps (dashed line), and combining intensity and disparity images (solid line).

corresponding to the just obtained image (see section 2.2). The closer the training points used to define the sensor model to the actual position of the robot, the better the sensor model and, thus, better the update of the robot's position estimation.

Figure 6 shows the average (and the standard deviation) of the error in the sensor model defined as

$$e = \min_{\forall n_j} \|r - n_j\|,$$

with $j \in [1, J]$, r the pose of the robot at the test position and n_j the training points used to define the sensor model (that are different for each test position). An error in the range $[25, 50]$ is quite reasonable taking into account that the distance between training points in X and Y dimensions is 50 cm.

We repeat the test in three cases: (a) using only histogram-normalized intensity images (dotted line on Figure 6), (b) using only disparity maps (dashed line on the figure) and (c) combining normalized images and disparity maps (solid line). In the first and second tests we use $J = 10$ (quite small compared with the total number of training points) and in the third case we use $J = 5$ but for both intensity and disparity (so, we also use 10 training points in the definition of the sensor model).

We can see that the use of disparity maps alone results in a reduction of the error w.r.t. only using normalized intensity images. However, when we combine the two types of information, the results are even better. As mentioned before, disparity maps are eventually affected noise. It is in these noisy cases in which the combination of normalized intensity images and disparity is better. Consequently, we can conclude that the use of features computed from disparity maps combined with those from intensity provide a better sensor model and, thus, it helps to obtain a more robust localization system.

5.3 Global Performance

The last experiment we performed was a navigation exercise to analyze the combined work of the active localization strategy and the use of disparity maps for localization. We instructed the robot to move along a close circuit with 6 passing points (the numbered crosses in Figure 7).

Initially (point A in Figure 7), the robot has no information about its location and it assumes a uniform distribution over its configuration space. This automatically triggers the active vision mechanism described in Section 3. After rotating the head in three different directions, the robot finds out its position with good accuracy and starts to move along the desired path. We considerer that the robot knows its location good enough when the variance of the set of particles is below a given threshold. The trajectory

