Visual Tracking and Localization of a Small Domestic Robot

Raymond Sheh and Geoff West

Department of Computing, Curtin University of Technology,
Perth, WA, 6102, Australia
{shehrk, geoff}@cs.curtin.edu.au

Abstract. We investigate the application of a Monte Carlo localization filter to the problem of combining local and global observations of a small, off-the-shelf quadruped domestic robot, in a simulated *Smart House* environment, for the purpose of robust tracking and localization. A Sony Aibo ERS-210A robot forms part of this project, with the ultimate aim of providing additional monitoring, human-system interaction and companionship to the occupants.

1 Introduction

This paper investigates using these two forms of sensors for the purpose of localizing a small quadruped robot in the presence of obstacles in a *Smart House*, as an aid to elderly and disabled people. Such a robot may be guided by the *Smart House* system, giving it the ability to interact with the occupants and visit areas not otherwise covered by *Smart House* cameras. For example, the robot could be used to disambiguate a person dropping out of view to retrieve an object from a person falling over out of view of the *Smart House* cameras. A combination of local (robot-based) and global sensing may be used to localize a robot in an environment. In domestic environments, low-cost and minimal interference with the existing environment are desirable, ruling out laser rangefinders, sonar rings and omnidirectional cameras. However, cheap surveillance cameras are available that can be mounted on the robot or fixed to the house.

This investigation makes use of the Sony Aibo ERS-210A Entertainment Robot, shown in figure 1(a). This robot possesses a low quality 176×144 pixel camera, a 385MHz MIPS processor, wireless ethernet, quadruped omnidirectional locomotion and approximately one hour of autonomous powered operation per battery charge, making it well suited for the *Smart House* project. However, its undulatory walking motion leads to poor odometry.

The system infrastructure used to operate and interface with the Aibo is derived from the code released by the UNSW/NICTA rUNSWift 2003 RoboCup

D. Nardi et al. (Eds.): RoboCup 2004, LNAI 3276, pp. 410–417, 2005.

team [1] [1]. This code provides a framework for controlling the robot's movements and acquiring and processing sensory information.

This paper describes how a partial model of the environment and information from robot odometry, the onboard camera and ceiling cameras can be combined to track and guide the robot through the environment. Preprocessed images from the robot camera are transmitted to the host PC and used to locate the robot relative to known carpet patterns. The ceiling cameras are also used to track potential robot positions. Both sensing techniques are complementary. Ceiling cameras often suffer from occlusion and clutter whilst information from local vision is often aliased or noisy. The process of sensor fusion makes use of Monte Carlo localization (MCL) to deal with incomplete or ambiguous data from one or both sources gracefully.

2 Environment

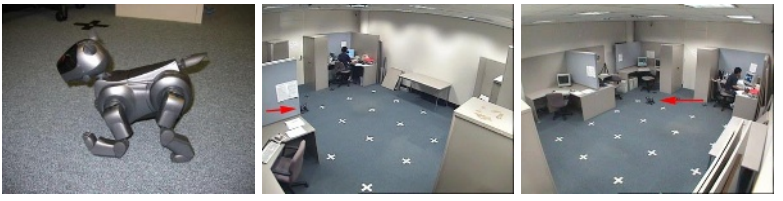
Four surveillance quality color overhead cameras are placed in the corners of the *Smart House* environment such that their fields of view overlap in the unobstructed center area as in figure 1(b,c). These cameras feed four PCs for analysis. The location of the robot on the floor is determined using a motion-based background subtraction technique [4] followed by a camera specific image to real world coordinate mapping[2]. The accuracy of the image to floor mapping is approximately 10cm at worst.

All of the joints in the legs and neck of the robot are equipped with angle encoders with sub-degree accuracy. However, mechanism slop and flexing of the robot structure reduce the accuracy of camera positioning relative to the ground plane to about 1° in angle and 5mm in height. These errors as well as the inability of the robot to determine which paws are in contact with the ground at any one time, means the accuracy drops further as the robot walks. The angle encoder values allow odometry to be obtained from the walk engine. Unfortunately, asymmetry in the robot's weight distribution, and variable slip in the paws, introduce significant errors into this odometry.

The rUNSWift software can color segment the images from the robot's onboard camera in realtime using a static 3D color lookup table. The resulting images, called *CPlanes*, can then be fed to the PC host via the robot's wireless ethernet interface at close to realtime speeds for further processing. Additionally, code exists to use the angle encoder values and known height to locate points observed with the camera in 3D space relative to the robot. However, positioning inaccuracies as discussed previously, combined with significant barrel and chromatic distortion severely limit the accuracy of this technique.

