ROBUST TRACKING AND STRUCTURE FROM MOTION WITH SAMPLING METHOD

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY IN ROBOTICS

Robotics Institute
Carnegie Mellon University

Peng Chang

July, 2002

Thesis Committee: Martial Hebert, Chair
Robert Collins
Jianbo Shi
Michael Black

This work was supported in part by the DARPA Tactical Mobile Robotics program under JPL contract 1200008 and by the Army Research Labs Robotics Collaborative Technology Alliance Contract DAAD 19-01-2-0012.

Keywords: structure from motion, visual tracking, visual navigation, visual servoing

Abstract

Robust tracking and structure from motion (SFM) are fundamental problems in computer vision that have important applications for robot visual navigation and other computer vision tasks. Although the geometry of the SFM problem is well understood and effective optimization algorithms have been proposed, SFM is still difficult to apply in practice. The reason is twofold. First, finding correspondences, or "data association", is still a challenging problem in practice. For visual navigation tasks, the correspondences are usually found by tracking, which often assumes constancy in feature appearance and smoothness in camera motion so that the search space for correspondences is much reduced. Therefore tracking itself is intrinsically difficult under degenerate conditions, such as occlusions, or abrupt camera motion which violates the assumptions for tracking to start with. Second, the result of SFM is often observed to be extremely sensitive to the error in correspondences, which is often caused by the failure of the tracking.

This thesis aims to tackle both problems simultaneously. We attribute the difficulty of applying SFM in practice to the failure of tracking algorithms under those degenerate conditions often seen in the uncontrolled environments. I propose to integrate the SFM with tracking so that the tracking algorithms can have the structure information recovered from SFM together with the image data so that it can explicitly reason about occlusions, ambiguous scenes and abrupt camera motion. Therefore it becomes more robust against those degenerate conditions by properly detecting them. In addition, the SFM results become more reliable by reducing error in the correspondences found the tracking algorithms. I aim to achieve this in a probabilistically sound way. Representing the uncertainty is one of the most important issues for a probabilistic framework can be applied. I propose a sampling method to capture the uncertainty in both tracking and SFM. The uncertainty is naturally represented with sample sets. A probabilistic filtering algorithm is developed to propagate the uncertainty through time. With the sample-based representation, our system can capture the uncertainty in both tracking and SFM under degenerate conditions, therefore exhibiting improved robustness by taking proper measures against them, which are usually difficult for the traditional approaches to achieve. We believe that with this sampling-based probabilistic framework, we are one step closer to a SFM system that can perform reliably in real robot navigation tasks.

Acknowledgments

The first thank-you goes to my advisor, Martial Hebert, for years of consistent support, for having confidence in my abilities, and for countless insights. It has truly been a privilege to work with such an effective and nice person.

I am deeply grateful to the members of my thesis committee, Bob Collins, Jianbo Shi, and Michael Black, for their time and effort put into my thesis research and thesis writing. I owe a special thanks to Professor Black, whose detailed comments and useful references help me better understand my own work and improve the overall clarity of the thesis.

Many people have contributed to my technical development at CMU. I am particularly grateful to guys in the misc-reading-group, for the insightful discussions and presentations. The discussions with Frank Dellaert, Daniel Morris, Mei Han, Matt Deans, Dennis Strelow, Qifa Ke and Alvaro Sato have been helpful during different stages of my thesis research. David LaRose has been an excellent officemate through the last year providing me with both friendship and Linux help. Also, the friendship from Mei Han, Yanghai Tsin and Huadong Wu has made the life in CMU more pleasant. In addition, I would like to thank the people who keep the computers running. Rob MacLanchlan is the goto guy in the TMR project, which directly leads to my thesis topic, Tim Doebler fixes everything related to computers, and Nicolas Vandapel helped me with the data collection.

Finally, it is important to recognize the people who, although not direct contributors to my work, have shaped my life so profoundly that it shows in everything I do. Special thanks to Yan Lu, my wife, without whom the thesis would not be possible, David Chang, now three year old, whose intelligence always surprises me and, my parents, for everything you did for me.

Contents

List of Figures

Chapter 1

Introduction

1.1 Visual Navigation

There is little doubt that cameras are crucial sensors for robot navigation. Images carry a lot of information about the outside world. The bottleneck is: we still do not understand enough to extract the useful information in an efficient and *reliable* way.

There is equally little doubt that a lot of successful vision applications have been developed for various robotics systems. Although a general purpose vision system is still well beyond the reach of the current state of art, many existing vision systems have achieved their successes by focusing on specified functionality. For example, stereo systems are used to generate the depth map which can guide the robot to avoid the obstacles and navigate for long distances [58]. Monocular vision systems are also successfully used for target tracking and visual servoing for robot navigation applications [37] [43]. It can be seen that for robot navigation purpose, one essential problem is to estimate the robot motion and keep tracking landmarks [65] [37]. Depending on the specified functionality, every system relies on certain assumptions or prior knowledge. For example, a stereo system usually assumes that the pixels corresponding to the same scene point should have similar brightness values; similarly, a feature tracking algorithm usually assumes that the feature appearances on consecutive frames are similar and they are located very close to each other. The systems would fail or behave unpredictably if the prior assumptions are violated. For real world applications, it is always desirable to have a system that relies on less strict assumptions, i.e., works under challenging situations.

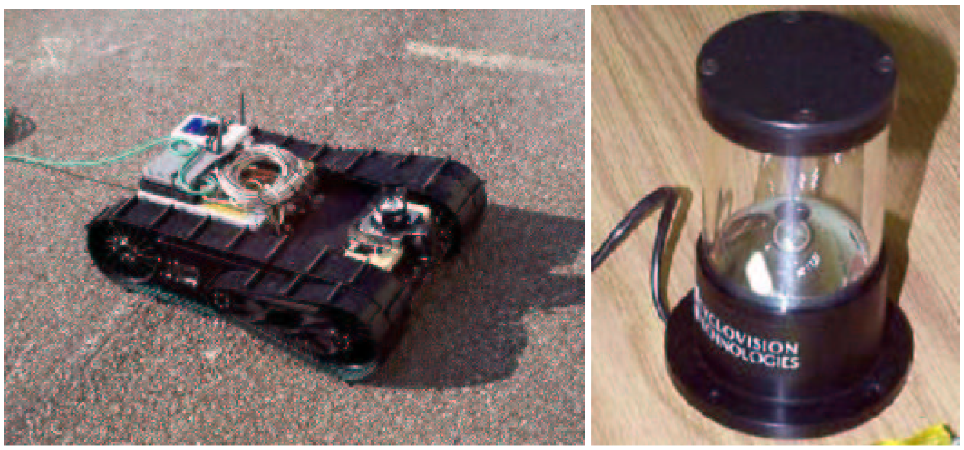
This dissertation addresses the problem of robust tracking of features (or landmarks) by embedding geometric reconstruction. In particular, how to apply the traditionally known *unreliable* geometric reconstruction in a reliable way, is the key question. We observe that the uncertainty representation in both tracking and geometric reconstruction is vital for the robustness of the overall system. We abandon the traditional covariance based uncertainty representation for correspondences and geometric reconstruction due to its inherent limitation for highly nonlinear problems and its restricted assumption on small Gaussian noise. We advocate the use of more expensive but accurate Monte Carlo method to capture the true underlying uncertainty. Accordingly, a sample based uncertainty representation is developed for both tracking and geometric reconstruction and a sample based filtering scheme is implemented to propagate the uncertainty through time. We demonstrate that our method provides reliable estimation results in real navigation tasks even in presence of large image measurement noise and under challenging conditions such as severe occlusion and abrupt robot motion, which usually can not be handled by traditional approaches.

We start by first introducing the application background to illustrate the need for geometric estimation during robot navigation. We then identify the limitations of current geometric estimation methods when applied to real navigation scenario. Finally we propose our approach to the problem.

1.1.1 Visual servoing for robots: successes and limitations

The subject of the thesis is partly inspired by the work done in the Tactical Mobile Robot (TMR) robotics program with the objective of developing small, fast robots capable of semi-autonomous operation in unstructured urban environments. In this system, a user controls the robot by designating goals which the robot attempts to reach with minimum interaction with the user. Visual servoing is the main approach taken to achieve this goal. The unique challenge in designing this visual servoing system is that the on-board driving system must be robust to shock, vibration and rapid change of attitude which commonly occur during the operation of a lightweight robot in an unstructured terrain.

Figure 1.1 shows the main hardware components of the visual servoing system: the robot and the camera. The mobility chassis is the *Urbie* tracked platform developed at IS Robotics. Tracked articulations in the front of the robot can do continuous 360 degree rotation and enable crossing curbs, climbing stairs and scrambling over rubble. The peak driving speed is 80 cm/sec on flat ground. The high level control electronics is developed by JPL and contain two PC104 stacks, with a 166 MHz Pentium II stack for navigation control and a 233 MHz Pentium II stack for vision processes. More details about the hardware configuration can be found in [60]. The vision sensors include a forward-looking stereo camera pair and an omnidirectional camera shown in Figure 1.1(b), among various other non-vision sensors. The visual servoing system uses images from the omnidirectional camera, which was developed at Columbia University by Professor Shree Nayar and later by RemoteReality Inc. The version of the omnidirectional camera used for the results and experiments throughout this thesis is the folded mirror camera as described in [63]. It captures full field of view of its surrounding environment and therefore is an ideal sensor for goal selection and



(a) The prototype robot for TMR: urbie

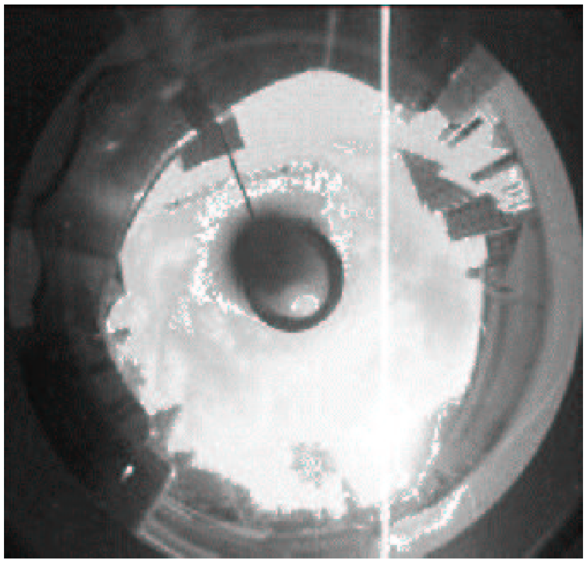
(b) The omnidirectional camera developed at Columbia University

Figure 1.1: The robot and the camera we use for the visual servoing system

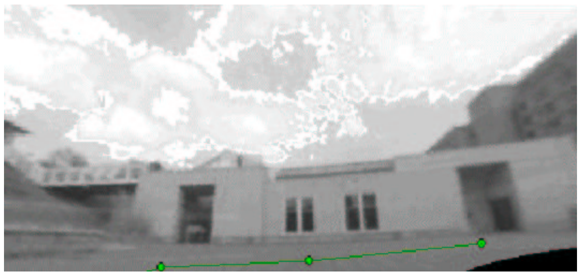
visual servoing.

In visual servoing mode, the omnidirectional image is sent to the operator from the robot platform, the operator designates the goal from the unwarped virtual perspective view, then the robot tracks the goal until it approaches the goal. An omnidirectional image taken at a test site is shown in Figure 1.2 along with its unwarped virtual perspective view. Tracking begins with a 2-D correlation search over predefined search region to accommodate large target translation. An iterative linearized affine matching procedure is followed to compensate the image plane rotation, scale and skew changes of the target caused by the robot motion. The transformation is accumulated over time and therefore the tracker can handle very large changes in the appearance of the target. More details on tracking with an omnidirectional camera can be found in [15] and [16]. As mentioned above, the challenge is to make the servoing system robust to drastic changes in robot motion and to maintain high tracking accuracy. Even though a number of techniques have been developed to cope with the substantial appearance change of the target during a long distance servoing task, targets still can be lost during a rapid robot motion or occlusion. A correlation based search scheme over enlarged window is implemented in an efficient way to recover from the tracking failures. For template sizes of $40 - 50$ pixels, the tracker runs at over 15 Hz on the 233 MHz Pentium II vision stack. More details about the algorithm and performance can be found in [40].

The servoing system has been extensively tested in the field, including a series of experiments at Ft. Sam Houston, TX, the demonstration site for the TMR project. Figure 1.3 shows two aerial

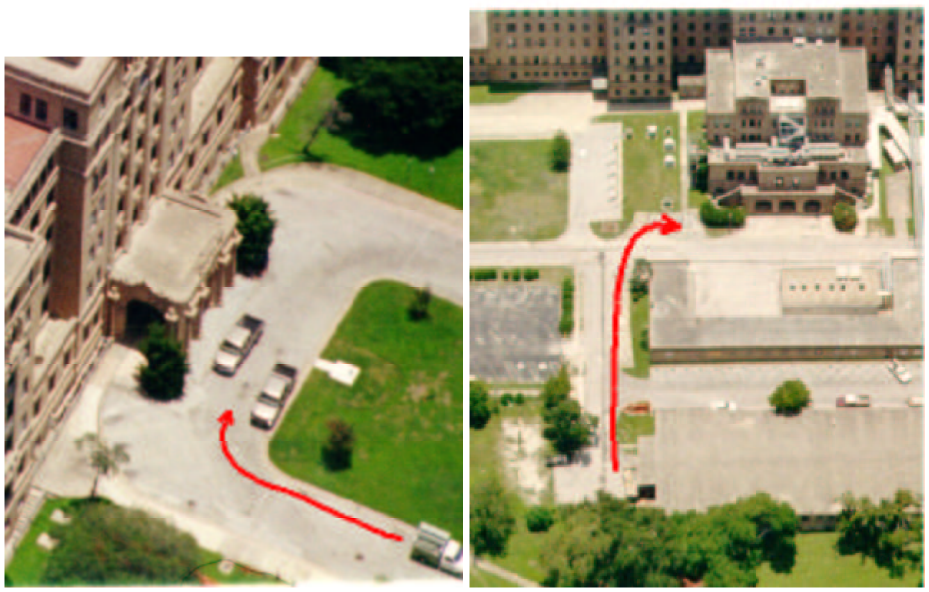


(a) The image taken by the omnicam



(b) The panoramic view unwarped from the original omnicam image

Figure 1.2: The omnidirectional image and its unwarped panoramic view



(a) Path one (b) Path two

Figure 1.3: Typical areas used for experimentation at the Ft. Sam Houston test site

Figure 1.4: Robot in action: going down a curb and driving over obstacles are typical exercises the robot has to face in a real navigation task.

views of the main building on this site. The typical paths used for testing semi-autonomous driving are also indicated in the figure. The paths were designed to simulate a mission for the robot to approach the building prior to entry and exposed the robot to all the challenges it could face in an urban environment. In Figure 1.4, several snapshots show the robot in action going down a curb and running over obstacles, which are typical when moving in an urban environment. Figure 1.5 shows two examples of tracking targets for visual servoing tasks at Ft. Sam Houston. Both targets were tracked over long distances while driving over rough terrain. As it is concluded in [60], "Mobile robot technology has reached a level of maturity at which it should soon be possible to field rugged, portable mobile robots for urban reconnaissance missions”.

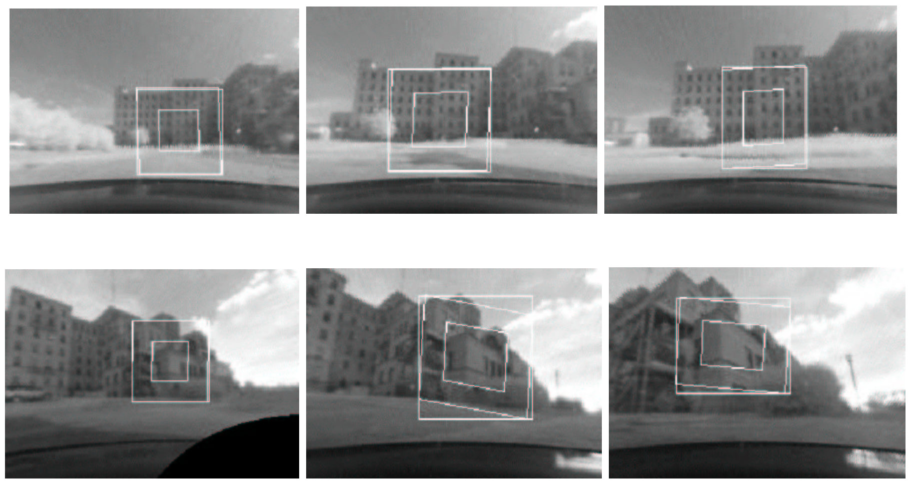


Figure 1.5: Two examples (top and down) of target tracking during visual servoing at Ft. Sam Houston test site. The inner box shows the template; the outer box shows the region of interest on the virtual image.

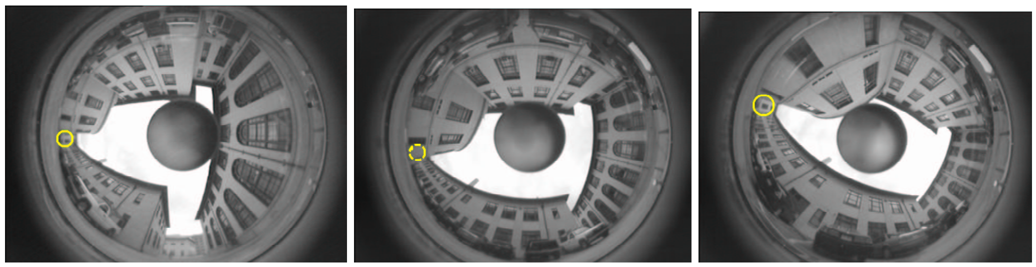
1.1.2 Enhancements: scene structure and robot motion estimation

Despite these successes, limitations of the current visual servoing system have also been identified. It is in general difficult for the robot to deal with occlusions, and the robot is prone to errors during recovery from abrupt robot motion or occlusions.

For example, Figure 1.6 shows a possible scenario in which the visual servoing could fail. In Figure 1.6(a) a driving goal is selected by the operator. After driving some distance the goal is occluded by an obstructing structure, as shown in Figure 1.6(b). Without knowing the location of the goal with respect to the robot position, the robot does not have any guidance in searching for the target place. An optimistic robot may make a risky decision to continue the driving and hope to recapture the goal. But even if the goal does appear again in the robot's field of view, it is still unlikely that the robot will capture the goal again, since by then the goal has moved far away from the place where it originally disappeared.

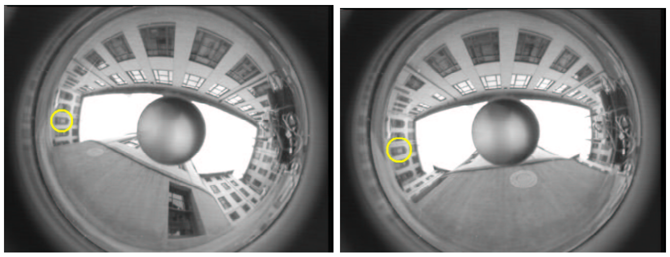
Another example is shown in Figure 1.7. The robot made an abrupt motion and, because of the limited frame rate of the camera, the target jumps out of the usual tracking region. Searching over an enlarged search region will find several possible matches for the original target and confuse the robot.

The challenges in both examples can be solved if the robot keeps an estimation of the scene structure and its own motion. How to do so in a principled way is the theme of this thesis.



(a) Goal selected (b) Goal occluded (c) Goal reappeared

Figure 1.6: Example images when the target is selected, occluded and then appeared again.



(a) Goal selected (b) Goal jumps

Figure 1.7: Example images when the target is selected, and jumps to other location due to abrupt camera rotation

1.2 Difficulties of SFM in Practice

To obtain the structure and motion information is not an easy task. Stereo can provide range estimation but its working distance is usually limited. The distances of selected targets in visual servoing missions are often beyond the range of stereo vision. The dead-reckoning data from the odometer is usually poor on fast-moving robots. A natural alternative way to estimate the scene structure and robot motion from images is to use the *structure from motion (SFM)* tools from computer vision.

Structure from motion is a well-studied field in computer vision, largely due to its promise to provide structure information from a video stream. A large number of algorithms have been published and successful applications have been demonstrated on real sequences. The progress has been summarized in a recent book by Hartley and Zisserman [39] and it is generally believed that the geometry of SFM is a well-understood problem. On the other hand, vision engineers who implement the existing SFM algorithms in real world applications have noted some serious shortcomings. We believe that the gap in performance is caused by the fact that the uncertainty of both tracking and SFM in real world applications has not been well addressed, and it is the central issue of this thesis. We start the discussion with a high level review of SFM.

