MIT 16.412J Spring 2004
Final Project Writeup
Vikash K. Mansinghka

Towards Visual SLAM in Dynamic Environments

Abstract

To build autonomous robots capable of operating wherever humans do, we must develop localization and mapping strategies that can handle changing environments. Recent work by Se, Lowe and Little has shown that current machine vision technology makes visual SLAM feasible in a potentially wide range of static environments. However, their system treats vision system errors in a manner which imposes a fundamental limit on its ability to handle environments with motion. In this paper, I first identify important limitations of Se et al.'s error handling approach common to virtually all metric SLAM systems and, based on this analysis, propose two techniques for overcoming these limitations. I also present a third technique, based on optical flow, which specifically addresses the issue of visual SLAM in dynamic environments. Finally, I discuss two proof-of-concept systems I built which explore the usefulness of this third technique in the context of place recognition.

Contents

1 Introduction **3**

2 Sensor Error Treatment and Data Association in SIFT SLAM **3**

2.1 SIFT SLAM Strongly Trusts Sensor Data 5

2.2 Predictions for SIFT SLAM in Dynamic Environments 6

3 Techniques for Improving Visual SLAM Performance **6**

3.1 Tuning Thresholds to Minimize Uncertainty 6

3.2 Identifying Outliers 7

3.3 Identifying Motion through Optical Flow 7

4 Optical Flow for Place Recognition **8**

4.1 Common Infrastructure 8

4.2 Simple Demonstration of Keypoint Rejection 8

4.3 Use of Optical Flow in Place Recognition 9

4.4 Choice of Optical Flow Scheme 11

5 Conclusions **12**

1 Introduction

To build autonomous robots capable of operating wherever humans do, we must develop localization and mapping strategies that can handle changing environments. Visual strategies are especially desirable, as they have the potential to produce substantially more useful maps than other approaches (e.g. with denser features at multiple height levels). Recent work by Se, Lowe and Little has shown that current machine vision technology makes visual SLAM feasible in a potentially wide range of static environments. However, their system treats vision system errors in a manner which imposes a fundamental limit on its ability to handle environments with motion.

In this paper, I first identify limitations of Se, Lowe and Little's error handling approach common to virtually all SLAM systems and, based on this analysis, propose two techniques for overcoming these limitations. I also present a third technique, based on optical flow, which specifically addresses the issue of visual SLAM in dynamic environments. Finally, I discuss a proof-of-concept system I built which explores the usefulness of this third technique in the context of place recognition.

It is worth noting that the original focus of my course project was the use of optical flow to enable visual SLAM in dynamic environments. While working with this idea, I made the observations about Se, Lowe and Little's error treatment and possible improvements that I outlined above; I believe these observations constitute the meat of my project.

2 Sensor Error Treatment and Data Association in SIFT SLAM

Virtually all metric SLAM systems operate by predicting relative landmark positions based on relatively coarse motion estimates, interpreting sensor data to identify measured landmark positions based on those estimates, and revising position accordingly. Predicted landmark positions are a key element of the data association process in virtually all metric SLAM systems, as follows. If SLAM landmarks cannot be distinguished by a sensor system other than on the basis of their position (as with the "flat region" landmarks typically produced by sonar or laser range analysis), they must be chosen very carefully to avoid confusion with other landmarks. In practice, this has resulted in the use of a "validation gate", a set of thresholds governing how close, in position, a measured landmark must be to an expected landmark to be considered a match.

