Appearance-based Concurrent Map Building and Localization

Josep M. Porta and Ben J.A. Kröse
IAS Group, Informatics Institute, University of Amsterdam
Kruislaan 403, 1098SJ, Amsterdam, The Netherlands
{porta,krose}@science.uva.nl

Abstract. Appearance-based autonomous robot localization has some advantages over landmark-based localization as, for instance, the simplicity of the processes applied to the sensor readings. The main drawback of appearance-based localization is that it requires a map including images taken at known positions in the environment where the robot is expected to move. In this paper, we describe a concurrent map-building and localization (CML) system developed within the appearance-base robot localization paradigm. This allow us to combine the good features of appearance-base localization without having to deal with its inconveniences. The preliminary results included in this paper qualify this approach as a very promising one.

1 Introduction

Robot localization methods can be divided in two families: methods based on landmark extraction and tracking [2, 5, 8, 11], and methods based on an appearance modeling of the environment [6, 7]. Landmark-based localization methods rely on the assumption that the position of the landmarks can be accurately extracted from the raw sensors readings. However, the transformation from sensor readings to geometric information is, in general, complex and prone to errors. As a counterpart, in the appearance-based methods 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 stored in the appearance-based map. This simple way to use the sensor readings makes appearance-based approaches really appealing.

A comparison between the two families of localization methods using vision as sensory input can be found in [14], showing that appearance-based methods are more robust to noise, occlusions and changes in illumination[1] than landmark based-methods. 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 and that is only valid as far as no important modifications of the environment occur. While much work has been done on Concurrent Mapping and Localization (CML) using landmarks [1, 3, 15, 10], it is not the case within the appearance-based approach.

[1] When a edge detector is used to pre-process the images.

In this paper, we introduce a system that is able to perform CML within the appearance-based approach. Our attempt is to combine the advantages of the appearance-based localization with the easy-use and adaptability of the CML approaches.

We first describe more formally the appearance-based localization method. Then, we introduce the modifications necessaries to get a CML system. After that, we show the preliminary results obtained with the new CML system, and we conclude summarizing our work and pointing to directions we need to explore in the near future to complete our system.

2 Appearance-Based Localization

The probabilistic localization method aims at improving the estimation of the pose (position and orientation) of the robot at time t, x_t, taking into account the movements of the robot $\{u_1, \ldots, u_t\}$ and the observations of the environment taken by the robot $\{y_1, \ldots, y_t\}$ up to that time[2]. Formally, 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. Thus 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 robot.

To discretize equation 2 and, thus, to make its computation feasible, we use the auxiliary particle filter [13, 16]. In this approach, 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}. Using the particles, we have that

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

where $\delta(x_{t-1}|x^i_{t-1})$ represents the delta function centered at x^i_{t-1}.

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*. Our sensory input for localization are images taken by the camera mounted on the robot. A problem with images is their high dimensionality, resulting in large storage requirements and high computational demands. To alleviate this problem, Murase and Nayar [12] proposed to compress images (z) to low-dimensional feature vectors (y) using a linear projection

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

The projection matrix W is obtained by principal component analysis (PCA) on the supervised training set ($T = \{(x_i, z_i)|\ i \in [1, N]\}$) including images z_i obtained at known states

[2] In our notation, the Markov process goes through the following sequence $x_0 \xrightarrow{u_1} (x_1, y_1) \xrightarrow{u_2} \ldots \xrightarrow{u_t} (x_t, y_t)$.

x_i. The numerically most stable method to compute this is using Singular Value Decomposition (SVD). This procedure gives us the eigenvectors and the eigenvalues for the initial set of images. After the SVD computation, we use the subset of d eigenvectors corresponding to the largest eigenvalues as rows of the projection matrix W. After the dimensionality reduction, a mixture model can be used to approximate the sensor model. We use the model approximation introduced by Vlassis *et al.* in [16]

$$p(y_t|x_t) = \sum_{j=1}^{J} \lambda(y_t, y_j)\, \phi(x_t|x_j), \tag{4}$$

with x_j the nearest-neighbors (i.e., the subset of training points x_j with a set of features y_j more similar to y_t), $\lambda(y_t, y_j)$ a weight that favors nearest-neighbors closer to y_t, and $\phi(x_t|x_j)$ a Gaussian centered at x_j.