There are generally two classes of approaches in performing SFM. One is called *direct method*. It refers to those approaches which either use the image pixel values directly or start by estimating the dense optical flow. Direct methods are exemplified by the work in [42, 94, 8, 7]. The other method often starts by tracking sparse features in the scene and estimates the structure only for those sparse features. It is usually less expensive and the focus of this thesis is on the second approach. In the computer vision literature, most SFM problems are formulated as a minimization problem in high dimension. Given correspondence data $\mathbf{x}$, the parameterization $\mathbf{p}$ of the scene structure and camera motion can be estimated by minimizing the criteria function $\mathcal{C}(\mathbf{x}, \mathbf{p})$ as defined in Equation 1.1.

$$\mathbf{p}^* = \underset{\mathbf{p}}{Argmin}\, \mathcal{C}(\mathbf{x}, \mathbf{p}) \tag{1.1}$$

The minimization can be carried out by either a batch method, where all the data are processed simultaneously, or a sequential method, where the current estimate is based on the old solution and the new measurement. The batch method is in principle more accurate once it converges to the global minimum, but it requires accurate initialization in high dimensions and is computationally more expensive. In practice, sequential methods are usually adopted and the batch method can be applied as a refinement if necessary. As a typical sequential method, Kalman filtering based algorithms have been repeatedly proposed and demonstrated.

Then why is robustness still a concern for SFM? In practice the relatively large noise in the tracking and various unexpected conditions such as occlusions during real navigation tasks are the main factors contributing to the brittleness of SFM.

It has been noted by researchers that the SFM algorithms are very sensitive to the noise in the measurement $\mathbf{x}$ since they attempt to recover a lost dimension. Good SFM results have been successful when the noise in $\mathbf{x}$ is relatively small. But just as asked in [79], *how much noise is too much?* It is a hard question to answer. One way to reduce the sensitivity to noise is to use techniques such as regularization to introduce smoothness priors that are commonly used for ill-posed problems in computer vision. Other methods include *active vision* which actively controls the camera motion, and sensor fusion which uses multiple sources of sensors, such as odometer sensors. Our approach here is to fully characterize the SFM uncertainty and integrate information through the time domain, namely, filtering.

Generally speaking, there are two types of noise in $\mathbf{x}$ that we need to deal with: local noise and outliers. Since all the correspondences are computed by matching the image data, which is inevitably contaminated by noise during the image formation process, the computed correspondences always deviate from their true locations. We call this type of error the local noise. The matching process could also mismatch a feature to a totally wrong location under certain circumstances and we call this type of error outliers. Both local noise and outliers cause errors in SFM and should be properly addressed.

Local noise is easier to handle, but only when it is relatively small. When the noise is assumed to be small Gaussian, it can be shown that the covariance of $\mathbf{p}^*$ can be approximated with traditional covariance propagation analytically. This covariance approximation is the cornerstone for the widely used Kalman filtering based sequential optimization method in SFM literature. Indeed, Kalman filtering is equivalent to the batch method for linear Gaussian systems and is the only optimal linear filter for nonlinear Gaussian systems. A detailed uncertainty analysis on SFM with small Gaussian noise can be found in [90]. The authors claim that the theoretic lower bound, the Cramer-Rao bound, has been reached for SFM on real sequences. However this first order approximation is only valid when the error surface is relatively smooth around $\mathbf{p}^*$, compared to the covariance of $\mathbf{x}$. In other words, if the noise in $\mathbf{x}$ is large or is quite different from a Gaussian, in principle the Kalman filtering can no longer handle it. Even though the uncertainty of SFM in case of small Gaussian noise has been thoroughly studied as in [90], and is recently incorporated into factorization [45], there is little work on cases in which the optimistic assumption on small Gaussian is broken. On the other hand, for a robot equipped with off-the-shelf hardware carrying out a navigation mission, it is rather unrealistic to assume that subpixel tracking error will be consistently achieved.

As a remedy, H_∞ filter has recently been introduced into SFM as an enhancement to Kalman filtering based methods in the case of unmodelled noise [69]. However, the proposed H_∞ filter shares the same form of a standard EKF except that the propagation of the covariance matrices is controlled by an additional attenuation factor. The covariance matrices can thus be made infinitely uncertain if necessary to guarantee overestimating the noise. Therefore H_∞ based approaches achieve robustness by being very or overly conservative and at the cost of performance of the SFM result. In other words, instead of giving the user a wrong answer, it gives the user an answer along with a very low confidence level. In addition, since it still relies on covariance matrices as primary uncertainty representation, it can not capture multi-modal distributions, which implies that the mean of the distribution is not a meaningful estimation of the true value for multi-mode distributions. As we will see later, that type of multi-mode distribution does exist in SFM during real navigation tasks.

Outliers cause unpredictable errors in SFM. A standard approach is to reject them from the correspondence set from which the SFM is computed. So far, the most popular robust algorithm on SFM is RANSAC, which rejects outliers in feature matching. The basic idea behind RANSAC is that if the correspondence set contains false matches, the SFM result will be inferior and cause large inconsistencies among all available feature matches. Therefore RANSAC rejects the outliers by finding the match which contains the largest consistent data points. There are two issues involved in applying RANSAC to SFM problem. First, in order to set the distance threshold for outlier detection, it needs prior knowledge about the noise in the correspondence data, which is generally assumed to be standard Gaussian distribution [39]. Second, it solves the robust matching for two-frame problems only and there is no information propagation through the time domain. Our method does not have these limitations, as shown later.

It is worth noting that earlier work by Black uses M-estimators to compute robust optical flow [7]. Recently the Error In Variable model is applied to SFM problem to reduce the bias in the linearization [33]. But again the noise model for the correspondence is assumed as Gaussian.

1.3 Thesis Outline

In this thesis, I explore the sampling based probabilistic approach to solve the challenges of applying SFM to real navigation tasks, such as large tracking error and degenerate conditions such as occlusions and abrupt camera motions.

A Monte Carlo filtering scheme is proposed to fuse the tracking and SFM results through time. It can be viewed as an alternative based on the recently popular sampling method to the traditional EKF based tracking and SFM approaches. The advantage of the sampling approach is its faithful representation of the uncertainty in very nonlinear systems, usually with more computational cost. While the EKF is more efficient, the conditions for it to work on tracking and SFM are usually *not* met in practice, especially in extreme driving situations. That is exactly the reason of many observations about the robustness of SFM algorithms from vision engineers in the field.

The thesis is organized as follows. First, we present the probabilistic SFM framework in Chapter 2. In Chapter 3, the uncertainty of tracking and SFM is analyzed in detail. We show that the underlying uncertainty can be quite different from a Gaussian distribution during normal navigation tasks. A sampling based technique is proposed to fully capture the uncertainty in both tracking and SFM.

In Chapter 4, we propose a Bayesian filtering approach based on the sampling representation. We show how the maximum likelihood estimate can be achieved for multiple frame SFM. We further show that the challenging situations such as occlusions and abrupt camera motions can also be handled in a principled way within this framework. From experimental data, we show that the uncertainty of tracking and SFM are indeed reduced by the fusion scheme we proposed, and the Bayesian filtering can reduce the SFM uncertainty through time.

In Chapter 5, we discuss the problem of planar patch based SFM and possible extensions of the sampling based filtering approach to this problem. In Chapter 6, experimental results are shown for this framework applied to various navigation scenarios. Image data include sequences collected from an omnidirectional camera mounted on a small mobile robot and from a hand-held video camcorder.

The contributions include:

- We advocate the sampling based uncertainty representation for SFM problems and identify drawbacks of the traditional covariance based representation for tracking and SFM in practice.
- With the help of recent advances in Monte Carlo filtering, we build a system that fully integrates tracking and SFM in a probabilistic fashion. The system performs robust tracking and SFM simultaneously and handles previously challenging situations such as severe occlusions and sudden camera motions which are commonly seen in practical robot navigation tasks.
- By making the tracking and SFM a practical tool for robot visual navigation, we open the door for much simpler and effective position based visual servoing algorithms.

Chapter 2

Probabilistic Framework

2.1 Introduction

Robot visual navigation is naturally a nonlinear dynamic process which involves both state estimation and control. The robot needs to (1) locate itself in the environment, which is the estimation process, and (2) control itself to approach certain positions. Depending on different onboard sensors, the state variables used to describe the dynamic process vary. While there are plenty of choices in term of the sensors that we can use, we are interested in using only one camera to achieve the goal, the problem known as visual servoing.

As stated earlier, the visual servoing problem includes two fundamental steps: state estimation and control. So far, most visual servoing algorithms take a 2D approach, meaning the state variables of the process are 2D locations of features and the estimation only involves 2D feature tracking [71, 91, 16, 84, 56, 95]. Due to the lack of 3D information, the initial and goal states in the control loop have to be specified with given feature configurations on image plane, which can be cumbersome. Furthermore the controller design is complicated and relies on approximations to the *image Jacobian* [43]. In addition, many of the results shown in the literature rely on tracking artificial markers or color blobs, which underline the importance and difficulty in robust 2D feature tracking in the visual servoing loop. Although as indicated in [43], the 3D position based visual servoing has simple and clean controller design, it has not been popular. An exception is [92] in which a 3D position based visual servoing algorithm is presented, but the problem is simplified by assuming known 3D feature points.

We attribute the lack of position based visual servoing to the estimation part of the problem, namely to estimate the 3D positions of the features, also known as structure from motion (SFM) in computer vision literature. SFM has largely claimed success in the vision community, and results have been shown on sequences taken in constrained lab environment or setups [1, 18, 48, 86, 85].

However, the limitation is equally apparent in the fact that no SFM system has been demonstrated to meet the challenges in uncontrolled environment, where feature appearances change substantially, occlusions commonly occur and camera motion can be jerky. That is part the reasons that we have not seen SFM applied to real visual servoing systems. Indeed as noted by Chaumette in [17], *it is well known that the (SFM) problem, as most of inverse problems, is sometimes ill-posed and sensitive to perturbations, ..., small errors in the image measurements may lead to very different results, in such cases, the control law can be completely unstable*. Evidently, applying SFM to real visual servoing tasks involves more than feature tracking plus Kalman filtering. The difficulty has been a fundamental one.

In Chapter 3 we show the uncertainty of SFM is commonly beyond the reach of traditional covariance representation, and henceforth invalidate all the approaches based on Kalman filter or EKF. Before that, in this chapter, we present the common probabilistic framework used in tracking and SFM, which subsumes both the Kalman filter type approaches and the particle filter, which is the approach we adopted. We illustrate the important novel feed-forward paths in our filtering structure, which seamlessly integrates the SFM with tracking probabilistically, and present it as an integrated particle filter.

2.2 Probabilistic Filtering

For estimating any stochastic process, filtering is critical to the reduction of state estimation uncertainty and the success of system control. Our problem of tracking and SFM is naturally a sequential process in the time domain and suitable for standard filtering.

We first define the problem and notations. We then illustrate the Kalman filtering approach with a graphical model. We then show the proposed filtering structure, which integrates the SFM with tracking by incorporating feed-forward paths.

2.2.1 Problem Description and Notations

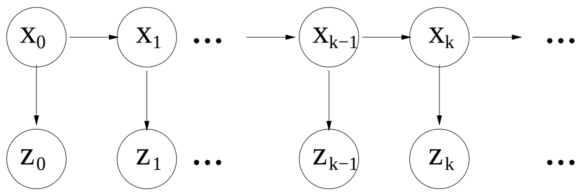
We assume that we are initially given a set of M features in a reference image I_0. We denote the position of feature j in I_0 by z_0^j, $j = 1, \ldots, M$ and the vector containing the positions of all the features by $\mathbf{z}_0$. As the robot moves, new images are acquired, which we denote by $I_1, \ldots, I_k$. We denote by $\mathbf{I}_k$ all the images from time 0 to time k. The features are located in the images using a feature tracker, which is described in Section 3.4. The location of feature j in image I_k is denoted as z_k^j and the vector of all the M feature locations is denoted by $\mathbf{z}_k$.

At time k, given $\mathbf{z}_0$ and $\mathbf{z}_k$, both the 3-D structure of the scene and the motion of the robot up to time k can be recovered. We denote by s_k^j the 3-D position of feature j reconstructed from $\mathbf{z}_k$ and by $\mathbf{s}_k$ the set of the 3-D coordinates of all M features at time k. We denote by $\mathbf{m}_k$ the robot motion recovered at time k w.r.t. the original location at time 0. Finally, we denote by $\mathbf{x}_k$ the pair $(\mathbf{s}_k, \mathbf{m}_k)$ reconstructed by SFM at time k.

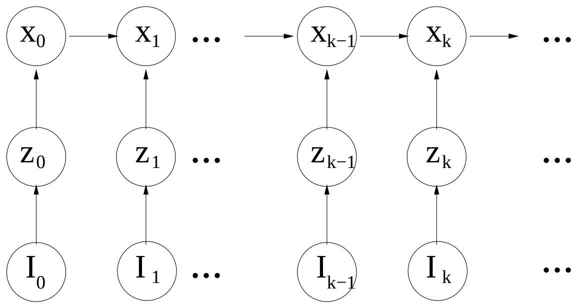
At time k, the goal is to maintain the probability $p(\mathbf{x}_k|\mathbf{I}_k)$. As stated earlier, we are interested in sequential method, where the current estimate $\mathbf{x}_k$ depends on both the previous $\mathbf{x}_{k-1}$ and the current measurement $\mathbf{z}_k$.

2.2.2 Previous Approaches and The Problems

All the sequential two-frame based SFM algorithms can be described by Figure 2.1. These include all the EKF and Kalman filter based approaches [1, 18, 48, 41, 86, 85].



(a) Causal model



(b) Inference model

Figure 2.1: Graphical models for traditional recursive SFM

There are two types of filtering schemes. The first scheme starts from some reasonably good initial estimate, and runs an (I)EKF to keep updating the estimate once new data comes in. We call it causal model as shown in Figure 2.1(a). The key character of this approach is that there is no explicit reconstruction at any time. The approaches in [1, 18, 48] belong to this type. The corresponding discrete time dynamic equations are:

$$\begin{cases} \mathbf{s}_k = \mathbf{s}_{k-1} \\ \mathbf{m}_k = \mathbf{m}_{k-1} \oplus \mathbf{v}_k{}^1 \\ \mathbf{z}_k = \pi(\mathbf{s}_k, \mathbf{m}_k) + \mathbf{n}_k \end{cases} \tag{2.1}$$

where π is the camera projection function, $\mathbf{n}$ is the measurement noise and $\mathbf{v}$ is the linear velocity if a first order motion model is used, and usually $\mathbf{v}$ can be estimated from the history of $\mathbf{m}$. In order to run the (I)EKF, the system often initializes the state to some random state with very large uncertainty and then updates the state with the new measurements according the Kalman filtering update equations. Notice that there is no explicit step in this scheme to solve the reconstruction, therefore this method heavily relies on the accuracy of the initialization. It easily converges to wrong minima if given a reasonably bad initialization. Although it is reported that, given a reasonable good initialization, the (I)EKF is able to lock on the correct mode and improve the estimate through time, the questions remain: (1) it is unclear how good the initial estimate has to be in order for the system to converge, (2) how much correspondence noise it can bear and (3) how the system performs in case of camera jittering. The authors do acknowledge in [18] that this approach does suffer from local minima and fails to converge if the correspondence noise is large or the camera motion is not smooth. Furthermore, results are mostly from simulations or smooth sequences on careful setups of lab scenes.

The second scheme is to first perform explicit reconstruction and then fuse the intermediate reconstruction results through time with a Kalman filter [41], [86] [85]. We call it an inference model as shown in Figure 2.1(b). The key difference from a causal approach is that explicit two-frame SFM is performed and filtering can be done by enforcing the rigidity constraint instead of relying on camera dynamics model. As noted in the *critique* [64], this approach suffers from the fact that if some of the intermediate reconstruction is *very inaccurate*, the fused reconstruction can grow less accurate as more images are acquired. What is not noticed in the critique is that the divergence is not due to the recursive filtering itself, but the tools used to characterize the uncertainty and filtering, namely the covariance representation and Kalman filtering. The uncertainty of SFM in practice is rarely a well-behaved Gaussian, given the large tracking errors and nonlinearity involved. Being too optimistic about the uncertainty and using simple Kalman filtering can cause unrecoverable errors once a bad estimate is fused into the final estimate. More details about the SFM uncertainty can be found in Chapter 3.

[1] $\oplus$ is used here to represent the necessary addition operation defined in the motion vector space.

2.2.3 Proposed Approach

We identify the following facts about SFM, some of them can be accounted for the limitations of various previous approaches.

1) The uncertainty of SFM is largely dependent on the tracking uncertainty. Previous Kalman filter based approaches only work when the tracking uncertainty is very small, a requirement usually *not* met in practice.

2) Given the tracking uncertainty usually achieved in practice, the uncertainty of SFM can be skewed (nonlinearity), multi-mode(ambiguous), or even wrong (local minimum), instead of a well-behaved Gaussian.

3) Robust tracking is essential to robust SFM. Robust tracking means robust against occlusion and camera jittering. Unfortunately, it is fundamentally impossible to achieve by appearance matching alone.

We explore the following key ideas to improve the robustness of both tracking and SFM.

- The first is to use a sampling method by fully characterizing the uncertainty of tracking to handle the large noise and ambiguities in correspondences and use Monte Carlo method to faithfully capture the uncertainty of SFM, especially in the case of skewed, multi-mode, and mixed (with incorrect estimate) distribution caused by tracking noise. Tracking ambiguity occurs in the situations when the feature point can be matched to multiple possible locations and usually can not be resolved from image appearance along.

- The second idea is to integrate tracking with SFM, which leads to more accurate and less ambiguous tracking results and more principled handling of occlusion.

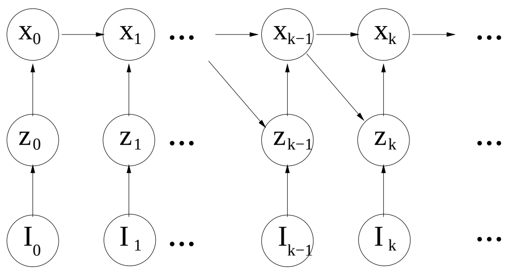


Figure 2.2: A graphical model for the proposed integrated tracking and SFM

Figure 2.2 illustrates the graphical model for our probabilistic filtering. It has a similar recursive structure as in Figure 2.1, except that extra links are added from $\mathbf{x}_k$ to $\mathbf{z}_{k+1}$ for $k = 1, ...$. These extra links imply that the correspondence $\mathbf{z}_k$ at time k not only depends on the current image I_k, but also the structure and motion estimated at time $k-1$. By adding those extra links, the tracking is integrated with SFM in a probabilistic manner and is a unique feature in our tracking and SFM system. As shown in later experimental results, the added constraints from the reconstruction can greatly help reduce the tracking ambiguity when it exists and help detect and track targets through severe occlusion.

In order to use the more general uncertainty representation than the Gaussian distribution and the more sophisticated filter structure, we need to extend the Kalman filtering to probabilistic filtering.

2.2.4 Probabilistic Filtering

Independently of any particular form of uncertainty representation and filters, the task of filtering SFM through time can be viewed as an inference problem which estimates the posterior $p(\mathbf{x}_k|\mathbf{z}_i, i = 0, ..., k)$ at time k recursively through time. As noted in [66] [25], the learning and inference from a general joint distribution in high dimension is usually computationally intractable, and probabilistic independence has to be used to simplify the problem. For the problem of tracking and SFM, the dependence among variables is natural and obvious, as shown in Figure 2.1, or in Figure 2.2, the one we propose is to integrate tracking with SFM. Given the probabilistic dependence, the probabilistic filtering task becomes an inference problem of a graphical model, in particular, a Bayesian network as shown in both Figure 2.1 and Figure 2.2. The graphical models shown in Figure 2.1 subsumes all the previous Kalman filter or (I)EKF approaches to SFM. Indeed, as noted in [77], a broad range of probabilistic models can be described with graphical models in a unified way, including Kalman filter, Hidden Markov Models, Markov Random Fields and Bayesian networks, etc.

The dependence in a graphical model is often specified with arrowed links. Once the corresponding conditional probabilities are defined, the inference can be carried out with this probabilistic model [66] [25]. It is worth noting that the inference is not always tractable. For the causal model in Figure 2.1(a), the following probabilities need to be specified:

$$\left\{ \begin{array}{c} p(\mathbf{x}_0) \\ p(\mathbf{x}_k|\mathbf{x}_{k-1}) \\ p(\mathbf{z}_k|\mathbf{x}_k) \end{array} \right. \tag{2.2}$$

The probability $p(\mathbf{x}_0)$ is the initial state uncertainty. Usually it is set to be very uncertain. $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ represents the prior knowledge of the system dynamics and $p(\mathbf{z}_k|\mathbf{x}_k)$ relates the current observation to a given state. With all three probabilities assumed to be Gaussian distribution and represented with covariance matrix, the uncertainty propagation can be efficiently carried out with matrix operations, the Kalman filtering equations.