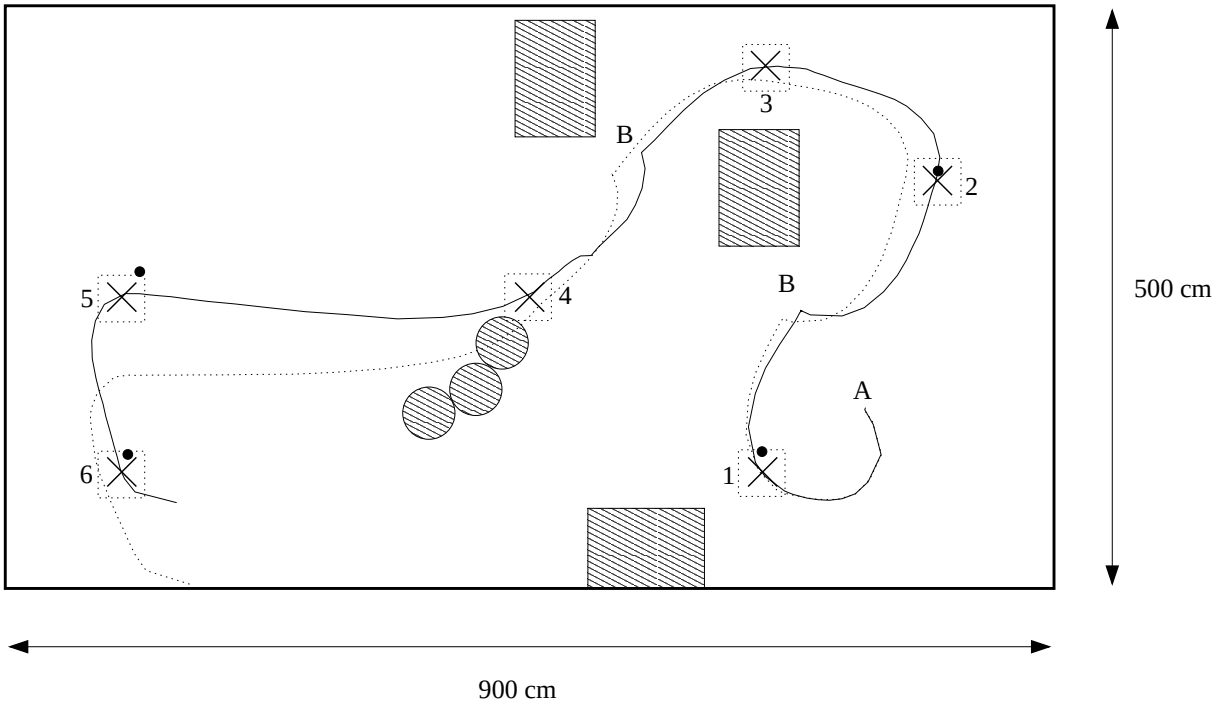

Figure 7: Trajectory of the robot as estimated by our localization system (solid line) and the trajectory according to odometry (dashed line). Shadowed boxes are obstacles. The black dots indicate the *real* position of the robot when our system indicates the robot is exactly on the nearest cross.

estimated by our localization system is shown as a solid line in Figure 7. The black dots close to positions 1, 2, 5, and 6 show the *real* position of the robot when our system indicate that the robot is exactly on the middle of the corresponding crosses. The difference between the correct robot position and the estimated one at these check points is always below 25 cm (the size of the dashed squares around the passing points is 40 cm). The dashed line in Figure 7 is the position according to odometry, initialized using the robot position determined by our localization system at point A. The path determined using odometry clearly diverges from the correct robot position, even in the short trajectory shown in the figure. At points marked as B in Figure 7 the obstacle avoidance mechanisms are triggered and this produces alterations on the path to the next passing points (2 and 4 respectively).

In Figure 8, we can see the effect of a robot kidnap. A kidnap consists in lifting and displacing the robot to a new pose (i.e., a new position or a new orientation or both). By lifting the robot, the odometry information (in which the action model is based) is unvalidated. Only global localization systems are able to overcome a robot kidnap (relative localization systems can not deal with discontinuities in the robot trajectory). In the experiment depicted in Figure 8 (that is a continuation of experiment on Figure 7) we lifted and rotated the robot 90 degrees anti-clockwise at point A. After the kidnap the robot moves toward point B, but the information provided by odometry erroneously indicates that the robot is going toward C. As described in section 2.3, in our localization system, we implemented a simple kidnap detection mechanism that re-samples all particles using the sensor model for the current observation if there is no agreement between the position estimated by the particle filter $p(x_t|u_t, y_t)$ and that provided by the sensor model $p(y_t|x_t)$ for some time steps. In the example, the re-sample is done at point B. After the re-sampling the uncertainty on the robot position is very large and the active localization mechanism is triggered. Again after 3 movements of the robot head, the position is determined with good accuracy