[1] The first author was a member of the 2003 rUNSWift RoboCup team. Components of the work described in this paper extend work undertaken by the first author and other members of the 2003 rUNSWift RoboCup team.

[2] The crosses in figure 1 are normally used to calibrate the overhead cameras.



(a) (b) (c)

Fig. 1. The Sony ERS-210A entertainment robot (a) and views from two of the *Smart House* overhead cameras (b),(c). Note the large overlapping region and the white crosses marked on the carpet. Arrows indicate the position of the robot

3 Monte Carlo Localization

The traditional approach to tracking is to use Kalman filtering that assumes that sensor measurements have Gaussian distributions. However, at any point in time the robot can only be represented in one location (with some variance). The MCL filter [2] allows multiple robot locations to be tracked because it does not commit to one location and instead stores many candidate locations.

MCL utilizes modified grid-based Markov localization [2] to represent the robot's possible location (with heading) or state $l = [x, y, \theta]$ as a probability map in a discretized state space. This representation is modified using a motion model and observation updates in order to track the robot's location over time. At any time, analysis of the distribution of locations allows the most probable location to be determined. The process is Markovian because the current distribution of possible locations is solely based on the distribution at the previous time step, the current observations and the robot's action. For efficiency, the grid is probabilistically sampled, each sample called a particle within a set of particles $S = \mathbf{s_i}|i = 1, ..., N$. Each particle $\mathbf{s_i}$ is a possible location l_i of the robot with an associated probability or weighting p_i.

Initially the distribution of particles can be random over the whole space of possible values of l but will soon converge to form a tight distribution after a number of iterations. In addition to S, two other parameters are required. First, a probability distribution $p(l|l', a)$: given the action a what is the probability that the robot goes to l from l'. Second, a probability distribution $p(s|l)$: given the location l what is the probability of a sensor derived location occurring. At each iteration, the steps in the algorithm are:

1. Generate N new samples probabilistically sampled from S. For each new sample, the location is called l'_i.
2. Generate l_i from $p(l|l', a)$ where a is the action or command given to the robot. Set the probability p for each l_i to be $1/N$.
3. Generate a new value of p_i for each new particle from $p(s|l)$ i.e. $p(l|s) \leftarrow \alpha P(s|l)$ where α is used to ensure $\Sigma_{i=1}^{N} p_i = 1$.
4. If required, determine the best estimate of location from the distribution S.

The main issues are defining the distributions $p(s|l)$ (based on observations) and $p(l|l', a)$ (based on odometry), and the number of particles N. In addition, in this work, the algorithm requires some approximations because of speed considerations.

4 Sensing

Preprocessing of data from the individual overhead cameras involves four major steps. Firstly, segmentation of the candidate robot pixels is performed by background subtraction using a mixture of Gaussians background model for each pixel and color channel [7]. Foreground objects are extracted as long as they move and are regarded as candidate robot pixels. Secondly, these pixels are projected onto the ground plane via a camera-specific transformation to form candidate robot points (assuming the robot is on the ground) [4]. Note that after projection, the points do not necessarily form a single connected grouping but will form clusters of points (figure 2).

Thirdly, these candidate points are filtered based on geometric properties to eliminate those that are unlikely to be part of the robot region. The filter applied to these points uses the kernel matrix of figure 2, tuned to give a high response when convolved with a cluster of points of a size that approximately matches the size of the robot. Finally the output is thresholded to segment out the candidate robot regions. The varying density of the candidate robot clusters is reduced by using morphological dilation [5] to directly yield a measure of $p(l|s)$.

A variant of the *NightOwl* system [6] developed by rUNSWift is used to extract localization information from the onboard camera. In order to determine a measure of $p(l|s)$ this algorithm matches known patterns on the ground and consists of three steps.

Firstly, patterns of interest on the gray-blue carpet are detected using a noise-reducing edge detector applied to the color segmented *CPlane* [6]. These patterns consist of white crosses on a 1m grid as in figure 1. These points are then projected onto the ground plane. Points projected further than 75cm from the robot are discarded due to potential inaccuracies. This process is illustrated in figure 3.