For the inference model in Figure 2.1(b), we need to specify the following probabilities:

$$\begin{cases} p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1}) \\ p(\mathbf{z}_k|I_k) \end{cases} \tag{2.3}$$

The probability $p(\mathbf{z}_k|I_k)$ represents the measurement uncertainty in the correspondence data $\mathbf{z}_k$, and it can be obtained from the tracker, which is used to estimate $\mathbf{z}_k$ from the image I_k at time k. Section 3.4 reveals the details of uncertainty representation for $p(\mathbf{z}_k|I_k)$. The key step is to compute the probability $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$. It is usually computed from two probabilities, namely $p(\mathbf{x}_k|\mathbf{z}_k)$ and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. If both of them are assumed to be Gaussian, there exists very efficient way to compute the joint conditional probability $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$, which is the Kalman filter.

For our proposed model for integrated tracking and SFM shown in Figure 2.1(b), we need to specify the following probabilities:

$$\begin{cases} p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1}) \\ p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1}) \end{cases} \tag{2.4}$$

The integration of tracking with SFM requires the joint conditional probability $p(\mathbf{z}|I_k, \mathbf{x}_{k-1})$, in addition to $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$.

Now the question arises that given a probabilistic filter illustrated with a graphical model, which tool should we use to propagate the uncertainty in case of general distributions, in our case, sample based distributions.

Particle filtering has been the most popular approach to propagate sampling based uncertainty through time [35, 50, 51, 32, 47, 54, 14, 75, 22]. In different contexts, it has been used under names such as Monte Carlo filters, bootstrap filters and Condensation. Most of the research has concentrated on the causal model as in Figure 2.1(a). Given the probabilities defined in Equation 2.2, the goal of particle filtering is to maintain the probability $p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k)$, which is given by applying Bayes rule:

$$p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k) \propto p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k-1) \tag{2.5}$$

and $p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k-1)$ is called the prediction distribution and can be computed by

$$p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k-1) = \int_{\mathbf{x}_{k-1}} p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_i, i = 1, ..., k-1) \tag{2.6}$$

Given a sampled representation $\mathbf{x}^s_{k-1}, s = 1, ..., N$ for $p(\mathbf{x}_{k-1}|\mathbf{z}_i, i = 1, ..., k-1)$, the sampled representation $\mathbf{x}^s_{k-1}{}', s = 1, ..., N$ for $p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k-1)$ can be generated by sampling from $p(\mathbf{x}_k|\mathbf{x}^s_{k-1})$ for each sample. Finally, the sample set $\mathbf{x}^s_k, s = 1, ..., N$ for $p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k)$can be generated by factored sampling according to Equation 2.5. This iteration can be carried out through time.

So far, the particle filter has been mostly studied for solving the causal model as in Figure 2.1(a). Applying it to a more sophisticated filter structure, as shown in Figure 2.2, is not immediately clear. We now explain how to realize the particle filter by viewing it as an inference problem in Bayesian networks.

The general tool to make inference for Bayesian networks is forward sampling. For the inference models one first finds a ordering of all the nodes (variables) on the model, so that all parents of a node precede it in the ordering. The ordering is quite obvious for the models shown above. Once evidences (I_k) come in, the sampling can be carried out according to the conditional probabilities defined beforehand. In our case, the two conditional probabilities defined in Equation 2.4 can be arbitrary distributions, and a *factored sampling* based algorithm is used to compute both joint conditional probabilities. Details are described in Section 4.3.3. Under certain assumptions, we can also prove that $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{I}_k)$, indicating that current uncertainty of state estimation, represented by probability $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$, integrates the information through all the images $\mathbf{I}_k$ probabilistically. See Appendix B for details.

2.2.5 Related Work

Using a sampling method to represent the state uncertainty in a nonlinear dynamic system has received a lot attention in statistics, robotics and computer vision [35, 50, 51, 32, 47, 54, 14, 75, 22]. Particle filter as the primary tool for filtering sample based uncertainty has been extensively studied in different fields. It is closely related to the sequential Monte Carlo methods, and many theoretical and experimental results have been summarized in a recent book [24].

Recently, sampling techniques have been applied to the topic of structure from motion [30, 23, 21, 68]. We compare our approach with these approaches in more details in Section 3.7.

Here we compare the philosophical difference between the causal model and inference model for the problem of tracking and SFM, and discuss its impact on the performance of the system.

The causal model takes a hypothesize-and-verify type of approach. It generates hypotheses $p(\mathbf{x}_k|\mathbf{z}_i, i = 1, ..., k-1)$ for the current system state, usually based on current estimation and system dynamics, and then verifies the hypotheses with new observations. Obviously the quality of the hypotheses greatly influences the overall performance. (It at least has to overlap with the true system state, so that they can be verified). In the case of particle filtering, the quality of the prediction decides the number of particles needed in order for the particle filter to function properly. In the case of poor prediction, usually large number of particles are needed to avoid sample impoverishment, an effect which usually causes failures of the system [9] [49]. In a recent paper [31], KL divergence between the proposing distribution and the true distribution has been proposed to measure the effectiveness of the hypotheses and change the sample size adaptively. If the initial guess of the state is reasonably good, then the quality of the hypotheses largely depends on the accuracy of system dynamics. For a problem with relatively small number of states, such as 2D robot localization [22] [32], the state space can be covered with a limited number of random samples, therefore it does not require accurate dynamics model. In the context of tracking and SFM, random sampling as in 2D robot localization is prohibitively costly, since robot localization involves only the 2D location of the robot, while SFM has a solution space of $5 + 3 * n$, where n is the number of points in the scene. For example, consider only randomly sampling the robot motion with 5 angles in $[0, 2\pi)$, it requires $1.4e8$ samples to guarantee to have one sample with 0.1 accuracy (with 0.99 probability), let alone to sample the whole structure. Therefore, to make the hypothesize-and-verify approach work, prior knowledge of the dynamics has to be incorporated to limit the sampling space. [68] is an example of this approach. By assuming camera smoothness and removing structure from the state, SFM results are shown on standard sequences. But what is not mentioned in [68] is the sampling size and the effect of non-smooth camera motion. A more detailed comparison is provided in Section 3.7.

We do not *hypothesize* the state, we *estimate* it directly, with the inference model. Prior SFM research provides us reliable tools to perform the estimation directly. Given the measurement $\mathbf{z}_k$, an estimation of $\mathbf{x}_k$ can be obtained from existing SFM algorithms, upon which $p(\mathbf{x}_k|\mathbf{z}_k)$ can be obtained. By recognizing the strong non-Gaussian effects in $p(\mathbf{x}_k|\mathbf{z}_k)$, caused by nonlinearity, local minimum and ambiguity, we use a Monte Carlo sampling method to obtain it, therefore it faithfully captures the true underlying SFM distribution. In other words, if the SFM results are correct, the KL divergence between the estimated $p(\mathbf{x}_k|\mathbf{z}_k)$ and the true uncertainty should be minimal, and therefore the samples are always maximally effective according to KL divergence based measurement [31]. In contrast, in the hypothesize-and-test scheme, $p(\mathbf{x}_k|\mathbf{z}_k)$ is achieved by weighting and resampling the proposal distribution, which is subject to sample impoverishment if the proposal distribution differs from the underlying true distribution a lot. Indeed, the most efficient way to approach the underlying distribution is to directly estimate it.

Figure 2.3 illustrates the difference between sampling by solving SFM directly and sampling by hypotheses based on system dynamics. By solving SFM, the samples directly reside on the surface of the SFM solution space, while in the hypothesize-and-verify type approach, the hypothesized samples occupy the whole high dimensional space and then only those samples which *happen* to fall on the SFM surface are *selected* according to the SFM constraint.

To summarize, the salient features in our proposed approach include the following. First, all the probabilities are represented by samples, which is a versatile representation for arbitrary distributions. It is very important for the uncertainty representation of SFM results, which can be quite arbitrary (even erroneous) in presence of large tracking noise, tracking ambiguity and camera jittering, etc. Uncertainty representation is the topic of Chapter 3. The proposed sampling representation

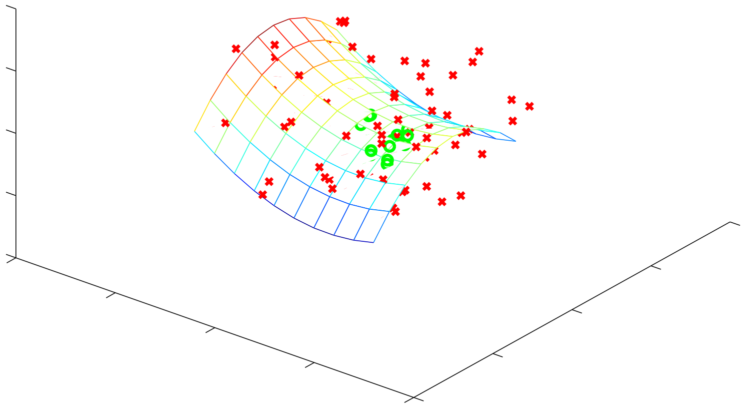


Figure 2.3: An illustration of the samples from solving SFM (green circles) and from hypotheses (red crosses)

is compared with the traditional covariance based methods.

Second, tracking and SFM are integrated based on the sampled representation. It can be best illustrated in Figure 2.2 that the correspondences $\mathbf{z_k}$ depend on both the current image I_k and the previous reconstruction $\mathbf{x}_{k-1}$, unlike the traditional approach, where $\mathbf{z}_k$ is solely decided by the current image I_k. Integrated tracking is much more robust against occlusion, camera jittering and ambiguous patterns. Chapter 4 is devoted to how to maintain the joint conditional probabilities through time and how the ambiguity and occlusion are handled probabilistically within this framework.

The following table puts our method in the context of filtering method on SFM, along with a sample of relevant references.

	Kalman Filter	Particle Filter
Hypothesize-and-test	[1] [18]	[68]
Direct SFM	[41] [86] [85]	Ours

In Chapter 4, we explain in detail how the particle filtering can be applied to SFM tasks, especially the integration of tracking and SFM, which we believe results in more accurate and robust tracking.

Chapter 3

Sample Based Uncertainty Representation for Tracking and SFM

3.1 Introduction

In Chapter 2 we outlined the probabilistic framework we propose to perform sequential tracking and SFM. The novel part of our approach is to explicitly represent the uncertainty of tracking (measurement) and SFM (state) with a sampling method. In this chapter, we explain in detail how we sample the tracking and SFM uncertainty. To be consistent with our probabilistic framework, they are defined as follows, respectively:

- $p(\mathbf{z}_k|I_k)$: tracking uncertainty, which is the possible locations of the specified features, extracted by a matching process, or, a tracker, given the current image I_k. It is explained in more details in Section 3.5.1;

- $p(\mathbf{x}_k|\mathbf{z}_k)$: SFM uncertainty, which represents the possible reconstructions by the SFM process, given the possible locations (correspondences) of all the features $\mathbf{z}_k$. It is explained in Section 3.5.2.

These two probabilities are the building blocks for the filtering process explained in Chapter 4.

3.2 Tracking

As a fundamental tool in computer vision research, tracking has been an important topic in computer vision research, ranging from corner point tracking [52, 73], region tracking [4, 38, 6, 20, 72, 2], to recent human body tracking [11, 75].

Feature tracking is a prerequisite for feature based SFM algorithms. One popular way to track feature points is to first detect corner points in adjacent image frames and then try to match them according to the similarity and proximity [27] [39]. This approach works with images with high quality and slow camera motion. In the case of large camera motion, the matching process is more expensive and less stable. Our experience in the field shows that it is always desirable to track regions, instead of points. By using regions, the tracking solution is stable in the presence of large camera motion. A nice summary on very efficient region based tracking algorithms can be found in [2], which is the basis of the tracking algorithm we use throughout this thesis.

3.3 Previous Uncertainty Representation for Tracking and SFM Based on Small Gaussian Assumption

In this section, we examine the covariance based uncertainty representation for SFM. Recall that most SFM problems can be formulated as optimization problems in high dimensions. Equation 3.1 revisits Equation 1.1, where $\mathbf{x}$ is the structure and motion parameters and $\mathbf{z}$ the corrupted correspondences data.

$$\mathbf{x}^* = \underset{\mathbf{x}}{Argmin}\, \mathcal{C}(\mathbf{x}, \mathbf{z}) \tag{3.1}$$

The criterion function $\mathcal{C}$ usually takes the form of summed square error, and nonlinear least square optimization tools are used to solve it [39]. When the global minima is reached and with the assumption that the noise in the correspondence data $\mathbf{z}$ is small Gaussian, it can be shown that the covariance of $\mathbf{x}^*$ can be approximated with a covariance propagation method analytically [96]. The covariance of $\mathbf{x}^*$ is denoted as $\Lambda_{\mathbf{x}}$. Given the covariance of $\mathbf{z}$ denoted as $\Lambda_{\mathbf{z}}$, $\Lambda_{\mathbf{x}}$ can be computed as:

$$\Lambda_{\mathbf{x}} = \mathbf{H}^{-1} \frac{\partial \Phi}{\partial \mathbf{z}} \Lambda_{\mathbf{z}} \left(\frac{\partial \Phi}{\partial \mathbf{z}} \right)^T \mathbf{H}^{-T} \tag{3.2}$$

where $\Phi = \left(\frac{\partial \mathcal{C}}{\partial \mathbf{x}}\right)^T$ is the Jacobian and $\mathbf{H} = \frac{\partial \Phi}{\partial \mathbf{x}}$ is the Hessian.

This covariance propagation is the base for all the Kalman filtering based methods [1] [10] [13] [59] [61] [78]. The uncertainty of SFM with small Gaussian noise in correspondence data has been well addressed in the literature. Bundle adjustment is often the most accurate method and Kalman filtering based method is the most popular if not the only vehicle for sequential processing.

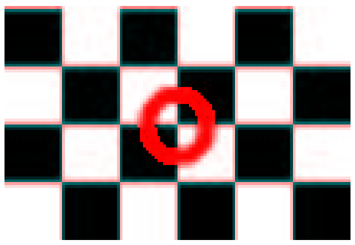
But the small Gaussian assumption is *not* always met in reality, especially for a robot driving in the field. In the next section, we closely examine the tracking uncertainty.

3.4 Tracking Uncertainty

One of the actual challenges in applying SFM to real applications is to find reliable correspondence, namely the tracking problem. An *ideal tracker* should find correspondence of a feature point through all image frames in which it is visible, with Gaussian uncertainty. To satisfy the *Gaussian assumption* about the correspondences noise that all the SFM algorithms impose, only the tracking result from an *ideal tracker* can be used as input to SFM algorithms. An *ideal tracker* should also detect occlusion whenever it happens.

So far, most of the tracking algorithms depend on the target appearance alone. For example, Lucas-Kanade tracker [52] is commonly used in the point based SFM research. It tracks a feature point by finding the nearest image patch which has a similar appearance to the original feature template. We will show that it is *not* an *ideal tracker* under certain circumstances. In fact, an *ideal tracker* does not exist in the cases described below when only the image appearance is used in the matching.

(1) Dramatic camera motion can move the feature being tracked far away from the original location and cause ambiguity, that is, from the appearance alone, the tracker can not decide a single location for the feature being tracked. As an example, Figure 3.1(a) shows a corner feature selected to be tracked. If the camera motion is large enough, there could be multiple matches to the feature being tracked, as shown in Figure 3.1(b). The usual tracker often greedily searches for the closest match and returns only one of them. In this case, directly using the result from a usual tracker in a SFM procedure is likely to produce an error.


(a) Corner feature selected with red circle

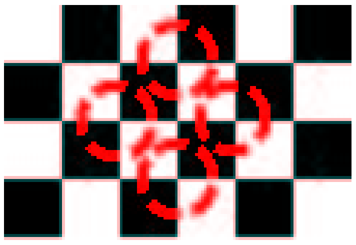
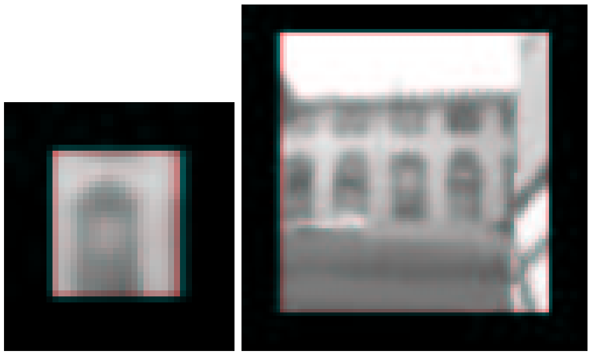

(b) Four possible matches (in dashed circles) when the camera motion is large

Figure 3.1: Multiple matches when the camera motion is large

An example from a real tracking sequence is shown in Figure 3.2. The window being tracked is similar to the windows around it. When a large camera motion occurs, it is difficult to track it from image appearance alone. The usual tracker assumes small motion in the image, therefore it often fails in case of large camera motion between frames and search over extended region is needed.

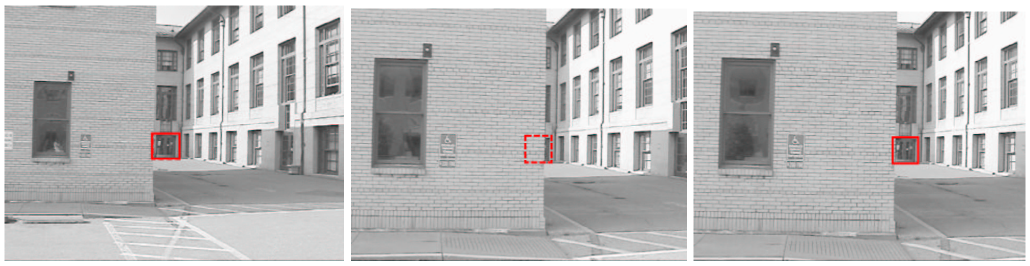


(a) Target template to be matched

(b) Multiple similar objects nearby

Figure 3.2: Multiple matches in a confusing scene

(2) Partial and full occlusion can cause Lucas-Kanade type tracker to lose track. The commonly used implementation of Lucas-Kanade tracker (LKT) uses simple heuristics to detect occlusion based on the match error. Figure 3.3 shows a target that is first occluded and then appears in a later frame which is a challenge case for the usual appearance based trackers.



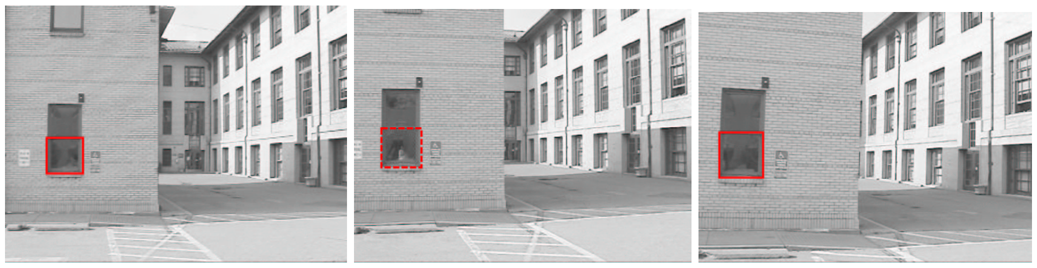
(a) The original target

(b) Target occluded

(c) Target appears again

Figure 3.3: A sequence where a target is first occluded and then appears in a later frame. An appearance based tracker is unable to track through the occlusion.

(3) Illumination change or non-lambertian surface can cause the appearance based tracking to fail unpredictably, since the appearance based tracking relies on the assumption of constant illumination and lambertian surface property. Figure 3.4 shows such an example, in which the appearance of the target changes due to the specular reflection.



(a) The original appearance of the target

(b) Target changes its appearance due to specular effect

(c) Target restores its original appearance later

Figure 3.4: A sequence where a window is being tracked. In most frames the specular effect is not obvious, but at certain frames, the specular effect is strongly observed and therefore violates the appearance constancy assumption for the usual template matching trackers.

More sophisticated appearance-based trackers have been developed to cope with those problems. For example, use robust statistics to reduce of effect of outliers [7], use hierarchical matching and pyramids to increase the tolerance of target motion [4], use robust matching to cope with partial occlusion [38], and specular effect has been studied extensively in the context of stereo matching [12] [5] [81]. Most of them need special algorithms. In our tracking and SFM problem, the focus is not to analyze those difficult cases carefully, instead it is more important to detect those degenerated cases and prevent the naive appearance based tracker to make unrecoverable mistakes.

