Online Control Policy Optimization for Minimizing Map Uncertainty during Exploration

Robert Sim
Department of Computer Science
University of Toronto
Toronto, ON Canada
Email: simra@cs.toronto.edu

Gregory Dudek
Department of Computer Science
McGill University
Montreal, PQ Canada
Email: dudek@cim.mcgill.ca

Nicholas Roy
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA USA
Email: nickroy@mit.edu

***Abstract*— Tremendous progress has been made recently in simultaneous localization and mapping of unknown environments. Using sensor and odometry data from an exploring mobile robot, it has become much easier to build high-quality globally consistent maps of many large, real-world environments. However, relatively little attention has been paid to date to the controllers used to build these maps. Exploration strategies are often used to cover the largest amount of unknown space as quickly as possible, but few strategies exist for building the most reliable map possible. Nevertheless, the particular control strategy can have a substantial impact on the quality of the resulting map.**

In this paper, we devise a control algorithm for exploring unknown space that explicitly tries to build as large a map as possible while maintaining as accurate a map as possible. We make use of a parameterized class of spiral trajectory policies, choosing a new parameter setting at every time step to maximise the expected reward of the policy. We do this in the context of building a visual map of an unknown environment, and show that our strategy leads to a higher accuracy map faster than other candidate controllers, including any single choice in our policy class.

I. Introduction

Simultaneous mapping and localization (SLAM) is one of the core problems of mobile robotics. A navigating robot requires an accurate and globally consistent model of the world in order to make decisions about how to get from point to point within the environment. As a result, substantial effort has been spent in having robots learn environment models, or maps, automatically. It is now possible to build high-quality maps of a wide variety of environments with a number of different sensors. Typically, the mapping process involves manually (or heuristically) controlling a robot around the environment while it acquires sensor data throughout the space, or possibly having the robot explore the space automatically. The recorded sensor data is assembled into the map either during the exploration process, or more commonly, after the fact.

Recording the sensor data for a good map, however, is not necessarily a straight-forward process. The control strategy used to acquire the data can have a substantial impact on the quality of the resulting map; different kinds of motions can lead to greater or smaller errors in the mapping process. Ideally, we would like to automate the exploration process, not just to drive the robot to explore unknown areas of the space, but to do so in a manner that will lead to high accuracy maps.

In this paper, we focus on the problem of finding such a motion controller, one that can explore quickly while gathering data in a manner that will lead to the most accurate map. This is a different problem to standard motion planning, in that we do not have an *a priori* model of the environment that we can use to find an optimal trajectory. The entire point of the trajectory is to build such a model, but without the model we cannot precompute a plan that will build the most accurate map. We will instead find an approximate solution, using a greedy strategy for generating plans that will give a reliable map in expectation over short sections of trajectories. While this algorithm allows us to estimate the optimal trajectory for maximum-coverage SLAM, the result is more important for the new paradigm it exemplifies. We show how to parametrically optimize SLAM to produce a general purpose solution with an efficiency that could not readily be achieved manually.

We will focus our attention on a parameterised policy class, building on our recent work in which a set of hand-crafted exploratory policies were examined for their accuracy, coverage and efficiency [1]. This policy class allows us to vary the strategy from very explorative to very conservative (e.g., returning to known space regularly to re-localize). The primary results from the previous work indicated that the most accurate exploratory policy was also the most inefficient.

We will use a vision-based representation of the environment for navigation [2]. A visual map is constructed by tracking salient image features over a set of training images and then computing generative models of the features as functions of the robot's pose. Even when the map is to be produced as a side-effect of the trajectory executed to accomplish some other task, it may be possible to alter the trajectory to improve the quality of the resulting map.