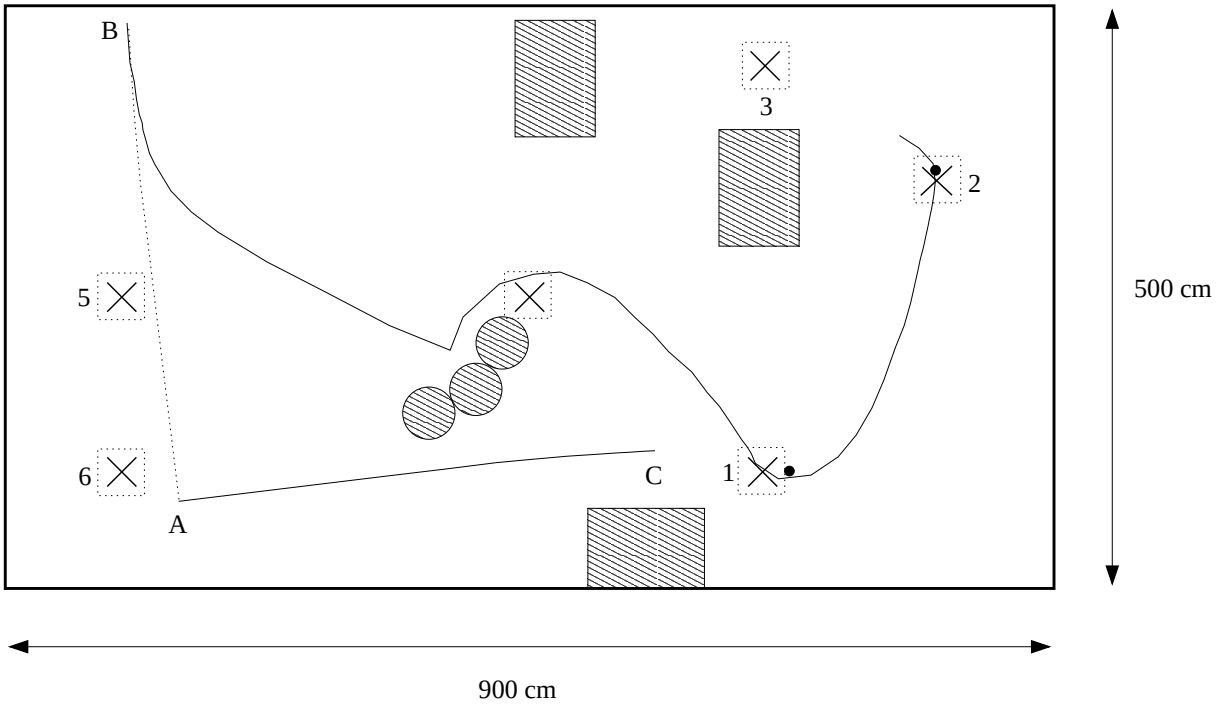

Figure 8: Trajectory of the robot as estimated by our localization system after a kidnap at point A.

and the robot moves to passing point 1 (its objective at the moment the robot is kidnapped). Again, the small black dots close to points 1 and 2 indicate the correct robot position when the localization system indicate the robot in right on the passing points. The fact that the robot passes very close to points 1 and 2 means that the position estimated at point B is correct and that it is properly maintained over time.

Tests including changes of illumination (switching on/off some lights while the robot is moving) show that, thanks to the use of disparity maps, the robot is able to keep a good estimation of its position and, consequently, the navigation task can be achieved without problems. When only intensity images are used, the sensor model is not always correct and the navigation is constantly stopped to perform active localization actions that, in many cases, converge to wrong positions (and, thus, the active localization is triggered again after few moments, and so on).

We let the robot to more over the six passing points for more than 30 iterations observing a stable behavior. Few changes in the environment (displacement/removal of some obstacles, people walking around/in front of the robot, etc.) do not have repercussion on the localization performance as far as the cameras are eventually pointing in a direction where those changes are small. More drastic changes in the environment would prevent our system to converge to the correct robot location.

Videos of the experiments reported here can be downloaded from our web pages (`http://www.science.uva.nl/research/ias`).

Experiments in larger environments (in a 12×25 meters area in the second floor of our building in Amsterdam, in a 12×19 meters area at Philips Research Lab in Eindhoven or at Philips Home Lab that is about 10×13 meters) showed always a similar localization precision and stability. As an example, the localization is accurate enough so that Lino is be able to autonomously dock in a charging station using position as the only feedback. This can only be achieved if the localization error is below ± 25 cm and ± 5 degrees. since the passive mechanism of the re-charging device can not accommodate errors above these thresholds.

6 Conclusions

In this paper, we have introduced two extensions that makes the traditional appearance-based robot localization framework more efficient and robust: active selection of robot actions to speed up the convergence to the correct location and the use of disparity maps for localization to deal with illumination changes. These two extensions are possible thanks to the use of a stereo camera mounted on a pan-and-tilt device.

We have described an active vision strategy that uses an entropy-based action evaluation criterion that can be computed in an efficient way within our localization system. The experiments we report show that this mechanism effectively helps to find out the location of the robot. This is mainly useful in dynamic environments, where our previous passive localization system exhibited some problems. In the examples we described in this paper, we only applied the action evaluation to head rotations in the pan degree of freedom, but the algorithm we have introduced can be applied without modifications to evaluate any possible movement of the robot. Note that the existence of a efficient re-localization mechanism allow us to reduce the number of particles used by the filter, reducing also the execution time per time slice.