In a controlled environment, some factors can help the appearance based trackers to approximate the performance of an *ideal tracker*. For example, the camera motion and setup of the objects in the scene may be constrained. But for a robot moving in uncontrolled environments, this is not feasible. Dramatic motion and occlusion are constantly expected. Illumination changes and non-lambertian surfaces are not unusual in practice as well. Therefore, the tracker generally can not be assumed to be *ideal* and its true uncertainty can not be simply assumed to be Gaussian.

The question arises as to how we can model the uncertainty of an appearance based tracker and the uncertainty of SFM based on the tracking result. Given the fact that the actual tracking uncertainty depends on the property of the selected target, it is generally unwise to assume all the features take the same tracking uncertainty. For example, well-focused images provide better feature localization than out-of-focus ones, feature located on a horizontal line should have larger uncertainty along the direction of the edge, and it is much easier to track a person in an open filed than on the crowded street, etc. All these facts imply that the tracking uncertainty has to be examined for each feature at each time frame. In order to capture all the possible tracking uncertainty, we propose a sampling scheme which is explained in detail in next section.

3.5 Sampling-Based Uncertainty Representation

There are two standard ways to represent an uncertainty distribution: parametric distribution function and non-parametric sample based representation. The parametric function is easy to manipulate mathematically and is suitable for relatively simple and known distributions. The sampled representation is an extremely versatile and efficient way to represent an unknown distribution. With the sampled representation, we avoid the usually problematic parameter fitting step that is unavoidable if a parametric distribution function is used instead. Sampling is the method used in this thesis to represent both the tracking and SFM uncertainty.

3.5.1 Tracking Uncertainty

In this section we define the probability $p(\mathbf{z}|I)$. Strictly speaking, the notation for the tracking uncertainty should be $p(\mathbf{z}|T_0, T)$, where T_0 is the initial template and T the current one. We assume there is an affine transformation between T and T_0 and only the translation component $\mathbf{z}$ is used for SFM purpose. Throughout the thesis we use I to represent the current template T, and omit the initial template T_0 since it is a constant.

Recall that our task is robust tracking and SFM on a mobile robot driving outdoors. The camera motion is usually non-smooth and the tracker has to maximize its capability to cope with the large motion and appearance change of the target being tracked. For this reason, the usual corner matching does not work reliably enough. Our experiments show that tracking regions is generally more reliable than tracking points. The actual tracking algorithm we use is developed based on the deformable template matching [38]. The long history of research on template matching and image alignment algorithms can be found in [2]. The essence of all the tracking algorithms is based on minimizing the Sum of Square Difference (SSD) between the current image patch and the original feature template. The deformation of the current image patch from the original feature template is specified by a family of parametric transformations, namely affine or homographies. By explicitly compensating the deformation the tracker can match the feature with current image patch even in presence of large deformation, such as scale change, rotation and skewing. It is crucial for the robustness of the tracker since we need to track features for long distance during navigation tasks and the appearances of the features usually deform substantially. After compensating the large deformation, the feature location $\mathbf{z}$ on the current image is computed by minimizing the SSD between the feature template and the unwarped current image patch. To simplify notations, we denote the difference between the reference template and the unwarped image patch at location $\mathbf{z}$ by $SSD(\mathbf{z})$. It is worth noting that both the target template and current image patch are normalized to the range of $[0, 1]$.

Given the surface $SSD(\mathbf{z})$ for each possible correspondence $\mathbf{z}$, there exists standard method to convert it into a probability for $\mathbf{z}$, which is defined as $p(\mathbf{z}|I)$ in this thesis. With Bayesian rule, it can be shown that

$$p(\mathbf{z}|I) \propto p(\mathbf{z})p(I|\mathbf{z})$$

where $p(\mathbf{z})$ is a prior for $\mathbf{z}$ and $p(\mathbf{z})p(I|\mathbf{z})$ a likelihood. The prior assumed here is simply a uniform distribution within the search region of the template matching process and the size of search region is specified according to the expected target motion, namely, the larger the target motion we want to handle, the larger the search region. The likelihood function can be directly related to the SSD surface. Following the work in [89, 76], the likelihood can be computed as in Equation 3.3 assuming Gaussian noise in the temporal derivative:

$$p(I|\mathbf{z}) \propto exp(-SSD(\mathbf{z})/2\sigma^2) \tag{3.3}$$

where σ is the expected variance of the noise in the temporal derivative. This parameter effectively controls the shape the final probability function. If σ is chosen to be small, then the corresponding probability function is concentrated around the minimum of the SSD surface and it is unlikely to deviate from it, while large σ allows the probability function to spread out around the minimum. In our implementation, the σ^2 is chosen to be 0.05. As it will be seen later, samples are generated directly from the distribution $p(\mathbf{z}|I)$ as the representation for the tracking uncertainty. With sampling method, the normalization constant has no effect on the final sample set, therefore we can directly use the function value $exp(-SSD(\mathbf{z})/2\sigma^2)$ as the weight for a sample at location $\mathbf{z}$. Figure 3.5 shows the weighting for $p(\mathbf{z}|I)$ given different values of $SSD(\mathbf{z})$. Note that the horizontal axis is actually the normalized correlation value, since the value of SSD is a function of the normalized correlation value for normalized image patches where $SSD = 2 * (1 - Normalized_Correlation)$. As we can see, according to Figure 3.5, a location with normalized correlation value of 0.95 is approximately 8 times more likely to be the true location than locations with correlation values of 0.8.

Once Equation 3.3 is defined, sampling from this 2D distribution is straightforward. To represent the uncertainty in the tracking, a set of N sampled locations is drawn according to the distribution of Equation 3.3. Given a set of M features, we denote by Z a set of samples for all the features drawn from the corresponding distribution, with $\mathbf{z}_{(i)}$ denoting the i_{th} sample containing all the feature locations, and $z^j_{(i)}$ denoting the position of the j_{th} feature in the i_{th} sample. More precisely, $\mathbf{z}_{(i)} = (z^1_{(i)}, \ldots, z^M_{(i)})i = 1, \ldots, N$ is drawn from the combined distribution

$$p(z^1, ..., z^M) = \prod_{j=1}^{j=M} p(z^j)$$

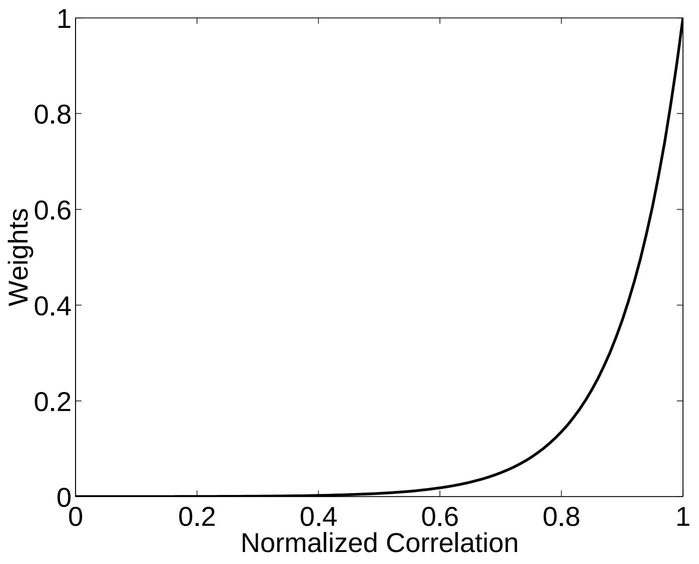


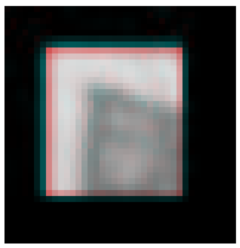
Figure 3.5: The weighting function for the normalized correlation values

where $p(z^j), j = 1, \ldots, M$ is the tracking distribution of Equation 3.3 for the j_{th} feature. We assume the noise in z^j is caused by independent Gaussian noise in each pixel, therefore $p(z^j)$ depends only on the j_{th} feature, therefore $z^j, j = 1, \ldots, M$ are independent.

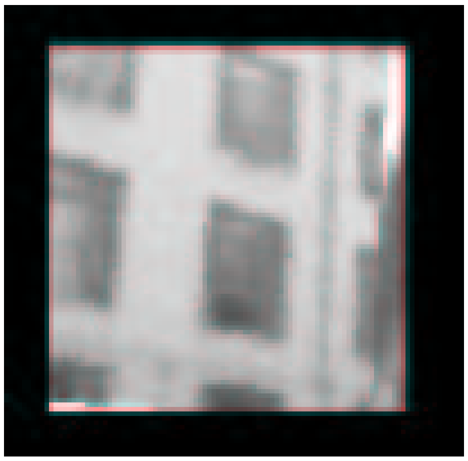
The following algorithm is used to generate $Z = (\mathbf{z}_{(1)}, ..., \mathbf{z}_{(N)})$. We now give several concrete examples of sampling the uncertainty of feature tracking. Figure 3.6 shows a typical feature with the matched image region. It also shows the SSD surface and the samples which are sampled from the distribution described by Equation 3.3.

```
for i=1:N {
    for j=1:M {
        sample the distribution Equation 3.3 to get $\mathbf{z}^j_{(i)}$
    }
    let $\mathbf{z}_{(i)} = (\mathbf{z}^1_{(i)}, ..., \mathbf{z}^M_{(i)})$
}
return $Z = (\mathbf{z}_{(1)}, \ldots, \mathbf{z}_{(N)})$
```

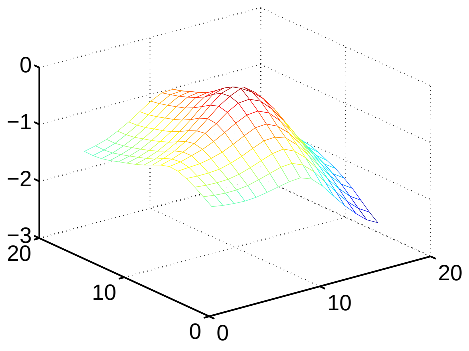
Figure 3.7 shows a situation in which the location of the feature is ambiguous. The feature being tracked, the door, is similar in the appearance to the other doors nearby. Searching the door in an extended area will return several possible locations for the match. In such cases, the uncertainty



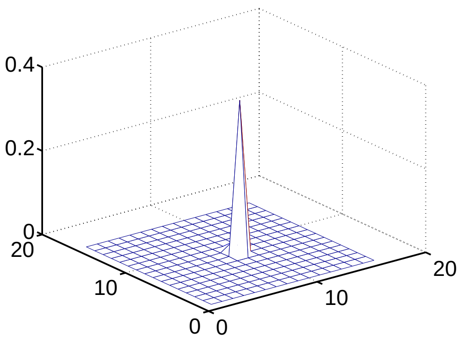
(a) one selected feature (corner point)



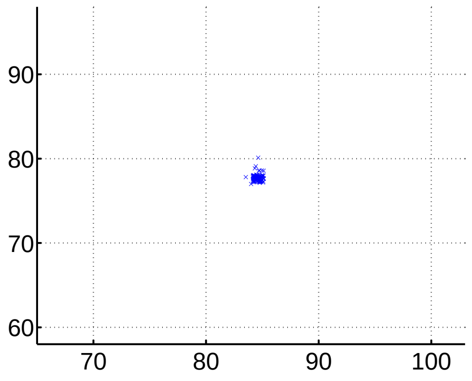
(b) the search region



(c) SSD surface (The negative of SSD is shown here to emphasize the peak)

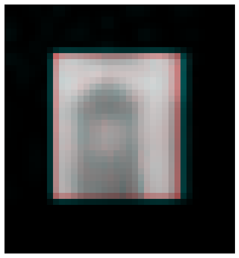


(d) density distribution converted from the SSD surface by Equation 3.3



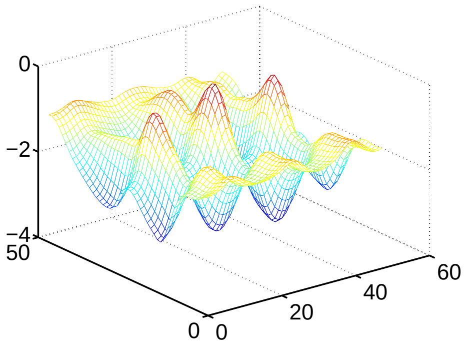
(e) samples from the density

Figure 3.6: Sampled representation of tracking uncertainty

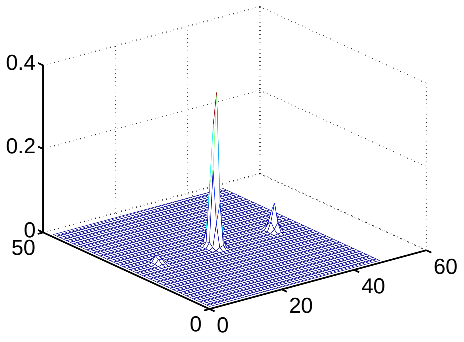


(a) selected feature

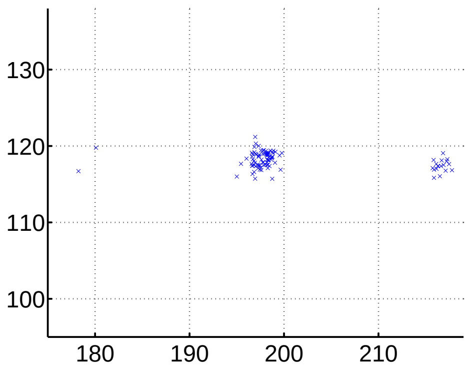
(b) the search region



(c) SSD surface (The negative of the SSD is shown here again)



(d) density distribution converted according to Equation 3.3



(e) samples from the density

Figure 3.7: Sampled representation of tracking uncertainty in case of ambiguity

distribution 3.3 is multi-modal and cannot be represented by a Gaussian distribution. With the sampling method, the multi-modal distribution is captured successfully.

Given the sampled uncertainty representation for tracking, our next task is to characterize the uncertainty of SFM.

3.5.2 SFM Uncertainty

In Section 2.2.2 we explain the traditional covariance based uncertainty representations and their limitations in robot visual navigation tasks. We propose to use sample based uncertainty representation instead. In the following we explicitly explain how to generate sample set for $p(\mathbf{x}|\mathbf{z})$, namely given the sample set for correspondence uncertainty how to generate sample sets representing the uncertainty of the SFM results.

3.6 Monte Carlo Method

The Monte Carlo method provides approximate solutions to a variety of mathematical problems by performing statistical sampling experiments on a computer [29]. Monte Carlo simulation is the most general method to characterize the statistical property of an estimator in a very precise way. Suppose we have a model $\mathbf{a}_{true} = f(\mathcal{D}_{true})$, where $\mathbf{a}_{true}$ is the true parameter set that we are interested in and it can only be computed from the data $\mathcal{D}_{true}$. In practice, the observed data $\mathcal{D}_0$ is always a noise corrupted version of $\mathcal{D}_{true}$, and $\mathbf{a}_0$ is normally obtained by fitting the data to the model through an optimization procedure.

The question is: what is the uncertainty of the estimate $\mathbf{a}_0$? If the uncertainty of the noise in the data $\mathcal{D}_0$ is known and the model $f(\mathcal{D})$ has a relatively simple expression, the uncertainty of $\mathbf{a}_0$ can be approximated analytically with standard covariance propagation methods. However, if the distribution of the data noise is not known beforehand, or the model is sufficiently complex, Monte Carlo method is then a general method for precise uncertainty estimation of $\mathbf{a}_0$. Figure 3.8 illustrates the procedure of a Monte Carlo simulation.

To study the distribution of fitted parameters $\mathbf{a}_0$, we first generate or *simulate* our own sets of data $\mathcal{D}^s_1, \mathcal{D}^s_2, ..., \mathcal{D}^s_N$ according to the knowledge of the uncertainty in the measurement process that generates $\mathcal{D}_0$. By construction these simulated data sets represent the uncertainty of $\mathcal{D}_0$. Next, for each $\mathcal{D}^s_j, j = 1, ..., N$, perform exactly the same optimization procedure to get $\mathbf{a}^s_j, j = 1, ..., N$. The Monte Carlo method generates a set $\mathbf{a}^s_j, j = 1, ..., N$ that is a sampled version of $p(\mathbf{a}_0|\mathcal{D})$.

Recently *bootstrap* techniques have been popular within the computer vision community as a tool for performance evaluation and uncertainty estimation [19, 57]. In fact, bootstrap is a *quick-and-dirty* Monte Carlo method. Instead of generating the synthetic data sets according to the under-

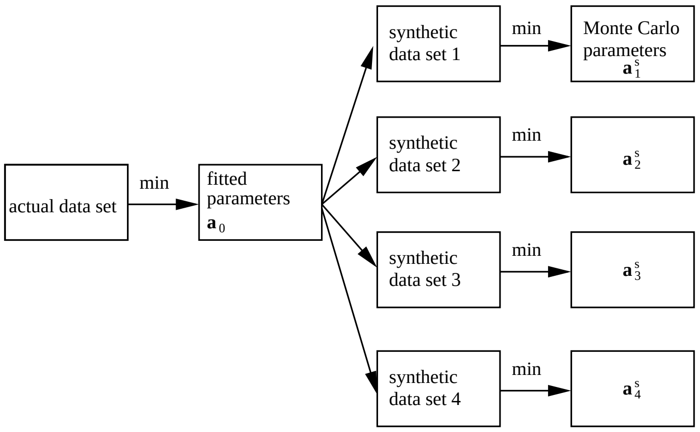


Figure 3.8: Monte Carlo simulation of an experiment, figure adapted from [67]

lying measurement process, bootstrap forms its synthetic data sets by sampling with replacement of the original data, based on the assumption that the noise in the measurement is *independent and identically distributed* (or i.i.d.).

For the problem of SFM, the correspondence $\mathbf{z}$ is the corrupted measurement data $\mathcal{D}$ and the structure and motion $\mathbf{x}$ is the parameter to estimate. As in Equation 3.1, once the minima is reached, we often can assume there exists an implicit function satisfying

$$\mathbf{x}^* = \mathcal{F}(\mathbf{z}) \tag{3.4}$$

which serves as the model in a Monte Carlo simulation setup, even though the exact form of $\mathcal{F}$ is unknown and is decided by the minimizing procedure.

Now we are ready to follow Figure 3.8 to perform Monte Carlo sampling for the uncertainty in SFM. The key step in the Monte Carlo simulation is to generate the synthetic data sets $\mathbf{z}^s$, which represent the uncertainty in tracking results $\mathbf{z}$. Bootstrap is not applicable, since the **i.i.d.** assumption on $\mathbf{z}$ does not apply. As we have seen, the uncertainty of each feature correspondence depends on the individual texture property and its surroundings, therefore they are not identical. A rigorous Monte Carlo simulation is required. Fortunately we already have $\mathbf{z}_{(l)}, l = 1, ..., N$, the sampled representation for the uncertainty in $\mathbf{z}$. It is exactly what we need for the synthetic data sets $\mathbf{z}^s$ in the Monte Carlo simulation procedure. The following algorithm gives the procedure for Monte Carlo sampling of the SFM.

```
for i=1:N {
    for j=1:M {
        sample $p(\mathbf{z}^j|I)$ to get $\mathbf{z}^j_{(i)}$, according to Section 3.5.1
    }
    solve Equation 3.1 with $\mathbf{z} = (\mathbf{z}^1_{(i)}, ..., \mathbf{z}^M_{(i)})$ to get $\mathbf{x}_{(i)}$
}
return $X = (\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(N)})$
```

The uncertainty on structure and motion is represented by a set of samples $X = (\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(N)})$, in which each sample contains the structure and motion pair, $\mathbf{x}_{(l)} = (\mathbf{s}_{(l)}, \mathbf{m}_{(l)})$ computed from a sample of image feature locations $\mathbf{z}_{(l)}$ defined as in the previous section. Operationally, the SFM algorithm is executed N times, one for each sampled set of image locations, $\mathbf{z}_{(l)}$, $l = 1, \ldots, N$.

Figure 3.9 shows the structure samples of SFM, that is, for, each structure and motion pair $\mathbf{x}_{(i)} = (\mathbf{s}_{(i)}, \mathbf{m}_{(i)})$ generated from a sample $\mathbf{z}_{(i)}$, we display the M 3-D points in $\mathbf{s}_{(i)}$.