II. SLAM and Visual Navigation

Our visual map representation employs a landmark learning framework developed previously. We review it here in brief and refer the reader to this work [2] for further details. The object of visual mapping is to learn a generative model of the image-domain features of an environment. We can use this model to predict the maximum-likelihood observations from arbitrary camera poses, and then use a standard Bayes' filter

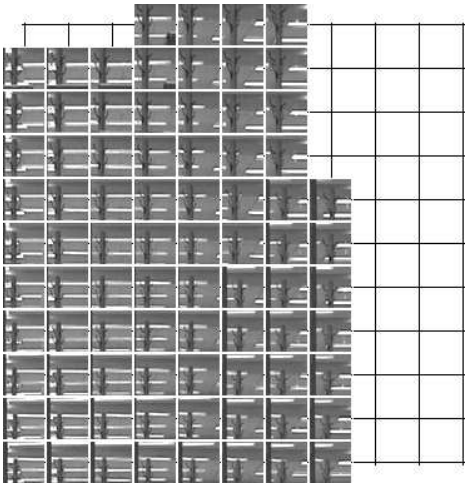


Fig. 1. A set of observations of an extracted scene feature. The grid represents an overhead view of the pose space of the camera, and feature observations are placed at the pose corresponding to where they were observed.

to accurately track the position of the robot during navigation. We make use of the Kalman filter (KF) such as described in [3], [4]. We first describe the process of building a visual map assuming an idealised, perfect ground-truth position data, and then describe how the algorithm uses a Kalman filter in conjunction with imperfect (but real) odometry to build the visual map in practice. In practice, the localization framework requires using an extended Kalman filter (EKF) by linearizing the prediction and measurement models. This is a standard practice in state estimation, and the reader is referred to [3], [4] for the details.

The framework operates as follows: assume for the moment that the robot has collected an set of observations or images of a scene, where each image is associated with a known, perfect ground-truth position. This set of images from known poses constitutes the visual map. A corner detector is applied to the set of images to select an initial set of candidate features [5]. The selected candidate features are then tracked across the ensemble of images by maximizing the correlation of the local intensity image of the feature. Figure 1 depicts the result of tracking one feature across an image ensemble, wherein the local feature intensity image is depicted at the pose from which it was observed.

Although we record each feature at specific, known poses in the environment, we can generate (or predict) new "observations" of the feature from arbitrary poses by interpolating between the recorded observations. The observation of a feature is represented as the position of the feature in the image, $\mathbf{z} = [x\ y]^T$, and an interpolator is constructed using simple bilinear interpolation between neighbouring observations. In practice, an arbitrary interpolation scheme can be employed and in this work we employ a triangulation-based approach for reasons of efficiency. In order to evaluate the features and guard against outliers, the resulting models are validated using leave-one-out cross-validation. Once the visual map and interpolators are constructed, we localize the robot during navigation using a Kalman filter.

The Kalman Filter and Visual Navigation

The standard Kalman filter provides a probabilistic state estimate as a Gaussian distribution, consisting of a mean state estimate $\hat{x}$ and an error estimate, or covariance C. The KF operates in two stages: a prediction step, based on the forward motion of the robot, and a measurement step, based on the observation or image. The conventional KF (and EKF) for localization typically assumes a geometric representation of the environment. In contrast, the visual map representation is using feature observations in the image domain. The advantage to this representation is that the only parameters maintained in our implementation are those of the robot pose, unlike EKF implementations which encode both robot pose and landmark position parameters. The disadvantage is that we can no longer use the observations of each feature to model the features probabilistically. We compensate for this by using cross-validation to estimate feature covariances.

The robot's current state estimate is given by $\mathbf{x}(k-1)$. At time step k, the robot executes an action $\mathbf{u}(k)$ and takes a subsequent observation $\mathbf{z}$. The filter is updated from the command $\mathbf{u}$ according to the standard EKF prediction model:

$$\mathbf{x}^{\ominus}(k) = A\mathbf{x}^{\ominus}(k-1) + B\mathbf{u}(k) \quad (1)$$

$$C^{\ominus}(k) = AC(k-1)A^T + Q, \quad (2)$$

where $\mathbf{x}^{\ominus}(k)$ is the prediction state estimate, $C^{\ominus}(k)$ is the prediction covariance or error, A and B describe how the state changes as a function of the previous state and the control respectively, and Q is the error resulting from the motion.

An image is acquired and those features $\mathbf{z}_i$ in the image that match features in the model are extracted. At the current pose estimate, a predicted observation $\hat{\mathbf{z}}_i$ for is generated each feature. The state and error estimates are updated as

$$\mathbf{x}(k) = \mathbf{x}^{\ominus}(k) + \mathbf{S}_i(k+1)\mathbf{v}_i(k) \quad (3)$$

$$C(k) = (I - \mathbf{S}_i(k))C^{\ominus}(k). \quad (4)$$

where $\mathbf{v}_i(k)$ is the innovation, the extent to which the actual image features differ from the expected image features, and $\mathbf{S}_i(k)$ is the Jacobian of the predicted observation.