Our second contribution is the use of sparse disparity maps to increase the robustness of appearance-based localization to changes in illumination. To compress sparse disparity maps to a reduced set of features, we have to deal with the problem of missing values. We have presented a novel EM-based algorithm to extract the principal components of a set of data that is more efficient in the way in which it deals with missing values than previously existing methods. After the dimensionality reduction, it remains a open problem of how to merge the information provided by disparity maps with the information provided by intensity images. We have proposed to use a linear pool option to combine the models built separately for each type of sensor since the particle filter update automatically takes care of filtering the outliers.

The results we have presented show that disparity maps provide features that are less sensible to changes in the lighting conditions than features obtained with other techniques: histogram equalization and gradient-based filters. These techniques work well when we have changes in the global illumination but they do not deal properly with different distributions of light sources. On the other hand, disparity maps are more consistent over changes in the number and in the position of the light sources. This is because only reliable correspondences are taken into account when defining the disparity map and those reliable matches are likely to be detected in almost all lighting conditions.

We have shown that, using features from disparity maps in addition to those obtained from intensity images, we can improve the quality of the sensor model when illumination conditions are different from those in which the training set is obtained. Thus, disparity maps are a good option to increase the robustness of appearance-based robot localization. However, the good results achieved using disparity maps comes at the cost of using a more complex hardware (we need not only one camera but two calibrated ones) and software (the disparity computation is quite a complex process). However, our experiments show that this cost increase is small enough to perform real-time localization. Additionally, the cost for each localization step for the presented system scales logarithmically with the size of the map and this is the same as the most efficient localization systems existing nowadays [50].

There are a number of issues we have to consider in our future work. The main drawback of appearance-based methods is that localization is only possible in previously mapped areas. The construction of a map is a supervised process that can be quite time-consuming. This problem limits the applicability of appearance-based localization to environments smaller than those tackled by other localization methods [17, 49]. However, there are many possibility to scale the method to larger environments. One is to use hybrid representations that is, for instance to use a geometric representation to obtain a appearance-model by means of simulation [22, 24, 68]. We can also exploit the continuity properties of the appearance manifold [15] to approximate it using just few real samples. Another possibility we already explored is to generate artificial training points from a 3D reconstruction of the environment generated from a reduced set of images [4, 14]. The possibility we are currently working on is to let the robot learn the appearance-based map by itself [58]. We are developing a concurrent mapping and localization (CML) system based in the appearance-based framework and not in the landmark-based one (as it is usually done). Observe

that, the localization methods presented in this paper can be used independently if the map is obtained in a supervised or in a automatic way and thus the work presented here is used without modifications in our new CML system. This new system would allow us to combine he good features of appearance-base localization without having to deal with its limitations.

Acknowledgments

This work has been partially supported by the European (ITEA) project *"Ambience: Context Aware Environments for Ambient Services"*

We would like to thank Bas Terwijn for helping us to perform the experiments reported on this paper. We would also want to express our gratitude to the anonymous reviewers of this paper whose thorough work greatly helped to improve the quality of the paper.

References

[1] T. Arbel and F.P. Ferrie. Entropy-based Gaze Planning. In *Proceedings of the Second IEEE Workshop on Perception for Mobile Agents, in association with the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado*, June 1999.

[2] K. O. Arras, J. A. Castellanos, and R. Siegwart. Feature-Based Multi-Hypothesis Localization and Tracking for Mobile Robots Using Geometric Constraints. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1371–1377, 2002.

[3] A.J.N. van Breemen, K. Crucq, B.J.A. Kröse, M. Nuttin, J.M. Porta, and E. Demeester. A user-interface robot for ambient intelligent environments. In *Proceedings of the 1st International Workshop on Advances in Service Robotics (ASER), Bardolino, March 13-15*, 2003.