In SIFT SLAM, the validation gate takes the form of a pixel window (10x10 in Se, Lowe and Little's paper) centered on predicted position in which landmarks must be to get matched, as well as similar constraints on disparity and SIFT scale (derived from depth). Although SIFT SLAM includes a matching threshold for the SIFT orientation component of a landmark, a measure of its actual image content, this threshold is *not* part of the validation gate, as its value has nothing to do with the position of the landmark. Note that as SIFT feature matching is a visual process, the difference between matched and expected SIFT feature position can be viewed as an estimate for the optical flow or image motion at the location of the SIFT feature being matched.[1]

[1]It might perhaps be better to term the difference as measured motion rather than optical flow, as "sparse optical flow" is somewhat of an oxymoron, and overloads the technical definitions of optical flow - as opposed to motion estimates - as "apparent

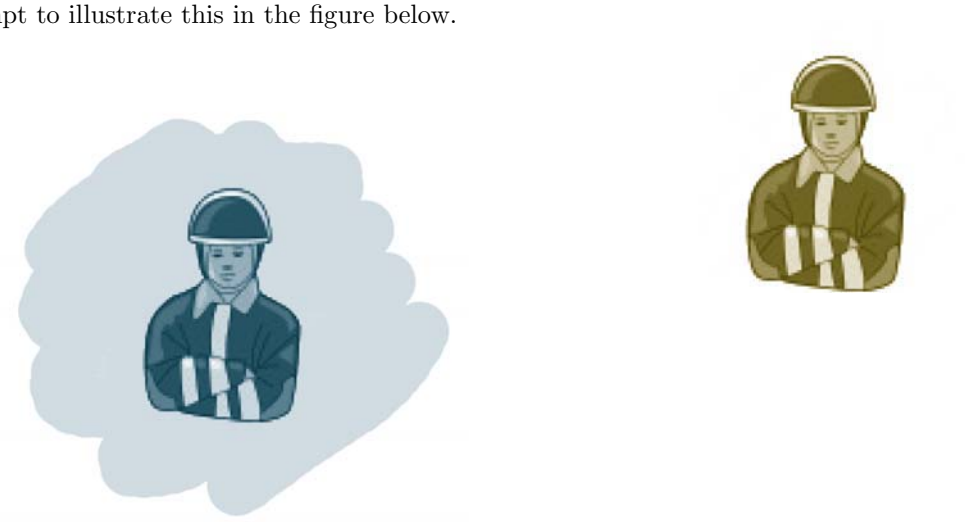
The parameters for this gate impose hard limits on the performance of the SLAM system in the following ways:

1. They imply a bound on the maximum odometry error that the system can handle, as if odometry readings are sufficiently far off, no landmarks previously known about by the sensor system will fall within the validation gate boundaries.

2. Perhaps more intuitively, they imply a maximum sensor error in locating landmarks, as again if the sensor's reading for a given landmark is sufficiently far off, it will not be counted.

3. If the sensor system cannot distinguish between landmarks at all other than by their position - that is, if a landmark is simply a signal that is stably extractable from the world at a given position - the validation gate imposes a limit on the closeness of landmarks. This is because if two landmarks were chosen with overlapping validation regions, they could not be distinguished by the system. Note that in SIFT SLAM, the vision system *can* distinguish between landmarks on a basis other than their position; I will discuss the implications of this later.

4. They imply a limit on the motion resolution of the SLAM system. That is, if the SLAM system attempted to notice landmark motion (either while moving, in which case the detected landmark motion would be first computed relative to the robot's motion, or while stationary) it would only be able to do so for landmarks moving faster per sensor processing cycle than the validation gate. I attempt to illustrate this in the figure below.



In this figure, the robot is represented by Mario and a slowly moving "adversary" by Wario. As long as wario's motion is within the grey region, corresponding to the validation gate, Mario will think his position estimate is off and adjust it accordingly. While other landmarks not exhibiting those errors might help to reduce the impact of Wario's motion, its effect will be nonzero, and possibly quite significant.

motion of the brightness pattern in an image".

2.1 SIFT SLAM Strongly Trusts Sensor Data

The overarching conclusion of many of David Lowe's evaluations of SIFT is that it is remarkably reliable; two SIFT features are relatively rarely matched when the real-world objects on which those features are located do not "look the same" to people. The stability results in his paper indicate that the measured parameters of SIFT features (their scale and orientation) are relatively stable at least to small displacements.