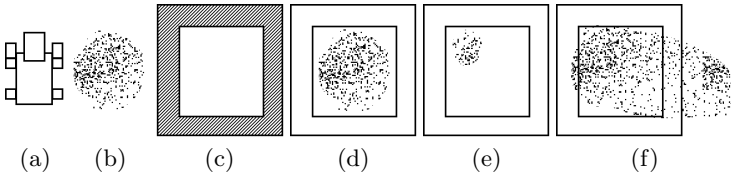
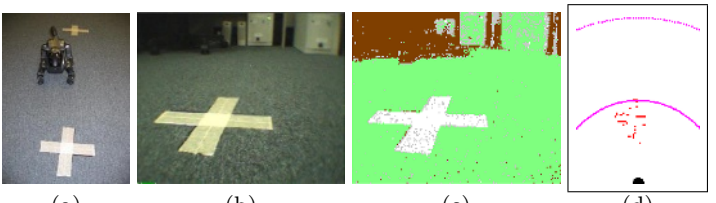
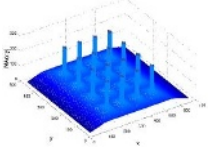


Fig. 2. The robot (a) from overhead appears as (b). The kernel (c) has a similarly sized positive inner region and negative outer region of a width equal to the minimum desired distance between detected robot clusters and other points. Three situations arise: the kernel matches a robot-sized cluster (d), a cluster that is too small (e) and too large (f)



(a) (b) (c) (d)

Fig. 3. Robot at (a) observing a feature through its camera (b), transmitting a color segmented *CPlane* to the host PC (c) and projecting onto the ground plane in robot-relative space (d). Arcs in (d) illustrate points 50cm and 100cm from the robot respectively. Note that the robot is in the same position as in figure 1



$$MA_{MAX} = 252$$
$$M(d) = \begin{cases} MA_{MAX} - d & , d < 3 \\ \frac{1}{2}\left(MA_{MAX} - d\right) & , d \geq 3 \end{cases} \quad (1)$$

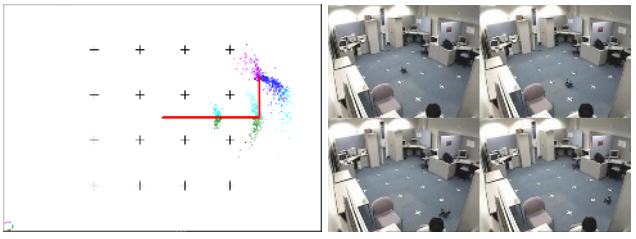
Fig. 4. Matching array MA in heightfield and equation formats. $M(d)$ is the value of an element of MA based on its distance d from the nearest matchable feature

The matching of these observed, projected points with the known environment model is performed on an as-needed basis given a query state l for which an estimation of $p(l|s)$ is needed. Each edge point is transformed by this state to real-world co-ordinates. These points are multiplied with a matching array MA (figure 4) which serves as a precomputed model of detectable environmental features. Locations in the array representing detectable features (crosses) have a high value and the remaining locations are assigned successively lower values based on their Manhattan distance from the nearest feature, through a non-linear mapping (equation 1) to reduce the effect of outliers on the matching.

5 Implementation

The PC receives odometry information from the robot that is used to update particle positions. These updates have most of their bias removed but retain significant errors. To account for these errors in a probabilistic manner, the position updates are randomly scaled to between 90% and 100% in both position and heading. This motion model may be demonstrated by ignoring sensors and starting from a known state (figure 5). As the robot moves, the density of particles S disperses to account for odometric errors. In this application, 3000 particles were used to balance accuracy and ease of tracking against computational load.

The intermediate probability measure of each particle p_i is updated based on the observations from the cameras. The combination of each observational



(a) (b)

Fig. 5. The MCL filter tracking a non-sensing (dead reckoning only) robot (a), initialized to the robot's actual state. (b) shows successive actual robot positions

probability $p(l|s_i)$ from each source i to form a consolidated observational probability measure $p(l|s)$ is performed via additive heuristic rules formulated to reduce the bias towards areas of the environment that may be observed by many cameras. The result is summed with a probability measure $p(l|s_o)$ obtained from the *NightOwl* subsystem evaluated at l. These summations are possible since the measurements may be regarded as independent. Note that the overhead cameras do not contribute heading information θ.

To extract a single most likely robot location, an iterative algorithm based on Expectation-Maximization [3] is used. The average Euclidian distance in (x, y) is found between the last iteration's average position and each particle. A weight of 9 is assigned to particles within this distance and 1 to all other particles to reduce the effect of outliers. A weighted average position of all particles is then computed and the process repeated. This is efficient since the algorithm tends to converge within four iterations. The vector average of headings for particles close to this average position is then used to determine the most likely robot heading.

6 Experimental Results

When used alone, the data from the overhead cameras localized the robot even in the presence of some ambiguity, as long as the robot moved. For example, in figure 6 the robot has stopped long enough to become part of the background. After it begins to move again, the background "hole" left by the robot temporarily appears as another candidate robot (figure 6(b)). The filter continues to track the correct state once the hole becomes incorporated into the background.