[4] R. Bunschoten and B. Kröse. Robust Scene Reconstruction from an Omnidirectional Vision System. *IEEE Transactions on Robotics and Automation*, 19(2):351–357, 2003.

[5] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the Absolute Position of a Mobile Robot using Position Probability Grids. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 896–901, 1996.

[6] W. Burgard, D. Fox, and S. Thrun. Active Mobile Robot Localization. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

[7] Y.H. Chow and R. Chung. VisionBug: A Hexapod Robot Controlled by Stereo Cameras. *Autonomous Robots*, 13:259–276, 2002.

[8] R.T. Clemen. Combining Forecasts: A Review of Annotated Bibliography. *International Journal of Forecasting*, 5:559–583, 1989.

[9] R.T. Clemen and R.L. Winkler. Combining Probability Distributions from Experts in Risk Analysis. *Risk Analysis*, 19(2):187–203, 1999.

[10] D. Cobzas and H. Zhang. Mobile Robot Localization using Planar Patches and a Stereo Panoramic Model. In *Proceedings of the Vision Interface Annual Conference, Ottawa, Canada*, pages 94–99, June 2001.

[11] D. Cobzas, H. Zhang, and M. Jagersand. Image-Based Localization with Depth-Enhanced Image Map. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[12] I.J. Cox and J.J. Leonard. Modeling a Dynamic Environment Using a Multiple Hypothesis Approach. *Artificial Intelligence*, 66(2):311–344, 1994.

[13] J. Crowley, F. Wallner, and B. Schiele. Position Estimation Using Principal Components of Range Data. *Robotics and Autonomous Systems*, 23(4):267–276, 1998.

[14] J. Crowley, F. Wallner, and B. Schiele. Position Estimation Using Principal Components of Range Data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3121–3128, 1998.

[15] J. L. Crowley and F. Pourraz. Continuity Properties of the Appearance Manifold for Mobile Robot Position Estimation. *Image and Vision Computing*, 19:741–752, 2001.

[16] A.J. Davison. *Mobile Robot Navigation Using Active Vision*. PhD thesis, Robotics Research Group, University of Oxford, 1999.

[17] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the CONDENSATION Algorithm for Robust, Vision-based Mobile Robot Localization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1999.

[18] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte-Carlo Localization for Mobile Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1322–1328, 1999.

[19] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood for Incomplete Data via the EM Algorithm. *J. Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[20] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, 1987.

[21] O. Faugeras and Q.-T. Luong. *The Geometry of Multiple Images*. MIT Press, 2001.

[22] D. Fox, W. Burgard, and S. Thrun. Active Markov Localization for Mobile Robots. volume 25, pages 195–207, 1998.

[23] D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[24] D. Fox, W. Burgard, S. Thrun, and A.B. Cremers. Position Estimation for Mobile Robots in Dynamic Environments. In *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[25] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based Navigation and Environmental Representations with an Omnidirectional Camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, 2000.

[26] C. Genest and J.V. Zidek. Combining Probability Distributions: A Critique and an Annotated Bibliography. *Statistical Science*, 1(1):114–148, 1986.

[27] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[28] I. Horswill. *Artificial Intelligence and Mobile Robots*, chapter The Polly System, pages 125–139. MIT Press, Cambridge, MA, 1998.

[29] M. Isard and A. Blake. Condensation - Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[30] P. Jensfelt and S. Kristensen. Active Global Localization for a Mobile Robot Using Multiple Hypothesis Tracking. *IEEE Transactions on Robotics and Automation*, 17(5):748–760, 2001.

[31] M. Jogan and A. Leonardis. Robust Localization using Eigenspace of Spinning-Images. In *Proceedings of the IEEE Workshop on Omnidirectional Vision, Hilton Head Island, South Carolina*, pages 37–44, 2000.

[32] M. Jogan, A. Leonardis, H. Wildenauer, and H. Bischof. Mobile Robot Localization under Varying Illumination. In *Proceedings of the International Conference on Pattern Recognition (ICPR), Quebec, Canada*, pages 741–744, 2002.

[33] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 2002.

[34] L.P. Kaelbling, A.R. Cassandra, and J.A. Kurien. Acting under uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1996.