SIFT SLAM exploits this fact in its position estimation scheme by essentially circumventing the bulk of traditional EKF SLAM. It first treats the odometry reading as both the measurement and prediction for the robot pose, as opposed to treating the odometry reading as the prediction and the individual position observations for landmarks as the measurements. The H matrix for the "Kalman filter" SIFT SLAM uses is simply the identity.

It then incorporates landmark positions by computing a least-squares best-fit position minimizing errors between predicted landmark pixel coordinates in its view and the observed ones. Even this process throws away information, as the measured depths of the landmarks are used only indirectly. The error terms being minimized in SIFT SLAM are below, taken from Se, Lowe and Little's paper, where r_i' is the predicted pixel row position of matched landmark i and r_{mi} is its detected pixel position:

$$e_{ri} = r_i' - r_{mi} \tag{1}$$

$$e_{ci} = c_i' - c_{mi} \tag{2}$$

Although the depth of a landmark is used to predict these values, SIFT scale error - a measurement analogous to pixel position produced by SIFT - is not directly minimized. Actually, SIFT SLAM computes two least-squares fits, using the first one to identify outliers (according to a standard residual measure, with a threshold of 2 pixels) and the second one to produce an estimate using only the non-outliers.

SIFT SLAM then apparently computes a measure of the quality of the least-squares fit and treats it as a kind of "measurement uncertainty", adding it to the position uncertainty predicted from its motion model. It then feeds this total uncertainty estimate into a Kalman-filter-like procedure, whose actual equivalence to an EKF I have not been able to verify, to weight the least-squares position estimate and the odometry estimate. The output is a position estimate along with a position uncertainty which is later fed into independent EKFs for the each tracked landmark.

This structure reflects a strong trust in sensor readings in two main ways. First, its definition of an outlier in the least-squares procedure is quite stringent. This means all the offsets in measured positions must be quite coherent, reflecting primarily motion and not sensor error. This is a somewhat subtle point, which I have not had the opportunity to verify in simulation, but worth investigating further. Second, it does not apply a full EKF to the SLAM problem; instead, it decouples it into pose estimation and landmark estimation, restricting landmark uncertainty to be independent (ala FastSLAM). The choice to throw away these sources of information reflects the confidence the SIFT SLAM designers had in the quality of SIFT matching. It also raises the issue that the uncertainty estimates produced by SIFT SLAM do not mean the same things as the uncertainty estimates produced by many other metric SLAM algorithms.

It would be interesting to check if the position estimates produced by SIFT SLAM vary significantly

from the least squares estimates produced by its vision system; if so, SIFT SLAM would be trusting its sensors almost absolutely. It is also worth noting that the sensor error "model" used by SIFT SLAM in the per-landmark EKFs use an arbitrarily chosen variance of 1 pixel; this value is quite low, another reflection of confidence. It is not as compelling, however, because SIFT SLAM does not appear to use its landmark EKFs for for much, given its decoupling of the state estimation.

2.2 Predictions for SIFT SLAM in Dynamic Environments

The specific choices made in SIFT SLAM suggest it will actually perform reasonably well in dynamic environments, as long as motion is fast enough that it doesn't introduce the validation gate errors I discussed earlier. This is because if a landmark is not matched (when expected) enough times, SIFT SLAM throws it away; landmarks on a moving person, then, are likely to get thrown out if the person moves too much. If a person moves slowly for awhile, it could conceivably corrupt the robot's position estimate a bit, but - combined with the possibility of rejection as an outlier in the least-squares estimate - it seems unlikely that the majority of moving bodies will confuse SIFT SLAM.

An interesting exercise would be to predict the motion thresholds implied by SIFT SLAM's error handling choices to determine what "amount" of motion in a scene is likely to fool it, or whether or not sufficiently slow motion is likely to be a problem in any real SLAM scenarios.

3 Techniques for Improving Visual SLAM Performance

The above analysis suggests several possible improvements to SIFT SLAM. Some of these may even be usable in other SLAM systems.

3.1 Tuning Thresholds to Minimize Uncertainty

The default threshold values provided in SIFT are were apparently set by Se, Lowe and Little based on their experience with the SIFT system. It is not at all clear that they were optimally chosen. Given the uncertainty measurements produced by virtually all metric SLAM algorithms, however, it should be possible to search for better threshold values, perhaps on a per environment basis.