3 Appearance-based CML

In a CML system, the underlying probabilistic model is very similar to the one presented in the previous section. The only difference is that the sensor model, $p(y|x)$, has to be learned on-line.

To define the sensor model we need to identify the areas in the space of poses where each observation y consistently occurs. If this information is available, for a given observation y_t we can evaluate $p(y_t|x)$ on equation 1 for different x's (i.e., for the different particles) just checking to which degree x is in the area where y_t is usually observed.

Due to aliasing, the same observation can occur in many areas in the environment. We propose to represent each one of these areas using a Gaussian. Therefore, the CML map would contain a set of couples (y, X) where y is an observation and X a Gaussian mixture: $X = \{x_k, \Sigma_k, w_k\}$, $k \in [1, M]$, with x_k the center of each Gaussian, Σ_k the covariance matrix and w_k a weight to indicate the confidence on the (y, x_k) association.

To learn the Gaussian mixture for each observation, we propose to move the robot around recording pairs (y, X_p), with X_p a Gaussian mixture representing the estimation of the robot's pose when y was observed (X_p can be computed from the particles). If the observation y is a new one, we create a new point in our map. If y is an already known observation, the Gaussian mixture stored in the corresponding map point, X', is updated using with the Gaussian mixture for the current estimation on the robot's pose, X_p. This update can result in an increase of the uncertainty on the positions where y can be observed (if we have to do the union of X and X_p) or in a reduction of that uncertainty (if we can intersect X and X_p). By performing these union and intersection operations long enough, we would identify all the poses where a given observation consistently occurs.

A wrong estimation on the robot's pose can induce wrong information to be added to the map. Our assumption is that, eventually, the estimation on the robot position would be corrected (using the correct points on the map) and that, from that moment, the map can be also corrected. Additionally, we assume that the initial position of the robot is known (all localization is performed w.r.t. that initial point) and, so, at least we have one correct point in the map.

Due to noise in the sensors, we are not likely to observe exactly the same observation y more than once. For this reason, we say that two observations, y_1, y_2, are the same if their distance $\|y_1 - y_2\|$ is below a given threshold δ_y. Since only observations that are further than

δ_y from previous observations are considered new (and, thus, used to create new points in the map), δ_y determines the granularity of the discretization of the feature space we use to approximate $p(y|x)$. When setting δ_y, we have to trade off tolerance to noise vs. uncertainty in localization: a large δ_y means high tolerance to noise but low location accuracy. In most of the cases, however, we don't need an extremely accurate localization to move the robot in a given environment, provided that the controller includes low level behaviors to deal with local details in the environment such as obstacles, doors, etc.

A important remark is that, since the appearance-map is learned on-line, we can not use PCA to define the projection matrix for the images. What we propose is to use a standard compression technique as, for instance, the discrete cosine transform.

Next, we assume that we have a map in the just outlined format and we describe how to use it to improve the robot's position estimation. After this, we enter in detail in how the map is generated and updated using the estimation on the robot's position.

3.1 Robot's Position Estimation

As mentioned, we represent the robot's position at time t with a set of particles $\{\pi_i, x_i\},\ i \in [1, N]$. After executing action u_t, we get an observation y_t and we want to update the robot's position estimation. To update the particles, the simplest thing we can do is to apply the action model for u_t. This provides a new position estimate with larger uncertainty than the previous one. If there is no observation in the map closer than δ_y to y_t, we can not refine the estimation on the robot's position. So, in the worst case, the robot's pose estimation is computed by the sequential application of the action model from the last moment at which the robot's pose was known.