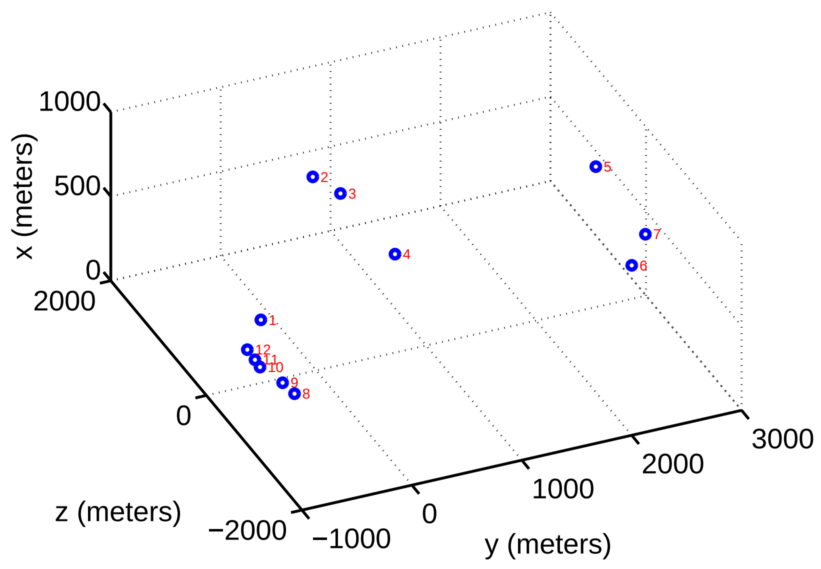


Figure 3.9: Reconstruction at true minima

3.6.1 Comparison with Importance Sampling

In [68] an importance sampling based scheme is used to generate samples for current uncertainty of SFM results.

Importance sampling is a technique to generate sample set for probability function $p(x)$ when (1) we can not sample directly from $p(x)$ and (2) it is feasible to evaluate $p(x)$ pointwise.

In [68] $p(\mathbf{x}|\mathbf{z})$ is considered as impossible to directly sample from. With Bayesian rule,

$$p(\mathbf{x}|\mathbf{z}) \propto p(\mathbf{x})p(\mathbf{z}|\mathbf{x})$$

$p(\mathbf{x}|\mathbf{z})$ is made possible to evaluate pointwise by specifying a prior for $\mathbf{x}$ and a likelihood for observation. Therefore the importance sampling scheme is readily applicable.

The usual problem with importance sampling is that the accuracy of the uncertainty representation of the weighted sample set relies on how much the proposing distribution deviates from the true distribution. Since the proposing distribution always deviates from the true distribution by an unknown amount (otherwise the distribution can be directly sampled from), it is difficult to assess the accuracy of the uncertainty representation. It is especially the case for high dimensional problems such as SFM.

The proposed sampling scheme in the previous section can be viewed as a direct method to sample $p(\mathbf{x}|\mathbf{z})$, therefore the importance sampling is avoided along with its drawbacks. To make it clearer, we do not use any proposing distribution to guide our sampling process, the samples are generated by solving SFM directly, and all the samples in turn can be treated as fair samples. That is the primary reason we have fair samples instead of the weighted samples. The advantages are (1) the accuracy of the uncertainty representation does not rely on any proposing distribution in an unknown way, it is always a good representation even with small amount of samples, since each sample is a fair sample from the underlying distribution (2) with fair samples, the proposed sampling scheme will not suffer from sample impoverishment, which is often a problem for schemes based on importance sampling.

3.6.2 Covariance vs. Sampling

In this section, we closely compare the traditional covariance based representation with the proposed sampling representation in characterizing the SFM uncertainty. We show that covariance approximation cannot capture the true uncertainty and would provide biased or erroneous estimation when (1) SNR is low in correspondence data, i.e., the noise magnitude is relatively large compared to the optical flow, (2) feature correspondence is multi-modal distribution, i.e., there is ambiguity in tracking or (3) whenever the tracking uncertainty is substantially different from a Gaussian, which happens when the feature has little or oriented texture.

We demonstrate this first with synthetic examples. We simulate a robot equipped with an omnidirectional camera and executing a forward motion. Fifty features are tracked and a two frame SFM is performed, which minimizes the reprojection error. The environment is simulated as a rectangle area, with features in front are 100 meters away and features on both sides are 20 meters away. The robot is simulated to move forward for 15 meters. Figure 3.10 shows the optical flow on the omnidirectional image.

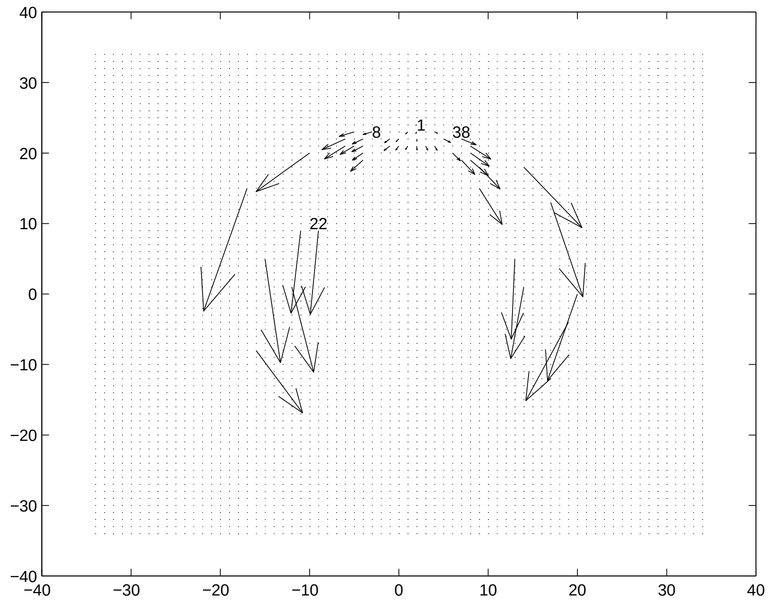


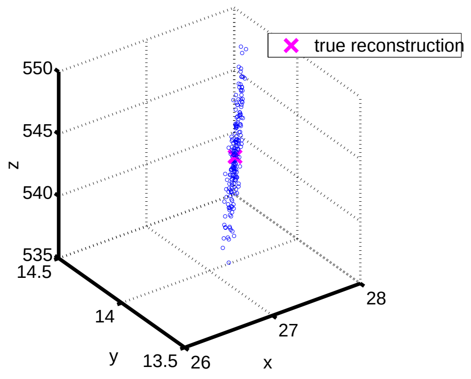
Figure 3.10: Optical flow observed from an omnidirectional camera

We first show the case when the noise in correspondence is indeed small Gaussian noise compared to the relatively large baseline. In this case, the uncertainty of SFM can be well approximated by covariance representations.

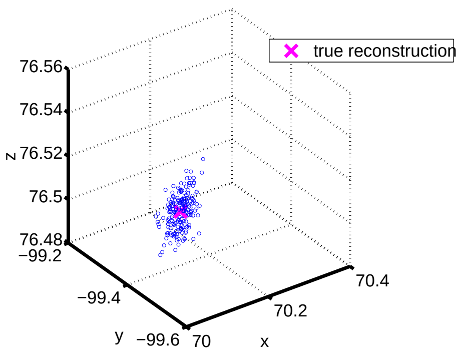
Figure 3.11 shows the Monte Carlo runs of reconstruction for feature No.1 and No.22, compared with the covariance representations at the true minima. The noise in the correspondences data is set to be Gaussian noise with σ set to be 0.1 pixel. They match with each other very well.

Since the covariance matrix is only a first order approximation to the true uncertainty, it cannot fully capture the true uncertainty when the noise is relatively large. We demonstrate this by increasing the variance of the Gaussian noise to 1 pixel.

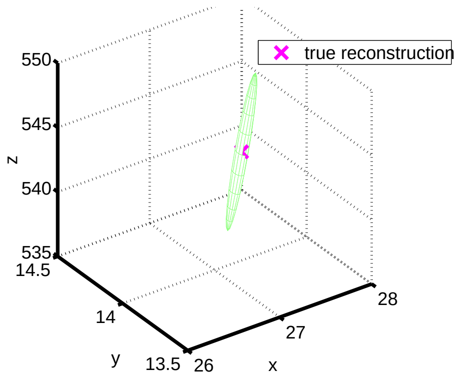
Figure 3.12 shows that with increased noise level, the covariance representation can not capture the true uncertainty for points far away but still capture the uncertainty for points close to the camera. For points far away (feature No.1 and Figure 3.12(a)) the distribution becomes long tailed



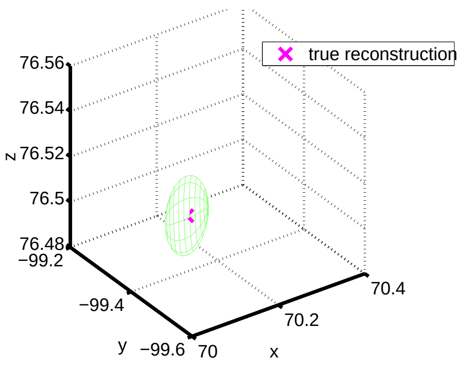
(a) Samples of reconstruction of feature No.1



(b) Samples of reconstruction of feature No.22

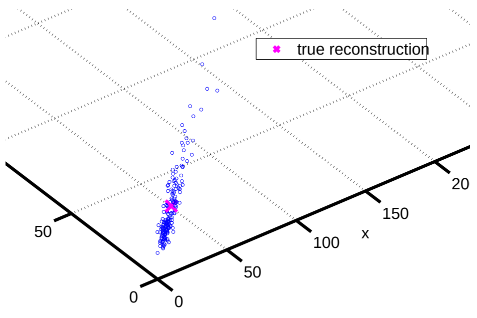


(c) Covariance approximation at the true minima

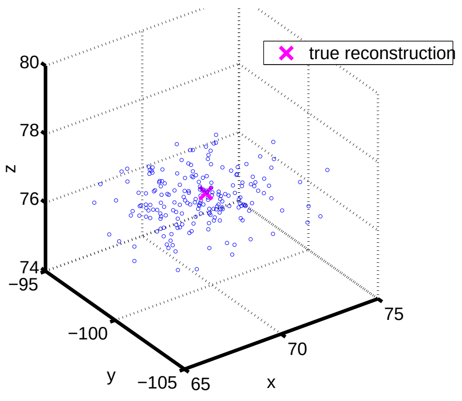


(d) Covariance approximation at the true minima

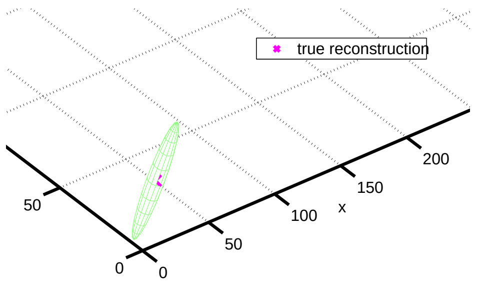
Figure 3.11: Covariance approximation at true minima



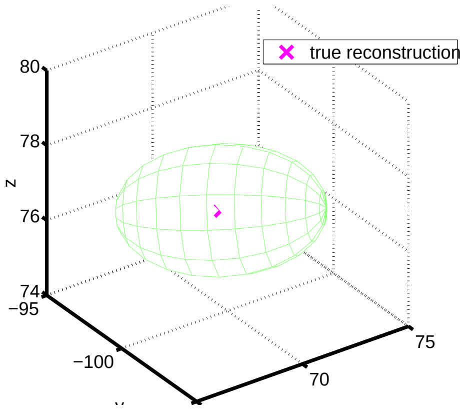
(a) Samples of reconstruction of feature No.1



(b) Samples of reconstruction of feature No.22



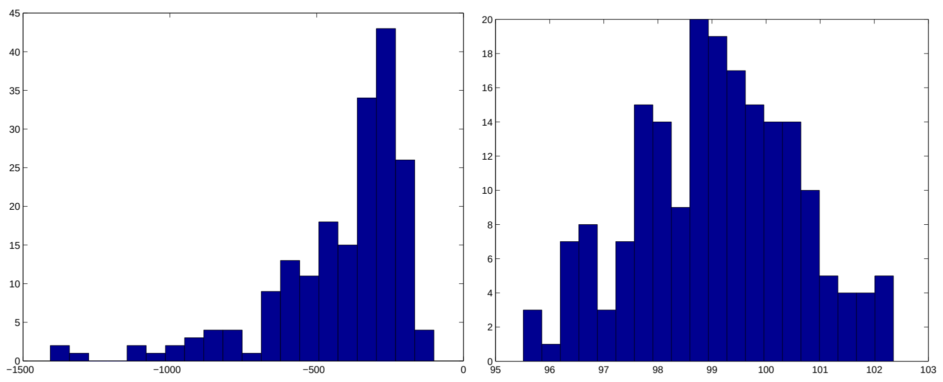
(c) Covariance approximation at true minima



(d) Covariance approximation at true minima

Figure 3.12: Comparison of sampling representation and covariance approximation in case of large Gaussian noise

which deviates substantially from the covariance representation. But for points which are close to the camera and have large optical flow magnitudes (feature No.22 in Figure 3.12(c)) the covariance is still a valid uncertainty representation. We further demonstrate the effect by projecting the reconstructions of one feature onto the main axis and compare the distribution of the projections along the main axis. We expect the distribution to be a Gaussian if the reconstructions can be approximated by the covariance matrix. In fact, we observe a distribution with a long tail for points far away, which is an indication that the underlying uncertainty cannot be fully captured by a Gaussian. In this case, the covariance approximation leads to a biased estimate. In other words, $mean(\mathcal{F}(\mathbf{x})) \neq \mathcal{F}(mean(\mathbf{x}))$. Our Monte Carlo simulation based SFM uncertainty representation can still capture the true uncertainty and therefore is unbiased.



(a) Histogram of the projections of the reconstructions for feature No.1

(b) Histogram of the projections of the reconstructions for feature No.22

Figure 3.13: Comparison of the projection distribution for feature No.1 and No.22

Like most tracking algorithms, our tracker find the correspondences by minimizing the SSD error. It works better for regions with high-contrast texture, such as corners. With other types of texture, the uncertainty of the tracking (or correspondences) can be quite arbitrary and may not be always approximated by a Gaussian. As a result, the uncertainty of the reconstructions can not always be approximated by Gaussian either. We demonstrate the argument above with synthetic examples. Instead of Gaussian noise, we simulate a cross-like distribution for feature No.1, as shown in figure 3.14. As shown in figure 3.15, the cross shaped distribution can not be represented simply with a Gaussian. It can be best viewed with a VRML model which can be found at *http://www.cs.cmu.edu/p̃eng/vrml/cross22.wrl*.

So far we have assumed unimodal distributions for tracking uncertainty. What if our tracking algorithm is confused and returns more than one possible mode for all the possible corre-

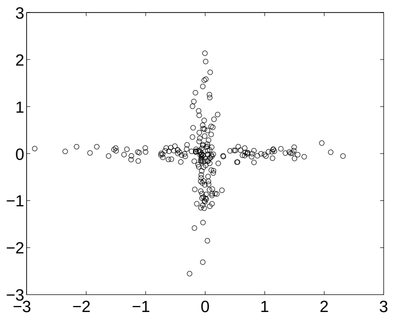


Figure 3.14: An imaginary uncertainty distribution for correspondences

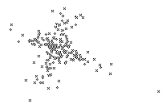


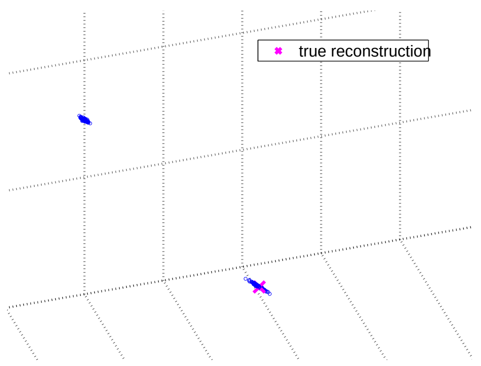
Figure 3.15: A cross section view of the reconstructions of feature No.22

spondences? We simulate this situation and Figure 3.16 shows that the distribution of reconstruction becomes multi-modal as well. In these situations, the simple covariance representation certainly would fail. Again, the VRML models for feature No.1 and No.22 can be found at *http://www.cs.cmu.edu/p̃eng/vrml/bi01.wrl and bi22.wrl.*

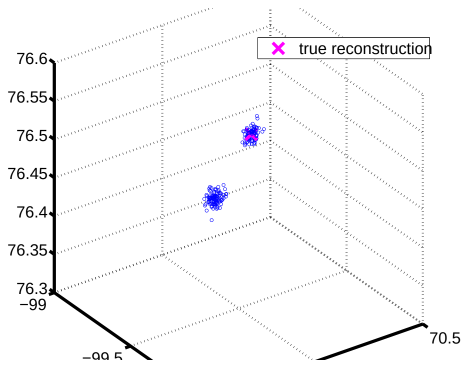
We believe we have shown the limitations of covariance approximation for the highly nonlinear SFM problem in case of large Gaussian noise, non-Gaussian noise such as multi-modal noise in the correspondence data. It in turn validates the proposed Monte Carlo sampling method in such situations, which is capable of capturing the true underlying uncertainty.

3.6.3 Sample Size

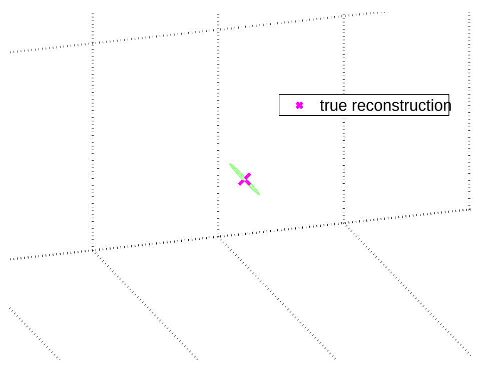
When using a sampling method to capture the underlying uncertainty, one important issue is to decide on the sample size. Even though sampling is quite general and capable of representing any distribution, it is still an approximation method. If sampled appropriately, larger sample size generally results in more accurate representation. The purpose of maintaining a distribution is often to calculate some statistics of specified variables. Therefore, the accuracy of a sampled representation can be measured by the variance of those statistics. For example, given sample set $x_i^s, i = 1...N$ from probability distribution $P(x)$, we can estimate the expectation $\hat{\Phi}$ of any given function $\phi(x)$.



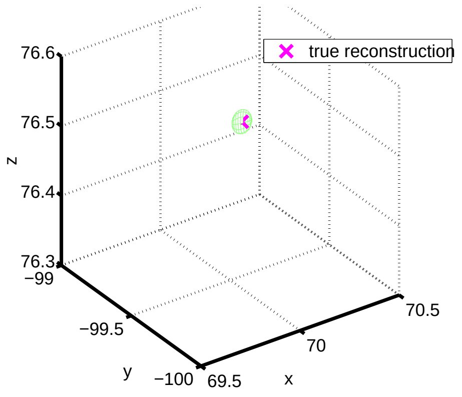
(a) Samples of reconstruction of feature No.1



(b) Samples of reconstruction of feature No.22



(c) Covariance approximation at true minima of feature No.1



(d) Covariance approximation at true minima of feature No.22

Figure 3.16: Comparison between sampling and covariance approximation in case of ambiguity

$$\hat{\Phi} \equiv \frac{1}{N} \sum_i \phi(x_i^s) \tag{3.5}$$

where $\Phi = \langle \phi(x) \rangle \equiv \int d^N x P(x) \phi(x)$. Clearly $\hat{\Phi}$ varies with different sample sets $x_i^s i = 1...N$. The variance of $\hat{\Phi}$ is denoted as $V_{\hat{\Phi}}$, and it can be estimated by repeating the sampling procedure many times. Intuitively $V_{\hat{\Phi}}$ is a function of the sample size N. Larger N leads to smaller $V_{\hat{\Phi}}$, and in turn, more accurate uncertainty representation in this sense.

Since the analytical form of the distribution is usually unknown (one of the reasons to use sampling method), it is generally not feasible to decide the sample size analytically. Monte Carlo simulation is a standard way to decide the sample size that fulfills certain accuracy requirement. Given a specified $v_{\hat{\Phi}}$, the following steps can be used to decide the sample size.

```
initialize $\sigma$, $n$
while $\sigma > v_{\hat{\Phi}}$ {
    increase $n$
    for i=1:LARGE {
        sample the distribution $p(x)$ and compute $\hat{\Phi}_i$ as in Equation 3.5
    }
    set $\sigma$ as the variance of $\hat{\Phi}_i, i = 1, ...LARGE$
}
return $n$
```

It is important to estimate the sample size as the first step in any sampling attempt. One can usually find out for the distribution at hand and with the specified sampling method, (1) if it is really feasible for the sampling scheme to capture the distribution with affordable samples, (2) if it is feasible, how many samples are needed to achieve a certain level of performance. In general, one wants to reduce the sample size in order to reduce the computational cost. It has been mentioned that sample size depends on both the underlying distribution and the effectiveness of the sampling scheme. With an inappropriate sampling scheme, it may take unnecessarily large sample size to achieve the same accuracy. It is especially true in the hypothesize-and-test sampling scheme. The effectiveness of those sampling schemes largely depends on the quality of the proposing distribution. Poor proposing distribution may lead to intolerable sampling size. In a recent paper [31] KL divergence has been proposed to measure the effectiveness of the proposing distribution and change the sample size adaptively.