The innovation $\mathbf{v}_i(k)$ is computed as

$$\mathbf{v}_i(k) = \mathbf{z}_i(k) - \hat{\mathbf{z}}_i(k). \quad (5)$$

The innovation covariance requires estimation of the Jacobian of the predicted observation given the map and the prediction estimate. We approximate this Jacobian as the gradient of the nearest face of the model triangulation and define it as $\nabla\mathbf{h}_i$. The innovation covariance then follows the standard observation model:

$$\mathbf{S}_i(k) = \nabla\mathbf{h}_i\mathbf{P}(k|k-1)\nabla\mathbf{h}_i^T + \mathbf{R}_i(k) \quad (6)$$

where $\mathbf{P}$ is the pose covariance following the action $\mathbf{u}$, and $\mathbf{R}$ is the cross-validation covariance associated with the learned feature model. It is important to note that $\mathbf{R}$ serves several

purposes—it is simultaneously an overall indicator of the quality of the interpolation model, as well as the reliability of the matching phase that led to the observations that define the model; finally it also accommodates the stochastic nature of the sensor.

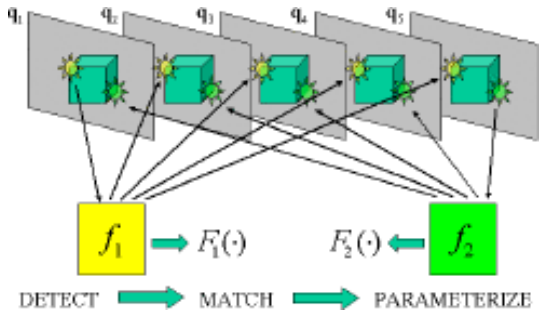


Fig. 2. Landmark learning framework: Salient features are detected in the input images and tracked across the ensemble. The resulting feature sets are subsequently parameterized as functions $F_i(\cdot)$ of the robot pose.

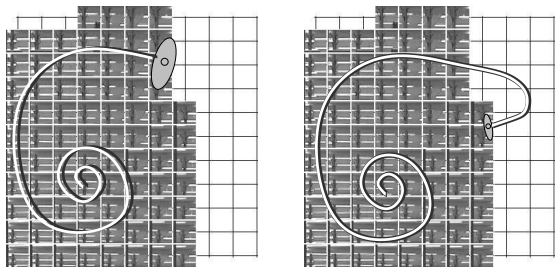
Building a Visual Map

The previous two sections described how to use a visual map to localize a robot, and how to do so using a Kalman filter. In order to actually build the visual map, we cannot assume (as we did previously) that the images are labelled with perfect odometry, so we use the Kalman filter to estimate the pose of each new observation based upon the previous observations. At each time step, given the new observations, the Kalman filter is updated according to the above formulation. Combined with the prediction model, a pose estimate and associated covariance are obtained. Once an updated pose estimate is available, the successfully matched features are inserted into the visual map, using the estimated pose as their observation pose. The features are initially selected using a model of visual saliency, and subsequently tracked over the pose space. The resulting models are then cross-validated in order to select only those features that demonstrate stability. It should also be noted that we use a gating procedure to detect outliers and remove outliers in the matching process. More details can be found in [2].

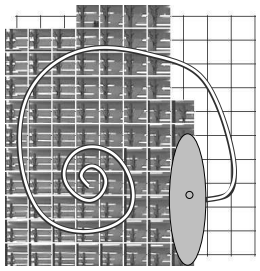
Figure 2 depicts the mapping algorithm: a set of images is collected at poses $q_1 \ldots q_5$, and features f_1 and f_2 are tracked in each image. The EKF gives a robust estimate q_i for each image. We then use interpolation to find a generative model $q = F_i(f)$ that allows us to compute the pose q for an observation of feature f_i.