A naive scheme for doing so would involve running SIFT SLAM (or perhaps a very similar SIFT-based SLAM setup using FastSLAM at its core) and measure position or total uncertainty (e.g. covariance magnitude or particle entropy) for various choices of validation gate values, matching thresholds, and landmark tracking miss count thresholds. The thresholds which produced the best output at the end of a run could then be used in future SLAM scenarios in a similar environment. Alternately, if the cost of evaluating multiple thresholds is small, the thresholds could be set adaptively (perhaps every few frames) over the course of a given SLAM task. This might help improve SLAM performance in tasks where SIFT performance varies, such as transition from regions with identifying marks on walls to fairly homogeneous ones.

Of course, it is not clear that minimizing an uncertainty measure will actually improve SLAM accuracy, as the world might conspire to produce sensor readings with lower uncertainty in an incorrect location. In fact, the motion/validation gate problem I discuss above is an instance of this, where minimizing uncertainty

via standard SLAM approaches naively results in erroneous position estimates. However, it is not clear if this will be a problem in real world applications, and in the worst case tuning thresholds during "training" SLAM runs where absolute position can be tracked might still improve SLAM performance.

3.2 Identifying Outliers

The central idea behind most metric SLAM systems is that motion errors should be consistently reflected in deviations from predicted and measured landmark positions (modulo sensor uncertainty). Therefore, even in the presence of slowly moving objects, the vast majority of the landmarks in a scene should deviate from predicted values in a consistent manner. Relative motion of some landmarks should show up as a cluster of landmarks with a different (but internally consistent) kind of deviation from motion error.

If these clusters could be programmatically detected, they could be used to rule out deviations from prediction which are more likely to correspond to landmark motion than robot motion error. Identifying these clusters via PCA or even some kind of regression might be possible. One interesting question is whether or not the least-squares fit with rejection of outliers in SIFT SLAM actually produces this effect to some degree.

3.3 Identifying Motion through Optical Flow

A central problem for SLAM in dynamic environments is the rejection of landmarks which actually lie on moving objects. The rationale is simple: if landmarks seem to move, any SLAM scheme will interpret this motion as an error in its position estimates, and erroneously revise them.

The problem, then, is to avoid selecting landmarks on things which can move. A reasonable first step towards this goal, reachable given current machine vision technology, is to avoid selecting landmarks on objects which are moving. As outlined above, this problem is currently intractable when the robot performing SLAM is moving, as it cannot distinguish between small errors in its sensor apparatus or motion model and small motions. A workable solution, however, may be to only assign new landmarks when the robot is stationary, and sensor uncertainty is therefore small, so all measured motion of landmark candidates is likely to correspond to real landmark motion.

The measurement of landmark motion could proceed in at least two ways. First note that he latest incarnation of SIFT includes "keypoint descriptors", a better measure of feature content than the SIFT orientations used in SIFT SLAM. If keypoint descriptors can be matched reliably, it would be interesting to see how SLAM performance is affected by only using matching thresholds on SIFT properties, not positional properties. If this did not result in large numbers of false matches, the SIFT system could then use landmark matching to estimate landmark motion when the robot is stationary, and reject those landmarks which moved more than some very small amount. This would essentially amount to setting the validation gate to some very low value as long as the robot is stationary, driving the speed needed to confuse the system to a value low enough so as to be irrelevant.

As long as the system did not acquire new landmarks while moving (stopping to acquire new landmarks whenever necessary), the system would be much less foolable by motion in the environment. Given the density of landmarks extracted via SIFT (as demonstrated in Se, Lowe and Little's paper), this should not

seriously impede the robot's ability to move; a robot should be able to travel a considerable distance (both angular and linear) before needing to acquire new landmarks. While the system might still assign landmarks to an object in the environment which moved later on, temporarily confusing the SLAM system, the standard miss count thresholds might be good enough to minimize impact.