On the other hand, if there are J observations in the map similar-enough to the current one, we use them to define the likelihood

$$p(y_t|x) = \sum_{j=1}^{J} \lambda(y_t, y_j) \sum_{k=1}^{M_j} w_k^j \, \psi(x|x_k^j, \Sigma_k^j), \tag{5}$$

with ψ a Gaussian centered at x_k^j and with covariance matrix Σ_k^j. This is a generalization of the sensor model on equation 4: the information on the position where observations y_j are obtained is replaced by the uncertain information given by the Gaussian mixture associated with each y_j.

If the weighted sum of the likelihood of the particles, $\sum_{i=1}^{N} \pi_i \; p(y_t|x_i)$, is significant (larger than 10^{-4} in our implementation), then there is an agreement between the current hypothesis on the robot's position and the estimation obtained using the map. In this case, we sample a new set of N particles using $\pi_i \; p(y_t|x_i)$ as a weighting factor (this is the sampling process used in the Auxiliary Particle Filter [13]). The result of this sampling is a concentration of the particles in the prior-likelihood intersection and, thus, a reduction in the uncertainty on the robot's pose estimation.

If the weighted sum of the likelihood of the particles is not large enough, the prior and the likelihood provide alternative hypotheses on the robot's position. This can be produced by many causes such as errors in the map, noise in the sensor readings, errors in the current estimation on the robot's pose, or due to a kidnap. There is no way to differentiate a priori in which of these situations we are. Therefore, what we have to do is to track all the hypotheses

and to wait for subsequent sensor readings to determine which one of them is the correct one and, thus, in which one of the situations we were. To preserve the prior and to add the new hypotheses suggested by the likelihood, we keep a percentage (80% in our implementation) of the particles selecting them at random and we sample the rest of particles from the sensor model (equation 5). This sampling method is closely related with the heuristic presented in [9] to deal with the kidnap problem, which is a particular case of mismatch between the prior and the likelihood.

3.2 Map Update

There are two operation to be performed on the map: adding new points to it and updating the information of previously added points.

We add a new point if there is no point in the map with an observation closer than δ_y to the current observation, y_t. The point added to the map includes the current observation and a set of Gaussian functions resulting from clustering the particles representing the current position of the robot. Particles that are close to each other are represented by the same Gaussian. The center of the Gaussian is the average of the particles and the covariance matrix is computed from the dispersion of these particles. Finally, the weight assigned to the Gaussian is the addition of the weights of the particles it represents. To avoid too weak hypotheses on the robot's position, Gaussians with low weight (less than 0.01) are not considered.

If the map includes observations similar to the current one, we use the current estimation on the robot's position to update the corresponding map points. We cluster the particles to get a Gaussian mixture as explained before and we fuse the resulting Gaussian mixture with the Gaussian mixtures stored in training points to be updated. The fusion consist in adding the new Gaussian mixture to the Gaussian mixture in the training points, merging those Gaussians that are close enough (i.e., with a small squared Mahalanobis distance between centers). The merge operation is a weighted average for the mean and the covariance matrices and an addition of the weights of those Gaussians. After the fusion of the Gaussian mixtures, we re-normalize the weights. The result is that overlapping Gaussians are reinforced since their (relative) weight is increased. If we know for sure that our action model is correct (i.e., no kidnap situations are possible), we can back-propagate the improvements in a given map point to previous points in the map along the path of the robot using, for instance, the method described in [4].

4 Experiments and Results

We tested the proposed CML system in two experiments. In both of them we perform concurrent map building and localization but in the first one we mainly check for the performance of the localization and the second one is more demanding as far as map building is concerned. In the two experiments the robot is taking images every 1 second and applying the CML algorithm described above.