III. Optimal Control for SLAM

For many environments, the extended Kalman filter approach has been shown to work well at building globally consistent maps that allow robots to track their position reliably. However, there is one issue with the algorithm that can cause difficulties in building large maps. As the robot senses new parts of the environment, it integrates the new visual features into the visual map. At some point, the exploration trajectory brings the robot back into previously explored space. If the robot has been able to maintain its pose estimate with high certainty, then robot should be able to detect its re-entry into explored space relatively easily. If, however, the robot's uncertainty has grown too large, or the robot has re-entered the map at a perceptually ambiguous location, then the robot may either make errors in maintaining its pose estimate, or even worse, fail to detect that it has re-entered the map.



(a) Before Exploration (b) Lower Uncertainty



(c) High Uncertainty

Fig. 3. Robot trajectory (spiral) and uncertainty ellipse (gray ellipse). Figure (a) illustrates a robot exploring within known territory. Figure (b) depicts a small excursion into unknown territory such that it retains a small uncertainty ellipse. Figure (c) shows a larger excursion that results in a much larger uncertainty ellipse. For such larger excursions, the uncertainty may become unrecoverably large and result in an inconsistent map.

Figure 3 illustrates the two possible scenarios. In figure 3(a), we see a robot trajectory (the white line) and the partial visual map it has constructed. In figure 3(b), we see the robot has covered a small portion of the unexplored space and then re-entered the map. Because the robot has only travelled a short distance through the unmodelled area, its positional uncertainty has only grown a small amount. In comparison, we see in figure 3(c) that the robot has travelled through most of the unexplored area. Lacking a map of this area the robot's uncertainty has grown substantially. When the robot re-enters the map, its uncertainty is sufficiently large to make accurate position estimation, even in the mapped regions, difficult. The robot may not be able to find correct correspondences between the sensed visual features and the features in the model. Even worse, the robot may not be able

to find enough correspondences, which would cause it to re-map this area leading to a globally inconsistent map. If we choose trajectories that can explore the space rapidly but allow us to return to the mapped regions sufficiently often to avoid tracking errors or mapping errors, then we can avoid such problems.

The approach we take is to use a one-step lookahead, choosing trajectories that maximize the space explored while minimizing the likelihood we will become lost on re-entering the map. In this case, our single step is over a path from the existing map through unexplored space to the first measurement inside the map. At every time step, we will choose a trajectory that will minimize our uncertainty as we re-enter the map, at the same time maximizing the coverage of the unexplored area. We use a parameterized class of paths, or policies, and repeatedly choose a parameter that maximizes our objective function.

Policy Class

We will use a parametric curve [6] expressed as a function of time that gives the the distance r of the robot from the origin as a function of time:

$$r(t,n) = \frac{kt}{2+\sin nt}, \tag{7}$$

where k is a dilating constant that is fixed for our experiments at $k = .2$ and n parameterizes the curve to control the frequency with which the robot moves toward the origin. We can re-write this in terms of the position $x(t), y(t)$ of the robot as

$$\begin{bmatrix} x(n,t) \\ y(n,t) \end{bmatrix} = \begin{bmatrix} \frac{kt}{2+\sin nt}\cos\left(\frac{\pi}{180}t\right) \\ \frac{kt}{2+\sin nt} sin\left(\frac{\pi}{180}t\right) \end{bmatrix}. \tag{8}$$

Some examples of the curve for a variety of values of n are shown in figure 4. Note that in the extreme cases, the curve never moves toward the origin ($n = 0$), or will do so with very high frequency ($n \to \infty$). Also of interest are integral values of n, where the curve never self-intersects, and has n distinct lobes. Finally, the rate of new space covered as a function of θ decreases roughly monotonically as n increases, since for larger n the robot spends an increasing amount of time in previously explored territory.

For a particular trajectory $r(\cdot, n)$, we define an objective function $q(n)$ for computing the optimal trajectory. We define q from t_0 to t_f as the amount of unexplored space covered from t_0 to t_f, reduced by the uncertainty of state estimate at time t_f. Let us use $U(r(t_i, n))$ as an indicator function, whether or not the pose of the robot at time t_i is in explored space:

$$U(r(t_i,n)) = \begin{cases} 0 & r(t_i,n) \text{ is in explored space} \\ d(r(t_i)) & r(t_i,n) \text{ is not in explored space} \end{cases} \tag{9}$$

where $d(r(t_i))$ is the Euclidean distance from the robot pose given by $r(t_i)$ to the nearest explored pose.