[35] J. Kleinberg. The localization problem for mobile robots. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 521–531, 1994.

[36] K. Konolige. Small Vision System: Hardware and Implementation. In *Proceedings of the 8th International Symposium on Robotics Research, Japan*, 1997.

[37] S. Kristensen and P. Jensfelt. An Experimental Comparison of Localization Methods, the MHL Session. In *Proceedings of the International Conference on Robotics and Intelligent Systems (IROS), Las Vegas, USA*, pages 992–997, 2003.

[38] B.J.A. Kröse and R.Bunschoten. Probabilistic Localization by Appearance Models and Active Vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2255–2260, 1999.

[39] B.J.A. Kröse, N. Vlassis, and R. Bunschoten. Omnidirectional Vision for Appearance-based Robot Localization. *Lecture Notes in Computer Science*, 2238:39–50, 2002.

[40] B.J.A. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A Probabilistic Model for Appearance-based Robot Localization. *Image and Vision Computing*, 19(6):381–391, April 2001.

[41] C. Kwok, D. Fox, and M. Meila. Adaptive Real-Time Particle Filters for Robot Localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[42] P. Lamon, I. Nourbakhsh, B. Jensen, and R. Siegwart. Deriving and Matching Image Fingerprint Sequences for Mobile Robot Localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea*, pages 1111–1116, 2001.

[43] S. Lenser and M. Veloso. Sensor Resetting Localization for Poorly Modelled Mobile Robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1225–1232, 2000.

[44] J.J. Leonard and H.F. Durrant-Whyte. Mobile Robot Localization by Tracking Geometric Beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[45] J. Little, J. Lu, and D. Murray. Selecting Stable Image Features for Robot Localization Using Stereo. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1072–1077, 1998.

[46] S. Maeda, Y. Kuno, and Y. Shirai. Active Vision Based on Eigenspace Analysis. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1018–1023, 1997.

[47] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-Based Monte-Carlo Localisation without a Map. In *Proceedings Conference of the Italian Association of Artificial Intelligence*, 2003.

[48] T. Minka. Expected-Maximization as Lower Bound Maximization. *http://www.stat.cmu.edu/ ∼minka/ papers/em.html*, 1998.

[49] M. Montemerlo, S. Thrun, D. Koller, and B. Webgreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the 18th AAAI National Conference on Artificial Intelligence*, pages 593–598, 2002.

[50] M. Montemerlo, S. Thrun, D. Koller, and B. Webgreit. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[51] H.P. Moravec. Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*, 9(2):61–74, 1988.

[52] H. Murase, F. Kimura, M. Yoshimura, and Y. Miyake. An improvement of the Auto-Correlation Matrix in the Pattern Matching Method and its Application to Handprinted HIRAGANA Recognition. *Transactions of the IECEJ*, J64-D(3):276–283, 1981.

[53] H. Murase and S.K. Nayar. Illumination Planning for Object Recognition using Parametric Eigenspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 16(12):1219–1227, 1994.

[54] H. Murase and S.K. Nayar. Visual Learning and Recognition of 3-D Objects from Appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[55] D. Murray and J. Little. Using Real-Time Stereo Vision for Mobile Robot Navigation. *Autonomous Robots*, 8(2):161–171, 2000.

[56] C.F. Olson. Probabilistic Self-Localization for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 16(1):55–66, 2000.

[57] M.K. Pitt and N. Shephard. Filtering Via Simulation: Auxiliary Particle Filters. *J. Amer. Statist. Assoc.*, 94(446):590–599, June 1999.

[58] J.M. Porta and B.J.A. Kröse. Appearance-based Concurrent Map Building and Localization. In *Proceedings of the 8th International Conference on Intelligent Autonomous Systems (IAS)*, pages 1022–1029, 2004.

[59] S. Roweis. EM Algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, volume 10, pages 626–632. The MIT Press, 1998.

[60] N. Roy and G. Gordon. Exponential Family PCA fo Belief Compression in POMDPs. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[61] R. Sim and G. Dudek. Comparing Image-based Localization Methods. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico*, 2003.