4 Optical Flow for Place Recognition

As a proof of concept the usefulness of optical flow in pruning landmarks associated with motion given a stationary camera, I built two simple systems which used Horn-Schunck optical flow to prune SIFT keypoint lists. The first system, whose results must be confirmed visually, was used to test if the idea was at all reasonable. The second system was an attempt to use optical-flow-based keypoint rejection to improve the performance of a naive place recognition system.

4.1 Common Infrastructure

Both systems were implemented in C (compiled with gcc 3.3.3 on Debian Linux) and made use of the binary version of David Lowe's SIFT keypoint extractor (available from http://http://www.cs.ubc.ca/spider/lowe/) and Intel's Open Source Computer Vision Library (OpenCV - version 0.9.4-1 from the `opencv1` and `opencv1-dev` Debian unstable packages). OpenCV was used for simple image input, drawing and output, as well as for optical flow computation via the classical Horn-Schunck algorithm.

Input MPEG movies were obtained either from the International Ray Tracing Competition (located at `http://www.irtc.org`) or from a Logitech Quickcam (used with the `pwc` driver) and the `ffmpeg` linux encoder program. Frame sequences in PPM format were extracted from each movie using `transcode` (a LINUX video stream processing tool) then converted to PGM format using ImageMagick's `mogrify` tool for input into OpenCV and Lowe's keypoint extractor.

For both systems, the keypoint matching procedure used was based on David Lowe's default keypoint matcher. It assumes a euclidean distance measure between keypoint descriptors[2]; the details of this measure are not important. The rough idea is that it compares the contents of the image around each match candidate to the keypoint to be matched, say m, and declares a match if the closest point to m is within some distance threshold and the next closest point is sufficiently far away from it. Note that this matching strategy includes no positional information; instead, it relies on the specificity of the keypoint descriptors.

4.2 Simple Demonstration of Keypoint Rejection

The first proof-of-concept system I constructed accepts a series of frames as input and produces the list of keypoints from the first frame which could be stably matched in each successive frame and were not located on an image region with substantial motion. The goal was to demonstrate that optical flow can reliably prune landmarks on moving bodies (if the camera is stationary) in a way that could be visually confirmed.

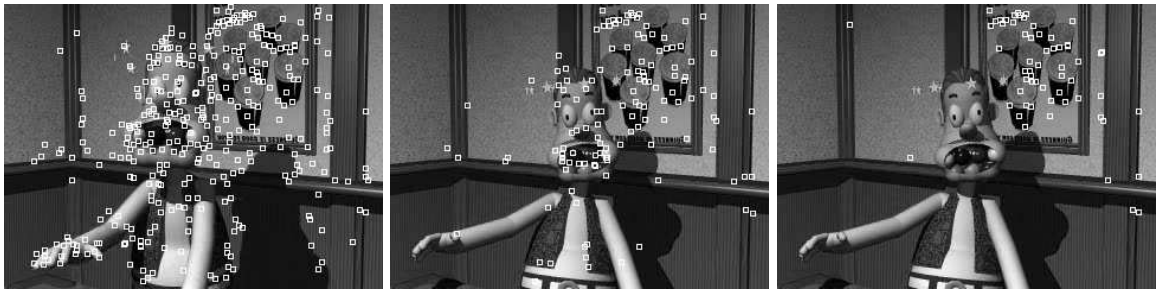
[2]This is a measure of image content around the location of a keypoint; for details, see Lowe's 2004 IJCV paper

The system executable is called `simple`. It accepts two command-line arguments. The first is the prefix for the filename of all the frames, and the second is the number of frames. Frame filenames must be of the form `<prefix><6 digit number>.pgm`; frame numbers must start at 0. For example, to run the system on the first 40 frames of a movie where the first frame is `foo000000.pgm`, the command line is `simple foo 40`.

The matching criteria used by the system[3] are as follows (with all counts initially 0):

1. If a feature is matched at an image point with flow below the current threshold, set its counts to 0.
2. If a feature is matched at an image point with flow above the current threshold, increment its moved count. If this exceeds the movement threshold, discard it.
3. If a feature is not matched at all, increment its missed count. If this exceeds the miss threshold, discard it.