There are many choices for quantifying the uncertainty of the EKF filter at time t_f: we will use one of the most common functions, the determinant of the covariance matrix, $|Cov(t_f)|$. These two functions give us our objective function

$$q(n;t_0) = \sum_{t_i=t_0}^{t_f} U(r(n,t_i))^2 - \lambda|Cov(t_f)|. \tag{10}$$

This function contains the one free parameter λ that allows us to calibrate how aggressive the exploration of unknown space should be compared with building a high-accuracy map.

The One-step Controller

Given the policy class $p_n(t)$ and objective function $r(p_n(t))$ described above, we want to find n to maximize $r(p_n(\cdot))$. Given a particular state of the EKF at time t_0, we cannot compute the covariance at time t_f in closed form, so we use numerical simulation, projecting the Kalman filter forward until the trajectory enters unexplored space and then returns into the map. We discretise n and evaluate $q(n_i)$ for each choice of n. The complete control algorithm is summarised in table I.

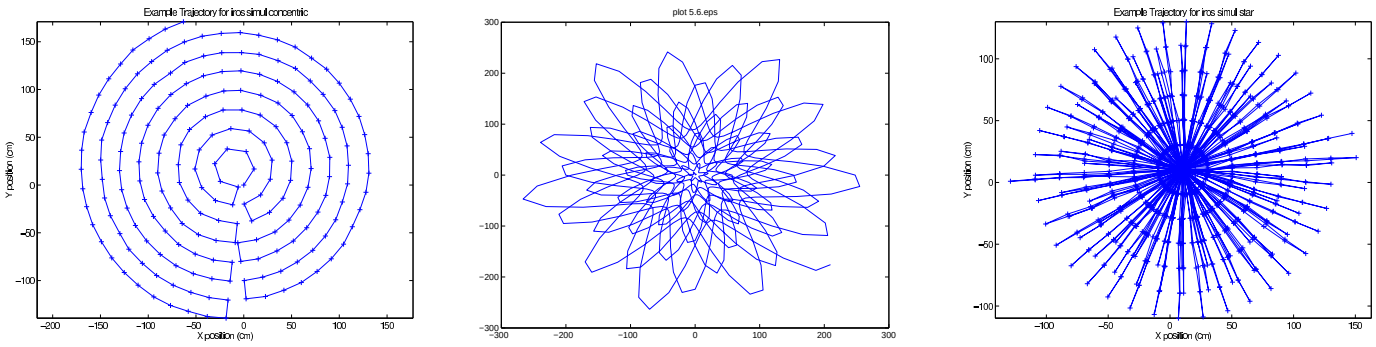
- At each time step t_0,
- For value of $n_i \in [0, n_{max}]$:
  1) Simulate forward $r(t_i, n_i)$ from t_0 until $U(r(t_i, n_i)) = 1$ or $t_i = t_{max}$
  2) Simulate forward $r(\cdot, n_i)$ from t_0 until $U(r(t_i, n_i)) = 0$ or $t_i = t_{max}$
  3) Set $t_f = t_i$
  4) Set $q(n_i) = \sum_{t_i=t_0}^{t_f} U(r(n,t_i)) - \lambda \det(Cov(t_f))$
- Set $n = \text{argmax}_{n_i}\, q(n_i)$

TABLE I
THE SLAM CONTROLLER

IV. EXPERIMENTAL RESULTS

We ran our experiments in a simulated office-like environment in order to obtain accurate ground truth. The environment was composed of a single $12m \times 6m$ rectangular room, with images from a real laboratory environment texture-mapped on to the walls. Visually, the environment was somewhat simplified compared to what the robot might encounter in a real-world setting. However, our experience indicates that the visual mapping framework is particularly prone to selecting environmental features that correspond to planar patches. In this sense, the simulated environment presented the visual mapping framework with the best possible scenario and we could concentrate on the behaviour of the framework due to odometric and modelling error.