In the first experiment, we manually drive the robot along a closed circuit four times (see figure 1-left). Since the robot moves in an non-explored area, the first loop is used to initialize the map. In the other three iterations, an error in odometry is induced by slightly lifting and rotating the robot clockwise when it is at position $(0, 0.75)$. This makes the information provided by odometry invalid and the position of the robot must be estimated by exploiting

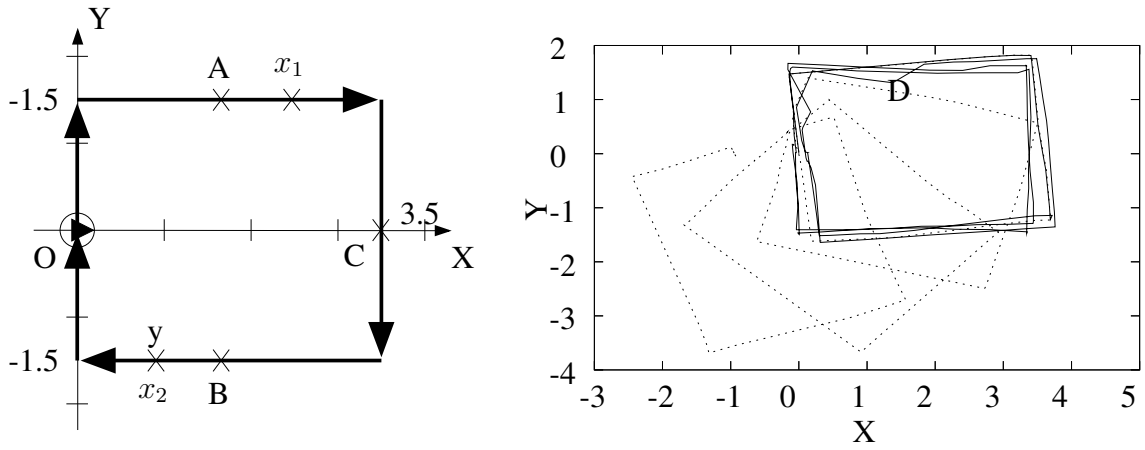

Figure 1: Left, the path of the robot used for the tests. The environment is a normal office. The initial position of the robot is at the origin O aligned to the X positive axis. Units are in meters. Right, the path followed by the robot according to odometry (dotted line) and the estimated position using our CML system (solid line).

the map built in the first loop. Figure 1-right shows the the path of the robot according to odometry (dotted line) and according to our localization method (solid line)[3]. At point D, we can see the initial divergence between the position according to odometry and the estimated one: the CML system first follows odometry but after a few seconds the matching between the observations obtained by the robot and those in the map allows for a correction in the position estimation. In the last three loops no new points are added to the map, but the existing ones are updated. This update allows the CML system to provide a good estimation of the position, even getting better in the last loops, while the position according to odometry is completely wrong.

The second experiments is more demanding as far as map construction is concerned. In this experiment, we repeat the same path as in the first one (see figure 1-left), but in the first loop, at position A the robot is kidnapped: lifted and displaced to position B and rotated $180°$. Thus, according to odometry the robot moves from A to C while it is actually is moving from B to O. The kidnap introduces a large error while the map is in its initial construction phase (observations initially assigned to points in the path from A to C are actually occurring in the path from B to O). When the robot gets to O the error is detected and the correct position is recovered. In the following iterations over the loop, the CML system corrects the map and it finally manage to provide correct localization. We can use an example to show how the map correction works. Initially, the incorrect association $(y, \{(x_1, \Sigma_1, w_1 = 1.0)\})$ is stored in the map (see figure 1). In the second loop, this map point is updated to $(y, \{(x_1, \Sigma_1, w_1 = 0.5), (x_2, \Sigma_2, w_2 = 0.5)\})$. In the following iterations, the weight of the association (y, x_1) decreases while the correct association (y, x_2) is reinforced. After some loops, the association (y, x_1) becomes weak and it is eventually removed from the map. Observations initially wrongly associated with positions close to A are corrected first and the correction is propagated toward observations initially wrongly associated with points close to C, but that are actually close to O.