In the context of high dimensional problems such as tracking and SFM, the sampling scheme is even more important compared to low dimensional problem such as 2D robot localization. With an inappropriate sampling scheme, the sample size can easily become intractable. We comment further on that point in Section 3.7.

Another concern with high dimensional problems is how the sample size relates to the dimensionality. It is actually not a concern due to an important property of Monte Carlo method: the accuracy of a Monte Carlo estimate is not related to the dimensionality of the space to be sampled from, but depends on the variance of the estimator instead [55].

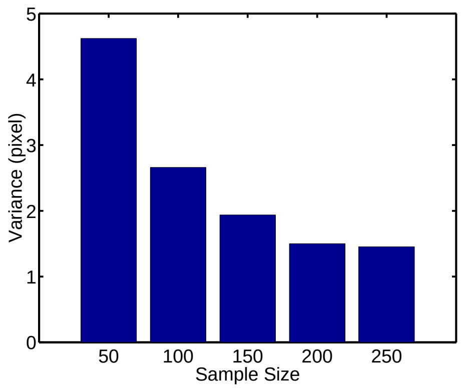
In our case, it can be shown that with our direct-estimate sampling scheme (compared to the hypothesize-and-test scheme) a small number of samples is sufficient to meet the accuracy requirement for the SFM uncertainty representation despite the high dimensionality of the sampling space.

Specifically, we choose the accuracy measurement to be the mean of the 2D reprojections of all the SFM samples, since we are most interested in fusing the SFM predictions with the tracking uncertainty on 2D. As is common practice [47] [54] [75], we evaluate the sample size from training data. We first construct a typical environment that the robot is supposed to work within. We then examine the tracking uncertainty for features at different distances. With the synthetic structure and given noise level similar to those found in the training data, we determine the sample size required for the sampled estimate to reach a pre-defined accuracy. By running the Monte Carlo algorithm described above, we plot the relationship of the sample size with the variance of the 2D reprojections of the SFM samples. The plot is shown in Figure 3.17. As predicted by the theory, the variance decreases as the sample size increases. In practice, for each feature, we choose a threshold of 2 pixels for the variance σ of the estimated mean reprojection from the sampled distribution. We find that in the worst case (for the feature far away) it requires a sample size of $N = 200$. For close features, it requires much less. As indicated in Figure 3.17(b), only 50 samples can ensure the accuracy to be less than 0.2 pixel. Figure 3.17 shows how the variances decrease with increased sample size for both distant features (Figure 3.17(a)) and nearby features (Figure 3.17(b)). Why so few samples? We believe that the small number of samples is due to the fact that the solution space is highly correlated and that the SFM algorithm fully exploits this constraint.

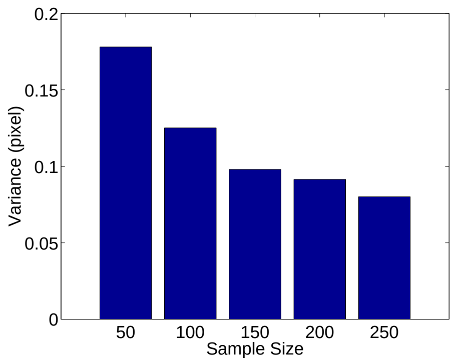
3.7 Related Work

Recently various Monte Carlo methods have been introduced to the problem of SFM. We highlight some of them which are most related to our work.

Realizing that finding correspondence is a key problem in SFM, [21] utilizes Monte Carlo EM to find the best match of feature points in multiple frames. But the problem was substantially simplified (although still a very hard problem) by pre-selecting all the feature points in the images



(a) feature No.1



(b) feature No.22

Figure 3.17: Variance of the mean reprojection v.s. sample size

and the problem becomes to find the best association among them. Random association is proposed and SFM has been used as a criteria to select the most appropriate ones. In our navigation problem we have to *automatically* track the feature points through all the frames in case of occlusion and non-smooth camera motion. The challenge here is how to deal with large local tracking error and ambiguity and reduce its effect on the robustness of SFM results.

In [30], a MCMC method is proposed as a random search scheme to solve the factorization based SFM problem. The goal of this work is quite similar to ours: to make the SFM more stable against tracker errors. There are some key differences from our approach, however. First, in [30] the correspondences are pre-computed and a batch method is taken to solve the SFM, therefore, the tracking is completely separated from SFM, whereas in our case we take a sequential method and the tracking and SFM are probabilistically integrated. The problem with batch method is that it has to try out all the possibilities in order to detect tracking outliers or occlusion, which is an extremely expensive process, as noted in [30]. Secondly, there is no explicit occlusion detection and prediction in [30] therefore the tracking is not improved in any sense while our method results more accurate correspondences and SFM by integrating information through time, as we will explain later.

In a more related work [68], a sequential Monte Carlo sampling method is proposed to solve for SFM. As noted in previous section, it belongs to the hypothesize-and-test approach and relies on prior knowledge of camera motion to generate sensible camera movement. Only when the proposing distribution of camera motion closely mimics the true camera motion uncertainty, the importance resampling scheme can succeed with tractable sample size. Unfortunately the critical problem of sampling size is not addressed in [68]. According to the motion model used in [68], it seems to difficult to deal with rough camera motion. Again, the correspondences are acquired with a separated tracking process and there is no explicit occlusion detection. Determining the sample size without prior knowledge about the camera motion or with unexpected camera motion is a hard issue for this approach.

The work in [21] [30] and [68] use MCMC sampling as a hypothesize-and-test search scheme to find an optimal solution, while in our method, the sampling is used to directly recover the uncertainty of both tracking and SFM, and based on these uncertainties we can improve the accuracy of both tracking and SFM by integrating them probabilistically and perform occlusion detections, which is the topic of the next chapter.

Chapter 4

Bayesian Filtering via Sampling

4.1 Introduction

In Chapter 2, we proposed the integrated tracking and SFM, illustrated as in Figure 2.2, reproduced below.

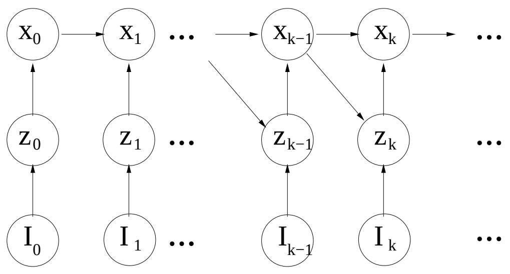


Figure 4.1: A graphical model for the proposed integrated tracking and SFM

Given this graphical model, particle filtering can be carried out as an inference problem, once the following probabilities are defined.

$$\begin{cases} p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1}) \\ p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1}) \end{cases} \tag{4.1}$$

In Chapter 3 we already described the sampling method to generate the tracking uncertainty $p(\mathbf{z}_k|I_k)$ and the SFM uncertainty $p(\mathbf{x}_k|\mathbf{z}_k)$. To recap, we define them again here:

$p(\mathbf{z}_k|I_k)$ is the tracking uncertainty. To be precise, it should take the form $p(\mathbf{z}_k|\mathbf{T}_0, \mathbf{T}_k)$, where T_0 and T_k are the initial and current templates for all the features respectively. To simplify the notation we use I_k to substitute $\mathbf{T}_k$ and omit $\mathbf{T}_0$ since it is constant. Assuming there is an affine transformation between $\mathbf{T}_0$ and $\mathbf{T}_k$ and $\mathbf{z}_k$ is the translation of this affine transformation, the probability $p(\mathbf{z}_k|\mathbf{T}_0, \mathbf{T}_k)$, or $p(\mathbf{z}_k|I_k)$, can be related to a function of the SSD error. In Section 3.5.1 a sampling method is presented to sample from this probability directly.

$p(\mathbf{x}_k|\mathbf{z}_k)$ is the SFM uncertainty, which represents the uncertainty of reconstructions given the uncertainty of feature locations. In most previous work, this uncertainty is assumed to be a Gaussian distribution. In Section 3.5.2 a sampling method is proposed to capture the true uncertainty of SFM.

In this chapter, we need to address the following aspects of the filtering algorithm:

- Generate the two probabilities in Equation 4.1, given the sampled representation for $p(\mathbf{z}_k|I_k)$ and $p(\mathbf{x}_k|\mathbf{z}_k)$, as explained in Section 4.2.
- provide a recursive algorithm to propagate the probabilities through time, as in Section 4.3;
- Detect occlusion, as in Section 4.4;
- Tolerate abrupt camera motion, as in Section 4.5.

4.2 Fusion: Two Important Probabilities

This section explains how to generate the two important probabilities defined in Equation 4.1: $p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1})$, the observation uncertainty, and $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$, the estimation uncertainty. Both probabilities have counterparts in a Kalman filter, except that in Kalman filter, they can be very efficiently computed with Gaussian assumptions. Here we seek to compute them without the Gaussian assumptions. To represent arbitrary distributions, sampled representations are used; to compute these two probabilities we apply factored sampling method with reasonable assumptions, as explained in the following.

$p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1})$ represents the fusion process which integrates current tracking uncertainty with previous reconstruction and results in more accurate feature correspondences. To directly sample this probability is difficult and we need the following assumptions to make it possible.

- All feature tracking uncertainties are conditionally independent given current templates I_k and previous reconstruction $\mathbf{x}_{k-1}$.

  It is a reasonable assumption since the uncertainty of feature tracking is caused by the independent imaging noise and property of the feature itself with its own surroundings. This assumption allows sampling feature uncertainty one by one instead of sampling all features together. Mathematically we have

$$p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1}) = \prod_{j=1\ldots M} p(z_k^j|I_k, \mathbf{x}_{k-1})$$

where z_k^j is the i_{th} feature location for time k. Now we can sample $p(z_k^j|I_k, \mathbf{x}_{k-1})$ one by one for all features and then simply concatenate them to form the sampled representation for $p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1})$. But the probability $p(z_k^j|I_k, \mathbf{x}_{k-1})$ is still hard to sample from directly. We need the following assumption to factor it into a product of two simpler probabilities so that we can sample it with factored sampling method.

- Current template I_k and previous reconstruction $\mathbf{x}_{k-1}$ are conditional independent given current feature location z_k^j. This leads to

$$\begin{aligned} p(z_k^j|I_k, \mathbf{x}_{k-1}) &\propto p(I_k, \mathbf{x}_{k-1}|z_k^j) && (4.2)\\ &\propto p(I_k|z_k^j)p(\mathbf{x}_{k-1}|z_k^j) && (4.3)\\ &\propto p(z_k^j|I_k)p(z_k^j|\mathbf{x}_{k-1}) && (4.4) \end{aligned}$$

where $p(z_k^j|I_k)$ is the current tracking uncertainty which we already have a sampled representation. $p(z_k^j|\mathbf{x}_{k-1})$ is the prediction from previous reconstruction. This is computed by assuming a constant speed model for the camera motion. Given $p(z_k^j|I_k)$ and $p(z_k^j|\mathbf{x}_{k-1})$, $p(z_k^j|I_k, \mathbf{x}_{k-1})$ can be computed by factored sampling, as in Section 4.3.3.

$p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$ represents the fusion of current reconstruction with previous reconstruction. Similarly we need the following assumption to apply the factored sampling scheme.

- Current tracking uncertainty $\mathbf{z}_k$ and previous reconstruction $\mathbf{x}_{k-1}$ are conditional independent given current reconstruction $\mathbf{x}_k$.

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1}) &\propto p(\mathbf{z}_k, \mathbf{x}_{k-1}|\mathbf{x}_k) && (4.5)\\ &\propto p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_{k-1}|\mathbf{x}_k) && (4.6)\\ &\propto p(\mathbf{x}_k|\mathbf{z}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1}) && (4.7) \end{aligned}$$

where $p(\mathbf{x}_k|\mathbf{z}_k)$ is the current SFM uncertainty which we already have a sampled representation. $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the dynamics model, which in our setting is the static environment and constant speed camera motion. With $p(\mathbf{x}_k|\mathbf{z}_k)$ and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ defined, $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$ can be sampled with factored sampling method, as in Section 4.3.3.

4.3 Filtering on SFM

The following sample sets are defined to represent the important probabilities in the integrated tracking and SFM respectively. Note that since they are all sampled directly from the underlying distributions, they are all fair sample sets, i.e., every sample in the sample set has the same weight. By directly sampling the unknown distributions we eliminate the usual difficulties with importance sampling, such as sample impoverishment, low accuracy and large sample size, etc., as discussed in Section 3.6.1.

$Z' = (\mathbf{z}'_{(1)}, \ldots, \mathbf{z}'_{(N)})$: the sampled representation for $p(\mathbf{z}_k|I_k)$, which is the tracking uncertainty measured from the image I_k directly;

$Z'' = (\mathbf{z}''_{(1)}, \ldots, \mathbf{z}''_{(N)})$: the sampled representation for $p(\mathbf{z}_k|\mathbf{x}_{k-1})$, which is the predicted correspondence from previous SFM;

$Z_k = (\mathbf{z}_{k(1)}, \ldots, \mathbf{z}_{k(N)})$: the sampled representation for $p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1})$, which is the correspondence uncertainty after fusing tracking and SFM;

$X' = (\mathbf{x}'_{(1)}, \ldots, \mathbf{x}'_{(N)})$: sampled representation for $p(\mathbf{x}_k|\mathbf{z}_k)$, which is the SFM uncertainty;

$X'' = (\mathbf{x}''_{(1)}, \ldots, \mathbf{x}''_{(N)})$: sampled representation for $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, which is the predicted structure according some motion assumption;

$X_k = (\mathbf{x}_{k(1)}, \ldots, \mathbf{x}_{k(N)})$: the sampled representation for $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$, which is the correspondence uncertainty after fusing tracking and SFM;

As a slight abuse of notations, we typically drop the subscript k for Z', Z'', X', X''. It should be noticed that they are generated at every time frame k and are not constant.

4.3.1 Uncertainty Propagation

The basic issue in uncertainty propagation is: given an uncertainty representation of the structure and motion $\mathbf{x}_{k-1}$ at time $k-1$, and a new image I_k, compute the uncertainty on the new estimate of structure and motion, $p(\mathbf{x}_k|\mathbf{x}_{k-1}, I_k)$. A crucial aspect of the problem is that we need to explicitly combine the uncertainty on tracking and the uncertainty on SFM reconstruction. This is in contrast with most prior approaches in which the two sources of uncertainty are treated separately. The problem is further complicated by the fact that features may become occluded.

We describe first the core uncertainty propagation algorithm. Practical implementation issues and occlusion detection strategy are described in Sections 4.3.3 to Section 4.8.

4.3.2 Propagation Algorithm

0. Initialization: M features $\mathbf{z}_0 = z_0^1, \ldots, z_0^M$ are selected in a reference image I_0 and the corresponding features $\mathbf{z}_1 = z_1^1, \ldots, z_1^M$ are located in the second image I_1 using the SSD feature tracker. A set Z_1 of N samples is drawn using the algorithm of Section 3.5.1. For each sample $\mathbf{z}_{1(i)}, i = 1, \ldots, N$, the corresponding structure/motion pair $\mathbf{x}_{1(i)}$ is computed. The sample set $X_1 = (\mathbf{x}_{1(1)}, \ldots, \mathbf{x}_{1(N)})$ is the representation of the uncertainty in scene structure and robot position at time 1, $p(\mathbf{x}_1|\mathbf{I}_1)$.

Step 1. At time k estimate tracker uncertainty $p(\mathbf{z}_k|I_k)$: A set Z' of samples of image locations is generated by using the result of the feature tracker in image I_k as shown in Section 3.5.1. Z' is a sampled representation of $p(\mathbf{z}_k|I_k)$.

Step 2. Propagate SFM uncertainty from time $k-1$ to time k to get $p(\mathbf{z}_k|\mathbf{x}_{k-1})$: Let $\mathbf{x}_{k-1}$ be the structure reconstructed at time $k-1$. We assume that we have a sample set X_{k-1} representing the uncertainty on structure and motion at time $k-1$, $p(\mathbf{x}_{k-1}|\mathbf{I}_{k-1})$. For each sample $\mathbf{x}_{k-1(i)}$, $i = 1, \ldots, N$, the corresponding set of 3-D points is transformed to image I_k using a motion model (a constant motion model in the simplest case), yielding a set of image locations Z''. Z'' is a sampled representation of $p(\mathbf{z}_k|\mathbf{x}_{k-1})$. We explain how to generate Z'' from X_{k-1} in the next section.

Step 3. Combine tracker and propagate SFM uncertainty $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k) \propto p(\mathbf{z}_k|\mathbf{x}_{k-1})p(\mathbf{z}_k|I_k)$: The sample set Z', representing $p(\mathbf{z}_k|I_k)$, is resampled based on weights computed from the sample set Z'', representing $p(\mathbf{z}_k|\mathbf{x}_{k-1})$. The resulting new sample set Z is a fair sample of $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k)$. The approach used for computing the weights and resampling - factored sampling - is described in detail in Section 4.3.3.

Step 4. Compute new SFM uncertainty at time k, $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) \propto p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_k|\mathbf{z}_k)$: For each element $\mathbf{z}_{(i)}$, $i = 1, \ldots, N$ of Z, the corresponding structure $\mathbf{x}_{(i)}$ is computed according to the SFM algorithm described in Appendix A. The resulting set X' is a sampled representation of the uncertainty $p(\mathbf{x}_k|\mathbf{z}_k)$. As in *Step 2*, the sample set X_{k-1} representing the reconstruction at time $k-1$ can be propagated to time k with a motion model (constant camera motion as an example). The resulting set X'' represents the distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. Given X' and X'' the sample set X_k can be obtained by factored sampling again, which is the sampled distribution for the reconstruction at time k, $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$. Details in Section 4.3.3.

It is shown in the Appendix B that this sampled representation for $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$ converges to the final uncertainty on reconstruction $p(\mathbf{x}_k|\mathbf{I}_k)$, where $\mathbf{I}_k$ represents all the images from time 0 to k, if the sample size N is made sufficiently large.

4.3.3 Factored Sampling

In both *Step 3* and *Step 4,* we need to sample the following type of distributions $p(x|o_1 o_2) \propto p(x|o_1)p(x|o_2)$, where we already have sample sets for $p(x|o_1)$ and $p(x|o_2)$. This is a sensor fusion problem, namely, we need to infer the posterior probability of x given both observations o_1 and o_2, while we have the posterior probabilities of x given individual sensor sources.

In our case, $p(x|o_1)$ and $p(x|o_2)$ are arbitrary distributions represented by samples. The fusion problem can be solved by factored sampling technique.

Factored sampling is first used by Grenander in [36] then made popular with the name *Condensation* in [46]. The purpose is to generate sampled representation for $p(x|o)$, which is too complex to compute directly. With Bayesian rule,

$$p(x|o) = kp(o|x)p(x)$$

The factored sampling algorithm first generates a random set from the prior density $p(x)$ then weights each sample with the conditional probability $p(o|x)$. Then the resampled set has a distribution approximating the desired posterior $p(x|o)$.

Here we face a slightly different problem. We need a sampled representation for $p(x|o_1, o_2)$, given sampled representations for $p(x|o_1)$ and $p(x|o_2)$ individually. By assuming that o_1 and o_2 are independent information sources,

$$\begin{aligned} p(x|o_1, o_2) &= k' p(x|o_1)p(x|o_2)/p(x) && (4.8) \\ &\propto p(x|o_1)p(x|o_2) && (4.9) \end{aligned}$$

Similarly as in the factored sampling for posterior probability, we start with the sample set for $p(x|o_1)$, for each sample x^i, we compute a weight $p(x = x^i|o_2)$ based on the sample set for $p(x|o_2)$. According to the theorem on factored sampling [36], the resampled set approximates the joint posterior distribution asymptotically. The weighting scheme can depend on the specific nature of the distribution. In the following we explain how it can be applied to *step 3* and *step 4*.

step 3 **Fusing SSD Tracking with SFM**

We now explain how to sample $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k)$, given sample set Z' for $p(\mathbf{z}_k|I_k)$ and sample set Z'' for $p(\mathbf{z}_k|\mathbf{x}_{k-1})$. Notice that since $\mathbf{z}_k$ contains all the feature correspondences, directly sampling $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k)$ is difficult. However, since the uncertainty in each feature correspondence is independent of the other feature, we can first sample $p(\mathbf{z}_k^n|\mathbf{x}_{k-1}, I_k)$ for the n_{th} feature individually and then simply combine them together to form $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k)$. Sampling $p(\mathbf{z}_k^n|\mathbf{x}_{k-1}, I_k)$ is much easier, given the sample set $Z'^n = (\mathbf{z}'^n_{(1)}, \ldots, \mathbf{z}'^n_{(N)})$ for $p(\mathbf{z}^n_k|I_k)$ and $Z''^n = (\mathbf{z}''^n_{(1)}, \ldots, \mathbf{z}''^n_{(N)})$ for $p(\mathbf{z}^n_k|\mathbf{x}_{k-1})$.