The simulated robot had a ring of sixteen evenly spaced sonar sensors which are employed solely for detecting collisions. The robot's odometry model was set to add normally distributed zero-mean, 1% variance error to translations and normally distributed zero-mean, 1% variance error to rotations. Each observation was collected by placing a simulated camera at the ground truth pose of the robot, and taking two images, one along the global x axis and one along the y axis. It was assumed that in a real-world setting the camera has the



(a) Concentric Policy (b) n=5.6 (c) Star Policy

Fig. 4. Sample trajectories for a variety of controllers. (a) and (c) heuristic policies that represent the extrema of n: the `Concentric` and `Star` policies. (b) A policy in the intermediate range of $n = 5.6$.

ability to align itself using a procedure which is external to the robot drive mechanism, possibly using a compass and pan-tilt unit or an independent turret, such as that which is available on a Nomad 200 robot. A single observation was defined as the composite image obtained by tiling the two images side by side. Figure 5 illustrates a typical image returned by the camera in one direction in the simulated environment.

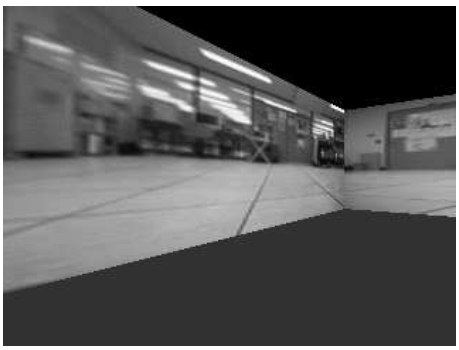


Fig. 5. Simulated camera view.

The experiments were conducted as follows: the robot was placed at the centre of the room, and the trajectory $r_n(t)$ was executed over five degree increments in t. At each pose, an observation was obtained and the Kalman filter was updated. The visual map was updated whenever the filter indicated that the robot was more than 6.7cm from the nearest observation in the visual map, and exploration terminated when 100 images had been added to the map. The constant k in equation (7) was set to $0.2m$. The ground-truth pose, the filter pose and the control inputs were recorded for each pose along the trajectory.

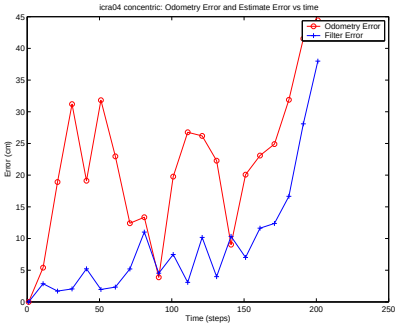
For the purposes of comparison, two additional exploratory policies were run, that represent the extrema of the different n settings. The `Concentric` policy involves tracing a series of concentric circles, reversing the direction of exploration with each sweep, as shown in figure 4(c). This curve also corresponds to $n = 0$. The `Star` policy returns to the origin to re-localize after each new image was added to the map, as shown in figure 4(b). This curve corresponds to a large setting of n.

Figure 7(a) depicts the trajectory traced by the robot for our exploration algorithm. The red curve, marked with 'o' symbols, corresponds to the ground truth position of the robot at each point along the trajectory, whereas the blue curve, marked with '+' symbols, traces the pose estimate computed by the Kalman filter. For the purposes of our analysis, we note that small rotation errors at the outset of the trajectory can lead to a map that correct, apart from a small rotation about the origin, and correct this rotation versus the ground truth. While in practise this information would not be available to a real robot, the rotation error would not affect the utility of the constructed map.

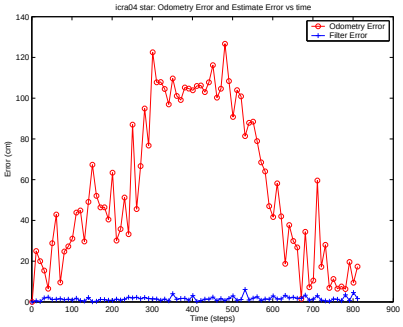
Figure 6 plots the deviation from ground truth for the Kalman filter ('+') and the robot's odometer ('o'). We define the mapping error to be the mean Kalman filter error over all time, and these results are reported for each policy in figure 7(b). In order to measure efficiency, Figure 7(c) plots the length of the trajectory for the three methods. Clearly, the one-step policy represents a significant improvement in efficiency over the `Star` policy, and is only about 50% more costly than the `Concentric` policy.