Figure 2 shows the error in localization as the robot moves around the circuit. Large errors are due to the wrong points in the map caused by the initial kidnap. We can see that the system

[3]In the first loop both lines are mostly overlapped, so the dotted one can be hardly appreciated.

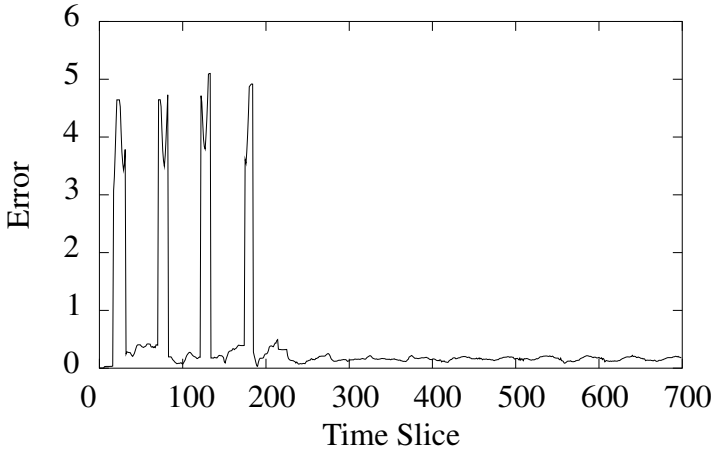

Figure 2: Error in localization when we kidnap the robot in the initial steps of the map construction.

is able to compensate for these wrong points. Every loop around the circuit takes 100 time slices (except the first one that is shorter due to the kidnap).

5 Conclusions

We have introduced a system that is able to simultaneously build an appearance-map of the environment and to use this map, still under construction, to improve the localization of the robot. The experiments reported on this paper show that this approach is able to provided good robot localization even when severe errors are introduced in the map.

The use of a particle-based representation for the position of the robot allows us to deal with the non-linearities of the action model. On the other hand, the representation of a map using highly compressed images and Gaussian mixtures (derived form the clustering of particles) allows us to represent the environment in a very compact way.

The on-line construction and update of the map allows us to overcome the major hurdles of traditional appearance-based localization. First, the robot can operate in previously unknown areas. Second, we can deal with changes in the environment: new observations obtained at already explored positions are added to the map and the old observations at those position are not used any more and they are slowly forgotten. Finally, the way in which the map is built guarantees a uniform sampling of the feature space and not of the geometric space, as it happens in normal appearance-based localization. Sampling uniformly the feature space is essential for achieving a good localization since the sensor model is based on the similarities (i.e., the distances) in that space.

When we start our system from scratch, we need to know the initial position of the robot but, as the map is built, our system is able to perform global localization.

The accuracy in localization that can be achieved by the presented CML system for a given position X is lower bounded by the error in odometry for a direct displacement from the origin O to X. Thus, the error in odometry limits the area that we can map with a given accuracy. With our Nomad Scout robot we can map an area of about 20×20 meters with a positioning error lower than 1 meter (an error below 5%). To enlarge this area, we need to use additional motion sensors (visual odometry, accelerometers, etc) to complement the information of the odometry.

The results presented in this paper qualify the proposed approach as a very promising

one, but much work has to be done to complete the system. The two main points we want to address in the near future are a better formalization of our CML system, and the integration of the navigation in the robot controller since, as mentioned, up to now the robot is manually operated using a joystick.

Acknowledgments

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

References

[1] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

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

[3] H.J.S. Feder, J.J. Leonard, and C.M. Smith. Adaptive Mobile Robot Navigation and Mapping. *International Journal of Robotics Research, Special Issue on Field and Service Robotics*, 18(7):650–668, 1999.

[4] S.H.G. ten Hagen and B.J.A. Kröse. Trajectory reconstruction for self-localization and map building. In *Proceedings of the IEEE International Conference on Robotics and Automation, Washington D.C., USA*, pages 1796–1801, 2002.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

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

[12] 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.

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

[14] 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.

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

[16] 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.