When the robot is initially detected, the cluster of particles around the robot's state have uniformly distributed headings. As the robot moves, only those with headings matching the robot's heading will receive motion updates that cause their future predictions to coincide with subsequent observations. This technique is able to reliably achieve an angular accuracy of around 5° and spatial accuracy of around 20cm after only 1m of motion using only one overhead camera. When

Fig. 6. Example of the particle filter coping with an ambiguous observation. Sequence (a) to (c) displays the input to the particle filter from the overhead camera, sequence (d) to (f) displays the corresponding state of the particles in the particle filter

there is significant clutter and noise, multiple cameras improve accuracy and speed of convergence.

The onboard camera is not able to globally localize the robot due to ambiguity i.e. observing a single cross provides four potential locations and with twelve crosses, 48 potential locations exist. Naturally, this ambiguity is reduced for more unique, natural and less repeating carpet patterns and features. Despite this, it is still possible to, over time, extract global localization from this data by tracking each of these potential locations. As the robot moves, some of these points may be eliminated as they coincide with environmental limits.

The combination of onboard localization information with that provided by the overhead cameras greatly improved the speed and accuracy with which the robot's heading was determined. If a carpet feature was visible whilst the robot's heading was unknown, the filter would often snap to two or three potential headings (due to similar features being visible in several directions). A small amount of movement, often only 20cm, would then eliminate all but one of these headings.

In the following example, only one overhead camera was used. The filter was able to track the robot (figure 7(a)) up to when it moved behind the barrier. The particles began to spread out as the robot was no longer visible (figure 7(b)). In a second test, the onboard camera observed the cross in front of it and was able to maintain localization and thus the cluster was tight even when the robot was clearly still occluded (figure 7(d)).

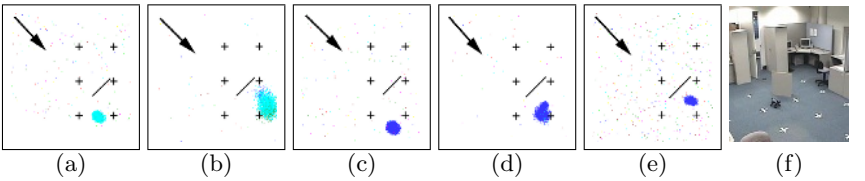


Fig. 7. Example of the robot moving behind an occlusion. In (a) and (b), no crosses are visible to the robot and the particles diverge. In (c) to (e), the robot continues to view crosses and maintain localization. Overhead camera vision (from the opposite corner of the room to the experiment camera) when the robot was occluded appears in (f)

7 Conclusion

This paper has demonstrated that the Monte Carlo filter can successfully track a robot using both local and global sensing and to cope when one of these is absent. It can accommodate ambiguous as well as erroneous observations. A number of heuristics have been used to speed up the algorithm such that real time performance is achieved. The *NightOwl* algorithm has been demonstrated to usefully contribute localization information in a real-world environment and proved to be very effective as a sensor despite the highly ambiguous information it provides because of the repeating pattern of crosses. Whilst *NightOwl* alone may not be effective for global localization in this specific application it allows the system to deal with inaccurate or missing global information.

Acknowledgements

We would like to thank the 2003 UNSW/NICTA rUNSWift RoboCup team, whose code infrastructure formed the basis for the robot and robot-PC communications code, and in particular Bernhard Hengst who initially proposed the *NightOwl* algorithm. We would also like to thank Sebastian Lühr and Patrick Peursum whose code infrastructure formed the basis for the overhead camera and PC-PC communications code. Finally we would like to thank Svetha Venkatesh for her useful advice and input through the course of this project.

References

1. Jin Chen, Eric Chung, Ross Edwards, Eileen Mak, Raymond Sheh, Nicodemus Sutanto, Terry Tam, Alex Tang, and Nathan Wong. rUNSWift, UNSW RoboCup2003 Sony Legged League Team. Honours thesis, School of Computer Science and Engineering, The University of New South Wales, 2003.
2. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots.
3. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.
4. Patrick Peursum, Svetha Venkatesh, Geoff A.W. West, and Hung H. Bui. Object labelling from human action recognition. In *IEEE Conference on Pervasive Computing and Communications*, pages 399–406, March 2003.
5. Robjert J. Schalkoff. *Digital Image Processing and Computer Vision.* John Wiley and Sons, 1989.
6. Raymond Sheh and Bernhard Hengst. Nightowl: Self-localisation by matching edges. Technical Report UNSW-CSE-TR-0406, School of Computer Science and Engineering, University of New South Wales, 2004.
7. C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.