V. Related Work

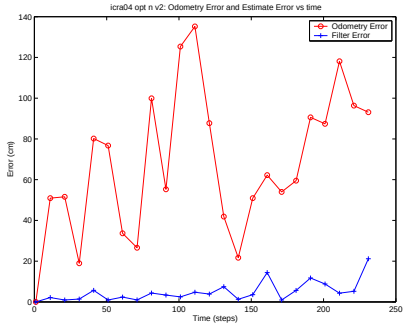
There exist planning methodologies that can exactly compute optimal plans or policies in expectation over more than a one-step horizon, e.g., by maintaining a distribution over possible states of the world and computing the strategy that is optimal in expectation with respect to that distribution. One such approach is the Partially Observable Markov Decision Process, or POMDP [7], [8]. However, the major disadvantage of the POMDP for our control problem is computational intractability. Most POMDP solution algorithms are known to be unable to scale to problems with more than a few hundred discrete states [9], [10], and we would like to compute the optimal trajectory over all possible maps, a continuous state space that is not amenable to standard POMDP formulations.



(a) Concentric policy

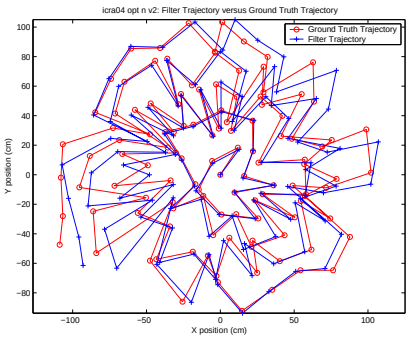


(b) Star policy

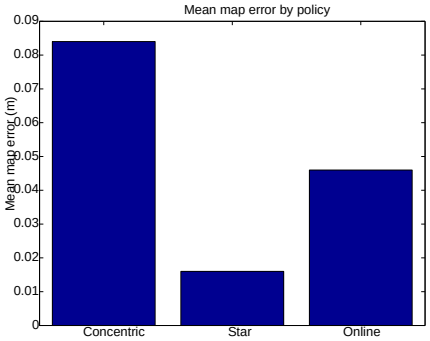


(c) One-step policy

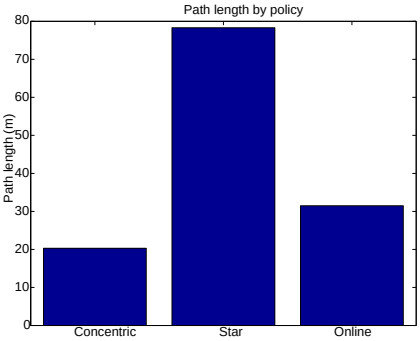
Fig. 6. Odometric error ('o') and Kalman filter error ('+') at each time step for each policy. Notice that the one-step policy has average Kalman filter error competitive with the `Star` policy over time, but has higher variance because of the greater exploration rate.



(a) One-step policy: Mean error 4.6cm



(b) Mean Path Error



(c) Mean Path Length

Fig. 7. Exploration results. (a) The ground truth trajectory of the robot (red 'o' curve) is plotted against the inferred map (blue '+' curve). (b) Mean path error by policy. (c) Path length by policy.

Our work is an instance of the problem of simultaneous localization and mapping (SLAM). This problem has received considerable attention in the robotics community [3], [4], [11], [12], [13], primarily in the context of computing range-based maps with spatially localized features. The state of the art can be broadly subdivided into one of two approaches (and various hybrids). One family of methods relies upon unimodal estimates, such as the Kalman filter description presented here. The map is represented as a set of landmarks derived from a range sensor, and a Kalman filter or particle filter is employed to minimize the total uncertainty of the robot pose and the individual landmark positions [14], [15]. A second family of approaches uses more complicated representations such as particle filters or mixture models [11]. There are hybrid approaches [16], [17] that reduce the computational expense, as each update for previous approaches is quadratic in the number of landmarks.

Of particular relevance to this paper is the problem of planning a trajectory for minimizing uncertainty while maximizing the utility of the observed data. MacKay considered the problem of optimally selecting sample points in a Bayesian context for the purposes of inferring an interpolating function [18]. Whaite and Ferrie employed this approach as motivation for their 'curious machine', a range-finder object recognition system that selected new viewing angles in order to optimize information gain [19], and Arbel and Ferrie further applied this approach to appearance-based object models, selecting the viewing angle that maximized the ability to discriminate between objects [20].