Note that these thresholds are analogous to those in the SIFT landmark tracking system. The results of running this system on frames 500-550 of the "Pool Shark" entry to the IRTC - extracted from the included file `poolshrk.mpg` - for the first 39 frames is shown below:



Courtesy of Neil Alexander. Used with permission.

The image on the left shows all keypoints found in the first frame. The middle image shows those keypoints which survived given a flow threshold of 30 (essentially arbitrarily large flow), a moved threshold of 5 frames and a miss threshold of 10 frames. The image on the right shows those keypoints which survived given a flow threshold of 0.1 (essentially no flow). All landmarks on the moving person have been pruned.

This serves as a useful proof of concept of the usefulness of an optical flow scheme for rejecting features on moving bodies.

4.3 Use of Optical Flow in Place Recognition

The second system I built tests the usefulness of optical flow for rejecting spurious features in the context of a naive place recognition system. It should be noted that the usefulness of flow in the context of SLAM is not at all measured by these results; a detailed explanation of my rationale is included below.

The naive place recognition system, embodied in the executable `placerec`, operates on two input movies (specified as for the simple optical flow demo above). The first is the training movie, containing an image

[3] In traditional bad-quality research coding style, these are set in the code; the current build of simple is set to an extremely forgiving optical flow threshold essentially corresponding to no use of flow. C'est la vie.

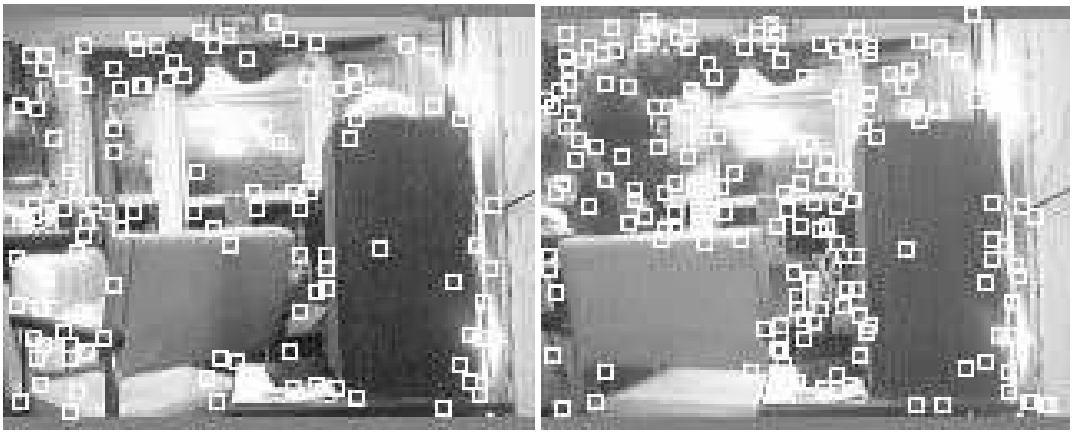
sequence from which features will be extracted defining the place. The second is the test movie which will be compared against the training sequence on a frame-by-frame basis.

The place recognition system is extremely naive. It builds up a place definition by performing the same process as the simple demo above - that is, it defines a place to be a collection of keypoints extracted from the first place frame which were stably matched (and stationary) over the entire training sequence. For each test frame, it applies the following procedure:

1. Extract the keypoints from the frame.
2. For each keypoint, determine if it lies in a region with flow below a certain threshold. If so, keep it.
3. Store the number of kept keypoints.
4. For each kept keypoint, check if it matches a keypoint in the place definition. If so, increment the number of matches.
5. If the percent of matches - that is, the percent of *kept* keypoints which matched a keypoint in the place - exceeds a threshold, declare the frame to be from the training place.

The idea is straightforward enough: determine which keypoints correspond to "environmental" rather than "transient" features, and check how many of them match. It is naive chiefly in that it does not attempt to weight features according to how informative they are, and that it has a single simple threshold for declaring a match.