[62] L. Sirovich and M. Kirby. Low-dimensional Procedure for the Characterization of Human Faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.

[63] I. Sobel. *Machine Vision of Three-Dimensional Scenes*, chapter An Isotropic 3×3 Images Gradient Operator, pages 376–379. Academic Press, 1990.

[64] V.A. Sujan and S. Dubowsky. Visually Built Task Models for Robot Teams in Unstructured Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Washington D.C., USA*, pages 1782–1787, May 2002.

[65] B. Terwijn, J.M. Porta, and B.J.A. Kröse. A Particle Filter to Estimate non-Markovian States. In *Proceedings of the 8th International Conference on Intelligent Autonomous Systems (IAS)*, pages 1062–1069, 2004.

[66] S. Thrun, W. Burgard, and D. Fox. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Machine Learning*, 31:29–53, 1998.

[67] S. Thrun, D. Fox, and W. Burgard. Monte Carlo Localization with Mixture Proposal Distribution. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 859–865, 2000.

[68] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 28(1-2):99–141, 2001.

[69] J.J. Verbeek. Probabilistic PCA with Missing Values. Matlab Source Code. *http://www.science.uva.nl/∼jverbeek/software*, 2002.

[70] N. Vlassis, B. Terwijn, and B.J.A. Kröse. Auxiliary Particle Filter Robot Localization from High-Dimensional Sensor Observations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Washington D.C., USA*, pages 7–12, May 2002.

[71] M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman & Hall, London, 1995.

Appendix I: The EM Algorithm

The EM algorithm [19, 48] aims at determining the maximum likelihood model for a given set of data Z. Of course, not all possible models are considered but only those inside a family of models parameterized by a set of parameters, collectively denoted as θ. The optimal model is determined by maximizing the likelihood or, equivalently, the log-likelihood[1]

$$\Phi(\theta) = \log p(Z;\theta).$$

In many cases, Φ can be defined using an auxiliary set of unobserved or *hidden* variables h. In this case, we have

$$\Phi(\theta) = \log \int_h p(Z,h;\theta).$$

The function Φ might be maximized using a gradient ascent process. However, the EM algorithm provides a simple to implement and step-size free algorithm using iterative lower-bound maximization.

For a given set of parameters θ, the EM algorithm first finds a lower bound Ψ of Φ, possibly such that Ψ touches Φ at point θ. After that, the lower bound Ψ is maximized for the parameters θ. The definition of Ψ for a fixed set of parameters θ is the E step of the algorithm and the maximization of the resulting Ψ is called the M step. The sequential iteration of E and M steps from an arbitrary initial set of parameters θ is guaranteed to find a (local) maximum of Φ.

The lower-bound Ψ used in the E step, can be defined as

$$\Psi(\theta) = \Phi(\theta) - KL(q(h)\|p(h|Z;\theta)) \leq \Phi(\theta),$$

where KL denotes the (non-negative) Kullback-Leibler divergence between two probability distributions. To make $\Psi(\theta) = \Phi(\theta)$ for the current parameters θ, the KL should be zero. The KL divergence is zero iff the two compared probability distributions are equal. Thus, in the E step we set $q(h) = p(h|Z;\theta)$.

In the M step, we have to maximize Ψ. We can rewrite Ψ as

$$\begin{aligned}\Psi(\theta) = \Phi(\theta) - \int q(h)\frac{q(h)}{p(h|Z;\theta)} = -E_{q(h)}\left[\log\frac{q(h)}{p(h|Z;\theta)}\right] + \log p(Z;\theta) = \\ = E_{q(h)}\left[\log\frac{p(Z,h;\theta)}{q(h)}\right] = \int_h q(h)\ \log p(Z,h;\theta) - q(h)\ \log q(h).\end{aligned}$$

The relevant term of Ψ to be maximized is

$$\Psi_M = \int_h q(h)\ \log p(Z,h;\theta),$$

since the other term of Ψ does not depend on θ.

For a more detailed presentation of the EM algorithm see [48], which we followed in the above brief description.

[1] The log is used since it makes the resulting expressions simpler.