VI. Discussion and Future Work

We have presented an algorithm for controlling a mobile robot during exploration, that allows us to build globally consistent maps quickly and automatically. We made use of a class of control strategies described by parametric curves. The algorithm updates the parameter setting greedily at each time step, choosing the parameterization that maximizes the objective function in expectation. We examined the performance of this algorithm on the task of building a visual map, and we showed the pose error of the online control strategy was in general competitive with the single setting that led to the most accurate map, while at the same time building the map

at a rate competitive with the most aggressive setting.

The online strategy currently contains at some open problems. Firstly, the reward function contains an explicit trade-off between exploration and map accuracy, represented by the free parameter λ. Ideally, the optimal control representation would not contain any free parameters. We may be able to eliminate this free parameter by choosing a different reward function, but this is a question for further research. Secondly, the particular parameterisation may not be the best policy class. This policy class is somewhat restrictive, in that the single parameter essentially represents a frequency of returning to the origin. We may be able to achieve better results using a more general policy class, such as one of the stochastic policy classes in the reinforcement learning literature. Finally, the strategy is greedy, in that it attempts to choose the trajectory based on a single projection into the future. However, as more of the map is acquired, it may become possible to infer unseen map structure and make more intelligent decisions, leading to even better performance in more structured and regular environments.

REFERENCES

[1] R. Sim and G. Dudek, "Effective exploration strategies for the construction of visual maps," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, October 2003.

[2] ——, "Learning generative models of scene features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Hawaii, 2001.

[3] R. Smith, M. Self, and P. Cheeseman, *Autonomous Robot Vehicles*. Springer-Verlag, 1990, ch. Estimating uncertain spatial relationships in robotics, pp. 167–193.

[4] J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot." in *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, Osaka, Japan, November 1991, pp. 1442–1447.

[5] J. Shi and C. Tomasi, "Good features to track." in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[6] R. Sim and G. Dudek, "Effective exploration strategies for the construction of visual maps," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) workshop on Reasoning with Uncertainty in Robotics (RUR)*, Acapulco, Mexico, 2003, pp. 69–76.

[7] E. Sondik, "The optimal control of partially observable Markov decision processes," Ph.D. dissertation, Stanford University, Stanford, California, 1971.

[8] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains." in *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI)*, Seattle, WA, 1994.

[9] N. Roy and G. Gordon, "Exponential family PCA for belief compression in POMDPs," in *Advances in Neural Information Processing Systems 15 (NIPS)*, S. Becker, S. Thrun, and K. Obermayer, Eds. Vancouver, BC: MIT Press, 2003, pp. 1043–1049.

[10] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, Acapulco, Mexico, August 2003.

[11] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning*, vol. 31, pp. 29–53, 1998.

[12] B. Yamauchi, A. Schultz, and W. Adams, "Mobile robot exploration and map building with continuous localization." in *Proceedings of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 1998, pp. 3715–2720.

[13] P. P. Hans Blaasvaer and H. I. Christensen, "Amor: An autonomous mobile robot navigation system," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, 1994, pp. 2266–2271.

[14] J. J. Leonard and H. J. S. Feder., "A computationally effi cient method for large-scale concurrent mapping and localization." in *Robotics Research: The Ninth International Symposium*, J. Hollerbach and D. Koditschek, Eds. London: Springer-Verlag, 2000.

[15] J. G. E. Nebot and H. Durrant-Whyte, "Simultaneous localization and map building using natural features in outdoor environments." in *Sixth International Conference on Intelligent Autonomous Systems*, Italy, 2000.

[16] 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*. Edmonton, Canada: AAAI, 2002.

[17] M. A. Paskin, "Thin junction tree fi lters for simultaneous localization and mapping," in *Proceedings of the Eighteenth International Joint Conference on A rtificial Intelligence (IJCAI-03)*, G. Gottlob and T. Walsh, Eds. San Francisco, CA: Morgan Kaufmann Publishers, 2003, pp. 1157–1164. [Online]. Available: citeseer.nj.nec.com/600271.html

[18] D. MacKay, "Information-based objective functions for active data selection." *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.

[19] P. Whaite and F. P. Ferrie, "Autonomous exploration: Driven by uncertainty." in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Los Alamitos, CA, June 1994, pp. 339–346.

[20] T. Arbel and F. P. Ferrie, "Viewpoint selection by navigation through entropy maps." in *Seventh IEEE International Conference on Computer Vision*, Kerkyra, Greece, 1999.