Snapshots of the features selected by the system from a movie taken in an apartment in Somerville are shown below. The image on the top left shows the keypoints extracted from the first 30 frames of the training movie (`dh-still`). The image on the top right shows the features identified in frame 20 without optical flow pruning; note the features on the interloper. The image on the bottom left shows the optical flow generated by the interloper's distracting dance moves, and the image on the bottom right shows the features identified given an optical flow threshold of 1.0.



Given a matching threshold of 20%, optical flow pruning improved recognition performance by roughly 13% (from a 43% frame recognition rate to a 56% frame recognition rate on a sequence of 30 frames). The average percent of matches increaesd by roughly 2%.

Optical flow only resulted in a small increase in performance primarily because the real-world test movies did not include enough motion distractors. A secondary factor was, again, the naive nature of the place recognition scheme. An HMM approach without flow would either be confused by features on moving objects (not present in all footage of a given place) or would assign those features low probability; optical flow could potentially prune those features entirely.

The system was also tested on video footage from MIT's Lobby 7 as well as other sequences from the same apartment in Somerville. For the majority of these scenes, place recognition performance in general was poor enough (that is, few enough keypoints matched unambiguously) that the system couldn't afford to discard keypoints based on optical flow; the few points discarded due to errors in the optical flow computation were too costly[4].

This poor performance was primarily due to two factors:

1. The resolution of the movies produced by the Quickcam was 160x128; the SIFT feature count was fairly low to begin with. with.

2. The camera placement compounded the low feature problem by including large homogeneous regions. These regions result in ambiguous matches (in the absence of position information as provided by stereopsis) which are rejected by the keypoint matching procedure I outlined above.

4.4 Choice of Optical Flow Scheme

It should be noted that it would have also been possible to use SIFT feature matching to approximate image motion instead of appealing to Horn-Schunck. This was not done initially as the idea did not occur to me. However, it is not clear that SIFT matching would have performed better.

[4]Detailed results on these test movies was not tabulated, as these were not relevant to the main thrust of the project, and would not be too informative in any case given the simplicity of the place recognition approach.

The strength in SIFT lies in its feature density, permitting substantial matching errors such as omissions or mismatches to be dwarfed by the number of correct matches in recognition tasks. The precise location of individual SIFT features, however, is not terribly stable; changes in the overall image content such as the addition of new objects or small fluctuations in brightness can cause individual feaures to jump several pixels or vanish entirely. These errors are evident in the feature location movies I provided.

In SIFT SLAM, the errors introduced by these jumps are minimized due to the use of 3D feature position in all matching tasks along with moderately stringent rejection thresholds. In a typical place recognition scenario, however, the camera position is not known; the only constraint is that the view from which a place is to be recognized overlaps substantially with training views. Geometric information is therefore unavailable and cannot be used to aid in the matching task.

While I could have designed a special-purpose SIFT matching approach incorporating the particular constraints of optical flow in the place-recognition context, this did not seem particularly relevant to my overall goals. Horn-Schunck's optical flow scheme, on the other hand, is formulated to incorporate substantial geometric information.

5 Conclusions

In this paper, I argued that **validation gates impose fundamental limits on SLAM accuracy, especially in the presence of motion**. I suggested several techniques for addressing this issue as well as other error handling issues in SLAM, such as **tuning thresholds to minimize uncertainty**, **detecting outlying displacements**, and **pruning landmarks based on optical flow**. Finally, I **built and described two systems using optical flow to prune landmarks** and tested them in the context of a naive approach to place recognition.

The conjectures in this paper cannot be tested without a reimplmentation of the SIFT SLAM system. A successful reimplementation which was capable of navigating in noisy environments, such as Lobby 7 during lunchtime, would constitute a significant contribution in robotics. I have confidence that some combination of the ideas in this paper (or perhaps even an only slightly modified version of SIFT SLAM itself) would be able to achieve that goal, and I hope to have the chance to investigate this in the future.