Again we have $p(\mathbf{z}^n_k|\mathbf{x}_{k-1}, I_k) \propto p(\mathbf{z}^n_k|\mathbf{x}_{k-1})p(\mathbf{z}^n_k|I_k)$. The factored sampling takes the following procedure:

For each sample $\mathbf{z}'^n_{(i)} \in Z'^n, i = 1, \ldots, N$, a weight π_i is generated as

$$\pi_i = p(\mathbf{z}^n_k = \mathbf{z}'^n_{(i)}|\mathbf{x}_{k-1})$$

Since the distribution $p(\mathbf{z}^n_k|\mathbf{x}_{k-1})$ is represented by sample set Z''^n, we need to convert it into a function. The standard way to convert a nonparametric density into a parametric density is through kernel function. By introducing a kernel function $\mathcal{D}(x)$ at each sample point, the weight can be computed as

$$\pi_i \propto \sum_{j=1}^{N} \mathcal{D}(\mathbf{z}'^n_{(i)} - \mathbf{z}''^n_{(j)})$$

Gaussian kernel function has been one popular choice. Here to simplify the computation, we choose $\mathcal{D}(r)$ to be a threshold function, instead of more complex (but smoother) functions like Gaussian kernels. Specifically, the weights are computed as follows: for each $z'^n_{(i)}$, the i_{th} sample of the n_{th} feature in the sample set Z'^n, the weight π_i is the number of sample points from Z''^n that lie within a certain distance of $z'^n_{(i)}$. In practice, a distance of 2 pixels is used to compute the weights.

Once the weights are computed, the sample set Z'^n is resampled by using π_i as the weight associated with each sample point. It can be shown that this weighted resampling procedure generates a fair sample set Z^n for $p(\mathbf{z}^n_k|\mathbf{x}_{k-1}, I_k)$ - see [47] for a justification of factored sampling and for details on the weighted resampling algorithms.

Z is constructed by concatenating all the Z^n and is the sampled representation for $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k)$. The reason we can sample the features one by one instead of sampling them all together relies on the assumption that the uncertainty of each feature location depends on the feature itself, not the other features, which is reasonable.

It is important to note that this procedure makes no assumption on the distribution of samples. In particular, the distribution is not required to be unimodal. Therefore, if there is ambiguity in the tracking, e.g., two parts of the image are similar and closely spaced, the algorithm will preserve both alternatives in the sample set until such time that they can be discriminated.

step 4 **Propagate SFM uncertainty from time $k-1$ to k**

We next explain the sampling of $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$, given sample set X'' representing $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and sample set Z for $\mathbf{z}_k$. There are different ways to complete the propagation.

Our method avoids the requirement of large sample size by solving the SFM directly, instead of a guided random search for the importance sampling based approaches. In other words, the samples for $\mathbf{x}_k$ in our method are all computed with an SFM algorithm according to the geometric constraints. An alternative view of the approach is as a constrained search in the solution space of the SFM, which usually resides in a much lower dimension space. Therefore, a small number of samples can capture the uncertainty in that space and satisfy the performance requirement.

Specifically, given sample set Z for correspondences, we solve SFM for each sample to get sample set $X' = (\mathbf{x'}_{(1)}, \ldots, \mathbf{x'}_{(N)})$ representing $p(\mathbf{x}_k|\mathbf{z}_k)$. And then we fuse X' with X'' which represents $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. Similar to Section 4.3.3, we observe

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) \propto p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_k|\mathbf{z}_k)$$

For each sample $\mathbf{x'}_{(i)} \in X'$, we need a weight proportional to $p(\mathbf{x}_k = \mathbf{x'}_{(i)}|\mathbf{x}_{k-1})$. Again, the probability $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is represented by sample set X'', and we define the probability to be:

$$p(\mathbf{x}_k = \mathbf{x'}_{(i)}|\mathbf{x}_{k-1}) = \mathcal{D}(\mathbf{x'}_{(i)} - \mathbf{x''}^*)$$

where $\mathbf{x''}^* \overset{\triangle}{=} Argmin_{\mathbf{x''} \in X''} \mathcal{D}(\mathbf{x'}_{(i)} - \mathbf{x''})$

Effectively the probability $p(\mathbf{x}_k = \mathbf{x'}_{(i)}|\mathbf{x}_{k-1})$ is measured by the similarity between $\mathbf{x'}_{(i)}$ and sample set $\mathbf{x}_{k-1}$. We use a weighting scheme which chooses the best match for $\mathbf{x'}_{(i)}$ from sample set X'' and uses the similarity between these two as the weight. Alternatively an average similarity between $\mathbf{x'}_{(i)}$ and every sample in X'' can be used as the weight.

One remaining issue is the scale factor. Since SFM algorithms can only recover the structure up to a scale factor, we need to decide the scale factor for each time frame. Assuming a static scene and $\mathbf{x}_k = \lambda \mathbf{x}_{k-1}$, λ can be decided by minimizing the least square error e,

$$e = 1 - \frac{\mathbf{x}_k^T \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T \mathbf{x}_{k-1}}{\mathbf{x}_k^T \mathbf{x}_k}$$

and the solution is

$$\lambda = \frac{\mathbf{x}_k^T \mathbf{x}_{k-1}}{\mathbf{x}_{k-1}^T \mathbf{x}_{k-1}}$$

Given sample set X_{k-1}, the previous reconstruction, and sample set X', the current SFM result, the procedure to fuse these two sample set to get X_k is listed as follows

```
for i=1:N {
    for $\mathbf{x}'_{(i)}$, find best match $\mathbf{x}_{k-1,(b)}$ in $X_{k-1}$ and $\lambda$
    scale $\mathbf{x}'_{(i)}$ with $\lambda$ to get $\mathbf{x}_{k(i)}$
    $v_i = \mathbf{m}'_{(i)} \ominus \mathbf{m}_{k-1,(b)}$ [1]
}
return $X_k = (\mathbf{x}_{k(1)}, \ldots, \mathbf{x}_{k(N)})$ and $(v_i, i = 1, ..., N)$
```

$(v_i, i = 1, ..., N)$ is used as current speed distribution.

With constant speed assumption, sample set X_{k-1} can be used for prediction at time k by simply adding the current speed estimation $(v_i, i = 1, ..., N)$ onto $\mathbf{m}_{k-1,(i)}, i = 1, ..., N$, the motion part of the sample $\mathbf{x}_{k-1,(i)}, i = 1, ..., N$.

4.4 Occlusion Handling

Occlusion is a real challenge for any tracking system. For a feature tracker, occlusion usually means the end of the tracking and the start of a search process. For a pure appearance based tracker, searching the occluded features can easily make mistakes. Many SFM algorithms discard those occluded features, and only keep the visible features in the loop. Our system keeps track of all the features, even when they are occluded, since the higher level navigation control is based on the relative position of the robot to those feature positions being tracked. These requirements present two critical technical problems: occlusion detection and tracking through occlusion.

When an occlusion does occur, the tracker would either (1) be unable to find any target within a search region or (2) find another feature with similar appearance to the tracked feature. We want to detect case (1) and avoid case (2). In reality, case (1) is relatively easy to detect by simply thresholding the SSD error or correlation value. Avoiding case (2) is considerably harder if no additional information is provided. In traditional JPDAF-type approaches [70], a gating method is used, where the feature has to be within some distance of the predicted location. The actual threshold for the distance is decided by the assumed Gaussian covariance in measurement noise and system dynamics noise. As we argued in previous chapters, the covariance representation can not fully capture the uncertainty in SFM algorithms, and therefore any gating method based on covariance matrices can lead to errors.

[1] $\ominus$ is used here to represent the subtraction operation defined in the motion vector space.

Our sampling based method can capture the true uncertainty of SFM. We now explain how to detect occlusion and guide the search process during the occlusion of the features.

Our SSD tracker can detect case (1) occlusion by thresholding the correlation value. The case (2) occlusion can be detected only with structure information of the occluded feature. The principle is to check the consistency between the tracker location from current frame and the SFM results from previous frames. Case (2) occlusion are detected at *Step 3* of the algorithm, that is, during the fusion of SSD tracking and SFM. Recall that in Section 4.3.3, for the n_{th} feature we have the sample set $Z'^n = (\mathbf{z}'^n_{(1)}, \ldots, \mathbf{z}'^n_{(N)})$ for $p(\mathbf{z}^n_k|I_k)$ and $Z''^n = (\mathbf{z}''^n_{(1)}, \ldots, \mathbf{z}''^n_{(N)})$ for $p(\mathbf{z}^n_k|\mathbf{x}_{k-1})$, we then compute the weight π_i for each sample $\mathbf{z}'^n_{(i)} \in Z'^n, i = 1, \ldots, N$. In our algorithm, π_i are the number of all the samples from Z''^n that fall within a 2-pixel neighborhood of $\mathbf{z}'^n_{(i)}$. The consistency of Z'^n and Z''^n can be measured by the sum of weights $C = \sum_{i=1}^{N} \pi_i$. If $C < T$ then the n_{th} feature is declared as occluded. T is a threshold that is currently set at $N/2$. It is worth noting that, in practice, the exact value of T is not critical to the performance of the algorithm.

It is important for features that are occluded to be allowed to "re-appear" at a later time. To allow this to happen, all the features currently occluded are examined after *Step 4* for possible re-insertion in the list of visible features. If feature j is flagged as occluded at time $k-1$, then, at time k, it is projected to I_k using the estimate of the motion $\mathbf{m}_k$. The tracker searches around this predicted location and a decision is made as to the visibility of the feature using the same consistency criteria to decide if the feature is visible or not. In practice, we use a same threshold $T = N/2$ to decide a feature has "re-appeared" if $C > T$ where C is the sum of weights as described above.

Figure 4.2 shows different situations in which occlusion occur. Figure 4.2(a) shows the samples of tracking uncertainty and SFM predictions are consistent when there is no occlusion. Figure 4.2(b) shows that they are no longer consistent if the target is occluded and tracker finds another similar feature by mistake. This inconsistency is applied to detect the occlusion, which is hard to notice if only appearance model is used. Figure 4.3 shows the effect of resampling. In some cases, such as searching over enlarged area, the tracker may return multiple locations for the feature. This ambiguity is reduced by fusing with the SFM predictions.

4.5 Handling Sudden Camera Motion

Sudden camera motion is another realistic challenge for SFM in visual navigation. Camera motion is often assumed to be smooth for most sequential SFM algorithms. By sudden camera motion we mean the cases when the actual camera motion is beyond the uncertainty model assumed for the camera movement. Sudden camera motion is not unusual in a navigation practice when the robot makes sharp turns or bumps into some small objects at high speed. How to handle it in a principled way within a Kalman filtering framework is unclear. It also poses significant difficulty for *Condensation* type sequential sampling based methods, since the samples generated according to the assumed camera motion model will not lie in the vicinity of the true motion, which is the key for this type of methods to succeed.

We show that the sudden camera motion is handled naturally within our sampling framework. In presence of a sudden camera motion (usually caused by large rotation), most feature trackers will report lost tracking due to the large displacement of the feature points. In turn, the search region for each feature will be enlarged automatically in the hope of recovering the lost features. Given the enlarged search region, the chance that the tracker finds some wrong locations also increases. It is often the case that with enlarged search regions, the tracker can find several possible locations as the potential matches for the lost feature points, which means the tracking uncertainty is multi-modal. If the tracking uncertainty is modelled as Gaussian and in turn the SFM uncertainty is also Gaussian, then one mis-tracked feature can corrupt the entire reconstruction. With the sampling based uncertainty the tracker ambiguity is properly represented by all the samples, in other words, both the correct and wrong correspondences are in the sample set. Our SFM sampling and fusion process in *Step 4* ensures that only the SFM estimate from correct correspondences survive, since the SFM estimate from wrong correspondences is unlikely to be consistent with previous SFM results.

4.6 Complete Algorithm with Disturbance Handling

We provide a complete algorithm for the integrated tracking and SFM. Note that it is the same as in Section 4.3.2, except it includes handling of special cases of occlusion and large motion.

0. Initialization: M features $\mathbf{z}_0$ are selected in a reference image I_0 and the corresponding features $\mathbf{z}_1$ are located in I_1. The correspondence sample set Z_1 is drawn from $\mathbf{z}_1$ and the structure sample set X_1 is computed accordingly, representing $p(\mathbf{x}_1|\mathbf{I}_1)$. Reset $\mathbf{O}$, the binary indicator of occlusion for each feature, assuming all features are visible for the first two frames. (Drop those features with low correlation values if there is any.)

Step 1. At time k estimate tracker uncertainty $p(\mathbf{z}_k|I_k)$: Appearance based uncertainty, which is represented by sample set Z', is drawn from the tracker result in image I_k. If the tracker does not find a match for i_{th} feature, then the occlusion flag O_i is set, and the corresponding field in Z' is empty. Decide if the camera motion is smooth according to Section 4.5 and set the flag $suddenMotion$ accordingly.

Step 2. Propagate SFM uncertainty from time $k-1$ to time k to get $p(\mathbf{z}_k|\mathbf{x}_{k-1})$: Propagate the structure and motion sample set X_{k-1} to time k with constant motion model and then project it to the image plane obtaining the prediction sample set Z'', which is a sampled representation of

$p(\mathbf{z}_k|\mathbf{x}_{k-1})$.

Step 3. Combine tracker and propagated SFM uncertainty $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k) \propto p(\mathbf{z}_k|\mathbf{x}_{k-1})p(\mathbf{z}_k|I_k)$: Do the following loop

```
for each feature $i$
    if $suddenMotion$
        set $Z = Z'$, bypassing the fusion with SFM.
    else
        if $O_i$ is not set (visible)
            check the consistency of sample sets $Z'^i$ and $Z''^i$
            if consistent
                the sample set $Z'$, is resampled based on $Z''$. The resulting new sample set $Z$ is a fair sample of $p(\mathbf{z}_k|\mathbf{x}_{k-1}, I_k)$
            if not consistent
                an occlusion is declared and $O_i$ is set
        if $O_i$ is set (occluded already)
            check the field $Z'^i$
            if empty (no match)
                feature is still occluded, set the search region according to $Z''^i$, which is the prediction from SFM
            if not empty (possible recovery)
                check the consistency of sample sets $Z'^i$ and $Z''^i$
                if consistent
                    a recovery is declared and $O_i$ is reset
                if not consistent
                    feature is still occluded, set the search region according to $Z''^i$, which is the prediction from SFM
end
```

Step 4. Compute new SFM uncertainty at time k $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) \propto p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_k|\mathbf{z}_k)$: For each sample i in Z, the corresponding structure $\mathbf{x}_{(i)}$ is computed. The resulting sample set X' is a sampled representation of the uncertainty $p(\mathbf{x}_k|\mathbf{z}_k)$. Same as in *Step 2*, the sample set X_{k-1} representing the reconstruction at time $k-1$ can be propagated to time k with a motion model (constant motion as an example). The resulting set X'' represents the distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. Given X' and X'' the sample set X_k can be obtained by factored sampling, which is the sampled distribution for the reconstruction at time k, $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$.

4.7 Discussions

In previous sections, we discuss how the occlusions and large camera motion are handled. As the way presented here, they fall outside of the probabilistic framework. They are treated as "outlier" cases which our algorithm can detect and handle correctly. There are two thresholds we use to detect feature occlusion and large camera motion respectively.

- We use $N/2$ (N is the number of samples) as a threshold to detect occlusion for one feature. If the overlap of the SFM prediction and the current tracking is less than $N/2$, an occlusion is declared. Certainly $N/2$ is not a hard threshold and other numbers can be used too, depending on how conservative the algorithm needs to be. As discussed before, the occlusion detection is particularly difficult, if the original target is occluded while the tracker finds another similar target nearby. In this case, the SFM prediction is unlikely to overlap with the tracking at all, therefore any number reasonably large can be used. The larger the threshold, the more conservative the algorithm is.

- We use $M/2$ (M is the number of features) as a threshold to detect large camera motion. If most of the features are found to have inconsistent SFM predictions with current tracking, then a large camera motion is declared. Again, $M/2$ is not a hard threshold. A more conservative algorithm may require less inconsistent features to declare a large camera motion.

It is possible to include both the occlusion and large camera motion into the probabilistic framework. In order to do that, we need to define a prior probability for both occlusion and large camera motion and then convert the inconsistency measurements into probabilistic functions.

4.8 Sample Impoverishment

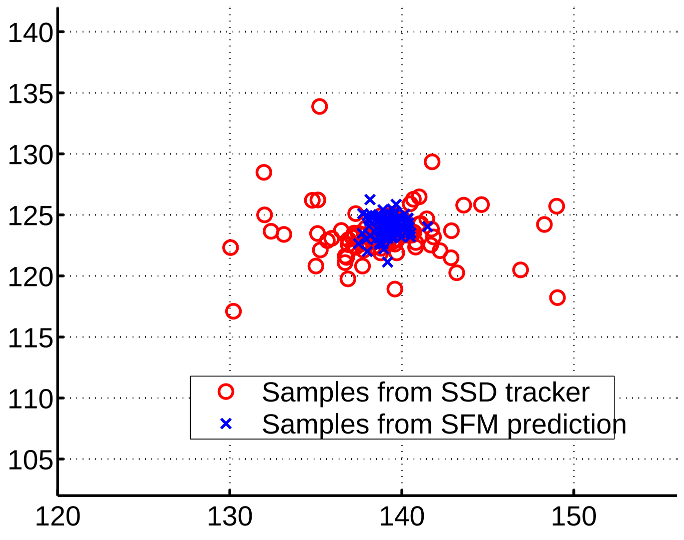
Sample impoverishment is a concern for any approach using samples to represent uncertainty [14]. This occurs when the sample size is not large enough to represent the uncertainty in the system. It is especially a concern if the sample set is generated from a re-sampling process such as in *Condensation*, since it is likely that several good samples will receive most of the weights and they will have many duplicates in the sample set. Sample impoverishment is usually alleviated by introducing artificial system noise, so that the samples are slightly perturbed and they are at least not the same. In fact this injected system noise can not eliminate the sample impoverishment phenomenon since most of the samples are still clustered around the few good candidates. Furthermore, the amount of noise to inject is always an issue.

However, sample impoverishment is *not* a problem for our method. There are two reasons.

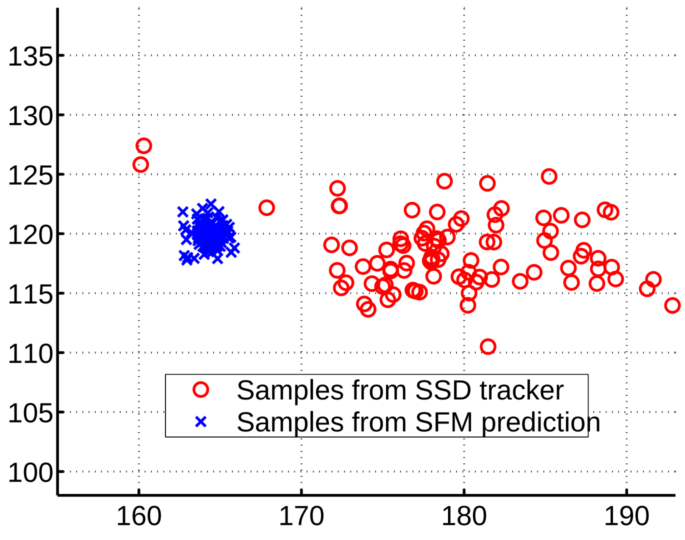
(1) As explained in Section 2.2.5, the way that we perform SFM to estimate the system state directly guarantees that the samples representing the state uncertainty is the most effective according to the KL divergence criteria [31].

(2) Another key difference between our sampling based approach and the conventional particle filtering approaches is that we generate samples from $p(\mathbf{x}_k \mid \mathbf{z}_k)$ at every time step, which means we effectively generate new samples for state variables at each time. In contrast, the usual particle filters only recursively resample from current sample set thus no new samples are generated.

Based on these two factors, the chance of sample impoverishment in our system is minimized. Sample impoverishment only happens when the SFM result is very uncertain and cannot be captured by the usual amount of samples. A possible scenario is that there is too much ambiguity in the correspondence. For example, if every feature has a multi-modal distribution, the resulting SFM solution space will be too large to be captured by limited sample size. But most scenes in real life contain enough distinguishable features so that the ego-motion and scene structure can be interpreted.

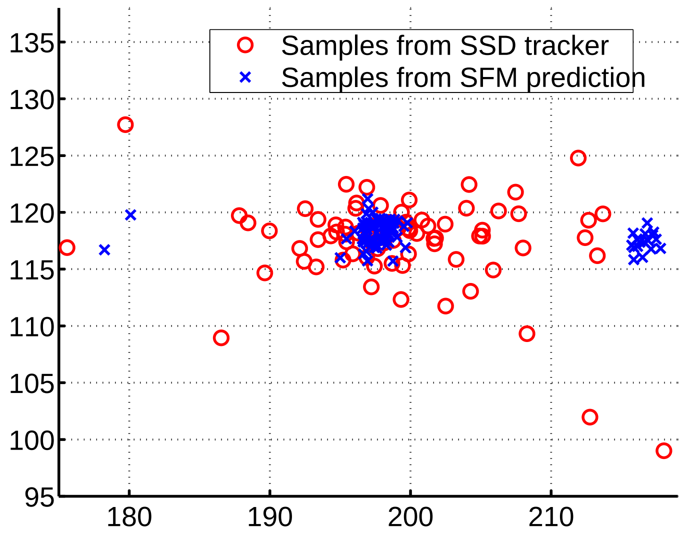


(a)

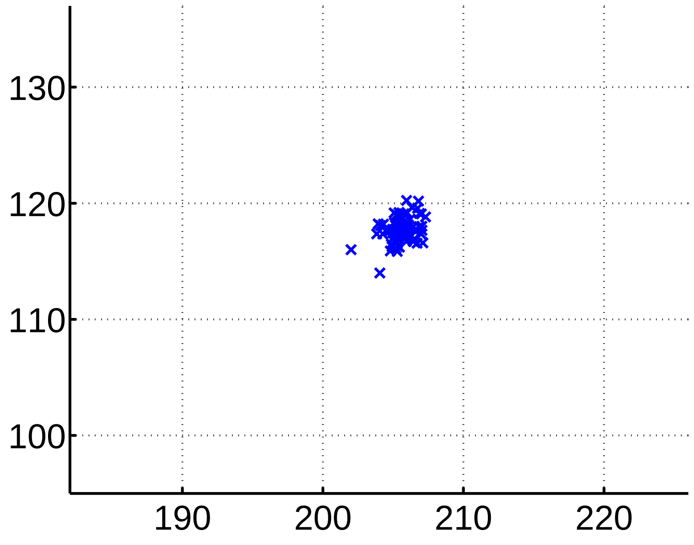


(b)

Figure 4.2: (a) no occlusion: two sample sets are consistent (b) occlusion case (2): two sample sets are inconsistent



(a)



(b)

Figure 4.3: (a) two sample sets before resampling (b) resampled sample set combining information from both tracker and SFM

Chapter 5

Extensions to Planar Patches

5.1 Introduction

Planar scenes are ubiquitous in man-made environments. In addition, many scenes can be approximated by planar scenes if viewed from a distance. In this chapter, we study the problem of planar-patches based structure from motion (PSFM) with a fixed calibrated camera. The structure reconstructed from PSFM can be included into the framework we propose in Chapter 2.

5.2 Previous work on PSFM

The perspective projections of a plane in different views are related by homography transformations. Formally, given the projection matrices for two views, $P_1 = K_1[I|\mathbf{0}], P_2 = K_2[R|\mathbf{t}]$, where K_1 and K_2 are calibration matrices, the homography induced by a plane defined by $\mathbf{n}^T X + d = 0$, $d \neq 0$ can be computed as Equation 5.1.

$$H = K_2(R - \mathbf{t}\mathbf{n}^T/d)K_1^{-1} \tag{5.1}$$

Given the homography induced by a plane and known camera calibration matrices, the camera motion $(R, \mathbf{t})$ and plane structure $(\mathbf{n}, d)$ can be recovered up to a scale. An algorithm is given in [28] for structure from motion in a piecewise planar environment with a calibrated camera. It is solvable by counting the number of variables: there are 10 unknowns with 9 degrees of freedom ($\mathbf{t}$ is up to a scale) and we have a total of 9 constraints.

If we have more than one plane in the scene, the fundamental matrix can be estimated from homographies associated with two or multiple planes as in [53], but the authors also report that the technique is not very stable. With more than one plane, the camera can be calibrated. In [93] a linear algorithm was presented to recover focal length, camera motion and planar structure from two or more homographies. In [83] various methods to perform plane based SFM with points on planes are studied, and as a result the authors advocate using hallucinated points on planes instead of using homographies directly. In most of the previous work, bundle adjustment is applied to the feature points on the planar patches as the final refinement step. It is worth noting that some recent work has been done on enforcing the planarity constraint within the bundle adjustment framework [3].

Most of the previous approaches on plane based SFM are feature based approaches, i.e., they require correspondences of feature points first and then minimize the distance between those correspondences. It is accurate given good feature correspondences. See [88] for a discussion of this class of approaches.

Another class of approaches use the intensity of the image pixels directly, and is often referred to as a direct method in the literature. This class of techniques often introduce a parametric transformation over regions or all pixels on two or more images and then recover those camera motion or scene structure parameters by minimizing an objective function based on image intensities, therefore naturally bypassing the difficulty of finding correspondences. See [44] for more discussion on direct methods. For the problem of PSFM, we claim that the direct method behaves better in presence of noise.

5.3 PSFM: Direct Method

5.3.1 Problem Statement

We are interested in recovering the structure of a piecewise planar scene along with camera motion from multiple images. Suppose the planar scene is composed of N independent planar patches, each of which can be parameterized by the surface normal $\mathbf{n}_i$ as $\mathbf{n}_i^T X = 1$. And $\mathbf{m}_j$ is the camera motion at j_{th} frame with respect to the first frame. Given image regions M_{ij} corresponding to the projection of j_{th} patch on the i_{th} frame, the planar scene $\mathbf{n}_i, i = 1...N$ and motion $\mathbf{m}_j, j = 1...K$ can be obtained by minimizing the following objective function

$$\mathcal{C}_1(\mathbf{p}) = \sum_{j=1}^{K} \sum_{i=1}^{N} (T_i - \mathcal{P}(M_{ij}; \mathbf{p}))^2 \tag{5.2}$$

where $\mathbf{p}$ contains all the motion and structure parameters $(\mathbf{n}_i, \mathbf{m}_j, i = 1...N, j = 1...K)$. $T_i \overset{\triangle}{=} M_{i0}$ is the i_{th} patch template on the first image, and $\mathcal{P}$ is a transformation between M_{ij} and T_i, which is well-known to be a homography and depends on $\mathbf{p}$.

Equation (5.2) provides a general form for PSFM from multiple views by minimizing the image measurement directly. However, it is a nonlinear optimization problem in high dimension. Solving it directly with nonlinear optimization method is at least expensive if not impossible. Therefore instead of a batch method, we choose to use a sequential method. Namely, we compute the reconstruction for two frame pair first, and then integrate them through the time domain with probabilistic filtering method to achieve optimal result. In Section 5.4 we detail the explanation for the two view case.

In a much related work [94], an elegant direct method was presented to recover planar motion from multiple frames. But since their goal was to perform plane alignment, only the 2D motion parameters are recovered while the planar structure and camera motion are not explicitly recovered. Furthermore, although the subspace method has the advantage of using multiple images simultaneously, it has to assume small motion throughout the sequence while our PSFM do not have this assumption.

This work is based on recent development on image alignment algorithms. A nice survey can be found in [2]. In [38, 74, 2] efficient algorithms were developed to align image regions with certain parametric deformations, notably the homography and affine models. We show in this paper that with a typical parameterization for multiple planar scene and camera motion, those algorithms can be further extended to recover the planar structure and motion efficiently, by minimizing the SSD error across multiple images regions simultaneously.

5.4 PSFM: Two Frames

Using the notations in Sect. 5.3, the objective function for the two-view case becomes

$$\mathcal{C}_2(\mathbf{p}) = \sum_{i=1}^{N} \sum_{\mathbf{x}} (T_i(\mathbf{x}) - M_i(\mathbf{W}(\mathbf{x};\mathbf{p}))^2 \tag{5.3}$$

where $\mathbf{p} = [\mathbf{n}_1; ...; \mathbf{n}_N; \mathbf{m}]$. Again, $\mathbf{n}_i$ is the scaled normal of the i_{th} patch, which satisfies $\mathbf{n}_i^T X = 1$ for all points on that plane. Note that this parameterization excludes all the planes intersecting the principal point of the reference view (therefore, invisible in reference view). $\mathbf{m}$ is the motion between the two views. $\mathbf{W}$ represents the homography warping function between two image coordinates, and it depends on $\mathbf{p}$.

Obviously, it is still a difficult optimization task in high dimension. To make it practical, we need a reasonable initialization and an efficient optimization algorithm. In practice we first track each planar patch with a SSD based homography tracker. Once we have the initial homography estimation H_i for each patch, the initial estimate $\mathbf{p}_0$ can be solved with a linear algorithm and used as the initialization to the nonlinear step in minimizing (5.3). In Section 5.4.1 we describe the linear method to recover $\mathbf{p}_0$ given a set of homographies H_i. In Sect. 5.4.2 a gradient descent algorithm is presented to recover $\mathbf{p}$ based on (5.3).

5.4.1 Initializing the PSFM

Initial values of the homographies H_i are obtained by tracking each patch individually [38, 74, 2]. Given the homography $H_i, i = 1, .., N$ the rotation R and translation $\mathbf{t}$ and the plane parameters $\mathbf{n}_i$ such that $H_i = R + \mathbf{t}\mathbf{n}_i^T$ are recovered using a direct method. First R and $\mathbf{t}$ are recovered from the sum of the homographies H_i using a method similar to the one described in [28]. This step also generates an estimate for the sum of the planes $\mathbf{n}_i$, S_n. Finally, the individual planes $\mathbf{n}_i$ are recovered linearly by finding the planes $\mathbf{n}_i$ such that $S_n = \sum \mathbf{n}_i$ and $H_i = R + \mathbf{t} * \mathbf{n}_i^T$ for all i. Although this method provides only an approximation to R, $\mathbf{t}$, and $\mathbf{n}_i$, it is sufficient to initiate the minimization below and it is a simple direct method computed directly from the homographies. Our experiments show that the reprojected homographies always closely match the original homographies from the tracking algorithm, which is critical for the nonlinear optimization step.

5.4.2 PSFM: Nonlinear Optimization

The parameterization for the nonlinear PSFM step is as follows: The rotation is represented by Euler angles (α, β, γ). Since the translation can only be recovered up to a scale, it is represented by a unit direction vector which can be parameterized by two additional angles (θ, ϕ). Therefore, $\mathbf{p}$ is parameterized as $[\mathbf{n}_1; ...; \mathbf{n}_N; \alpha, \beta, \gamma; \theta, \phi]$.

As a baseline method for nonlinear least square problem, we tested the PSFM using the Levenberg-Marquard method. Generally speaking, Levenberg-Marquard is a better optimization method, but in the case of PSFM, it is more complicated to implement and much less efficient computationally. Targeted for real-time applications, also as a natural extension to the existing image alignment algorithms, we derive a gradient descent algorithm which achieves comparable results while computationally efficient. We now describe the gradient descent iteration. Assuming we have an initial estimate $\mathbf{p}_0$, the update $\Delta\mathbf{p}$ can be found by minimizing the following metric:

$$\mathcal{C}(\Delta\mathbf{p}) = \sum_{i=1}^{N} \sum_{\mathbf{x}} (T_i(\mathbf{x}) - M_i(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})))^2 \tag{5.4}$$

A first order Taylor expansion to $\mathcal{C}(\Delta\mathbf{p})$ gives:

$$\sum_{i=1}^{N} \sum_{\mathbf{x}} (e_i(\mathbf{x}) + \nabla M_i \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p})^2 \tag{5.5}$$

where $e_i(\mathbf{x}) = M_i(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_i(\mathbf{x})$. Define $A_i = \nabla M_i \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, the least square solution to (5.5) is

$$\Delta \mathbf{p} = -(\sum_{i=1}^{N} A_i^T A_i)^{-1} \sum_{i=1}^{N} A_i^T e_i \tag{5.6}$$

Of course, one can choose to compute $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ directly, but then the Jacobian matrix becomes quite involved. Alternatively, the computation can be simplified by using the chain rule, $A_i = \nabla M_i \frac{\partial \mathbf{W}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{p}}$. If H is the current homography matrix in function $\mathbf{W}$, then $\mathbf{h}$ is the vector containing all the elements in matrix h, where $H = I + \mathbf{h}$. It has been shown repeatedly in the literature that

$$\frac{\partial \mathbf{W}}{\partial \mathbf{h}} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x^2 & -xy & -x \\ 0 & 0 & 0 & x & y & 1 & -xy & -y^2 & -y \end{bmatrix} \tag{5.7}$$

where (x, y) is the image coordinate. Additionally, $\frac{\partial \mathbf{h}}{\partial \mathbf{p}}$ can be evaluated according to $H = K_1(R + t * n^T)K_2^{-1}$, where $K_{1,2}$ are known calibration matrices in our case.

According to a recent survey on image alignment algorithms [2], the above algorithm falls into the category of the additive alignment algorithm. It is a less efficient algorithm since at each iteration, the image Jacobian and Hessian matrices have to recomputed. Following the terminology of [2], a more efficient algorithm is called compositional image alignment. Instead of updating $\mathbf{p}$ additively as $\mathbf{p} = \mathbf{p} + \delta\mathbf{p}$, a compositional update uses $\mathbf{W}(x;\mathbf{p}) = \mathbf{W}(x;\mathbf{p})\mathbf{W}(x;\delta\mathbf{p})$. By updating $\mathbf{p}$ compositionally, the Jacobian and Hessian are only evaluated once at the beginning. It requires that the warping function forms a semi-group, that is, $\mathbf{W}(x;\mathbf{p})\mathbf{W}(x;\delta\mathbf{p})$ is always a valid warp function which can parameterized as $\mathbf{W}(x;\mathbf{p})$, see [2] for details. Unfortunately this requirement can not be satisfied for multiple PSFM problem with our parameterization.

All we need to show is that $(R_0 + \mathbf{t}_0\mathbf{n}_0^T) * (\delta R + \delta\mathbf{t}\delta\mathbf{n}^T)$ can not be represented by $R_1 + \mathbf{t}_1\mathbf{n}_1^T$, where R_1 and $\mathbf{t}_1$ are independent of $\mathbf{n}_0$ and $\delta\mathbf{n}$. Expanding $(R_0 + \mathbf{t}_0\mathbf{n}_0^T) * (\delta R + \delta\mathbf{t}\delta\mathbf{n}^T)$ leads to $R_0\delta R + A$, where $A = \mathbf{t}_0\mathbf{n}_0^T\delta R + \mathbf{t}_0\mathbf{n}_0^T\delta\mathbf{t}\delta\mathbf{n}^T + R_0\delta\mathbf{t}\delta\mathbf{n}'$. Then R_1 has to be equal to $R_0\delta R$, but it is impossible to decompose A as $\mathbf{t}_1\mathbf{n}_1^T$, because A is of rank 2 in general.

Although the parameterization is only an approximation, our experiments show that the compositional algorithm still achieves comparable results with the additive algorithm, in terms of both SSD error and 3D reconstructions. Therefore it still can be used in situations when computing cycle is most critical.

5.5 Examples

To demonstrate the usefulness of the algorithm we ran it over sequences taken from a robot equipped with an omnidirectional camera moving through an urban environment. The omnidirectional camera is calibrated and then the image is warped onto eight concentric virtual perspective cameras. The user selects several textured planar patches and the homography tracking is performed for each patch individually. Then the estimated homographies are fed into the linear algorithm described in Sect. 5.4.1. Using the result from the linear algorithm as an initial value, the nonlinear optimization is carried out to achieve the final estimation based on minimizing the total SSD error. To make a comparison with the proposed gradient descent algorithm, we also show the minimization result with a standard Levenberg-Marquard method. Ideally the larger the planar patch in the image, the better. Limited by the resolution of the omnidirectional camera, a typical patch window in the image is 50×50. We show two sets of results in the following.

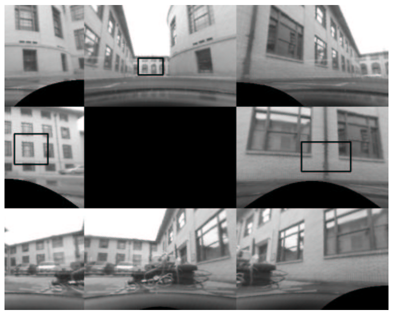
Figure 5.1 shows the tracking result for four planar patches selected from the scene. They are all windows on the surrounding buildings. The tracker estimates the corresponding homography transformation for each patch.

The homographies estimated from the tracker are used by the linear algorithm to generate the initial estimation. Figure 5.2 shows the PSFM result obtained from the linear algorithm. It can be seen that the positions of all the patches are qualitatively correct, but the patches are deformed. The top view shows that all the patches close to vertical as they should be.

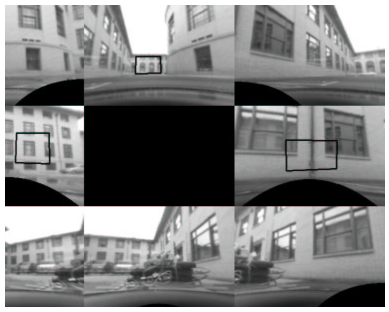
Figure 5.3 shows the PSFM result obtained by minimizing (5.4) with the Levenberg-Marquard method. Judging from observation, we find that the patches are less deformed and the shape and position are closer to the true structure. For example, the patch in the front is much larger than the others, and the reconstruction reflects that. And the top view shows that again all the planes are close to be vertical and the parallel planes are preserved as parallel Figure 5.4 shows the PSFM result achieved by the gradient descent method. It is at least comparable with the LM method.

It is worth noting that the nonlinear optimization seems to always find a good reconstructions even when the initial reconstructions given by the linear algorithm are off substantially. Figure 5.5 shows one such example. Note that even the 3D reconstructions are largely wrong, the resulting homographies still match the ones estimated from tracking, therefore it is still a legitimate starting point to minimize the SSD error. This example also shows the instability of SFM from homographies directly.

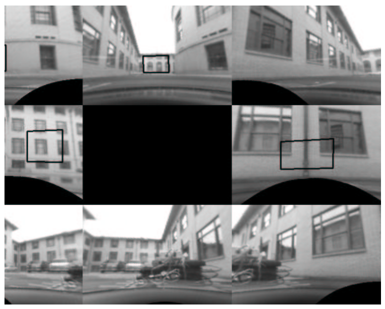
Figure 5.6 illustrates the SSD errors versus the iteration cycles for the three methods used in our experiment: the Levenberg-Marquard, the additive gradient descent and compositional gradient descent. They all start from the same initialization from the linear algorithm. As can be seen, the nonlinear optimization brings the total SSD error down by substantial amount. It also suggests that



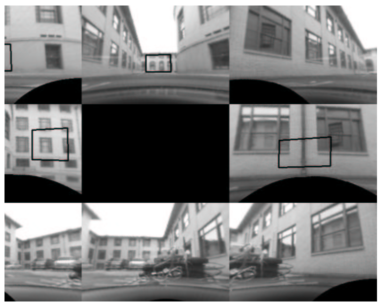
(a)



(b)



(c)



(d)

Figure 5.1: (a) four selected planar patches (b, c, d) homography tracking at three intermediate frames

all the nonlinear optimization methods are comparable, but the gradient descent algorithms are more computationally efficient.

5.6 Discussion

As stated earlier, according to [94] it is possible to recover the planar structure and motion with factorization based method. It has the advantage of using multiple frames simultaneously, but it assumes small motion and, as a linear method, the planarity constraints are only enforced by rank constraints. With the nonlinear optimization PSFM, there is no assumption on small motion and the planarity constraints are enforced by parameterization, therefore, it leads to a more accurate result, which in fact is the maximum likelihood estimation.

The results we show in this paper are based on omnidirectional camera. A potential advantage is that it sees all the scene structure around it, therefore we can select planar structures that can not usually be seen in a conventional camera. The influence of the spatial distribution of the planar scene to the stability of the algorithm is still under investigation

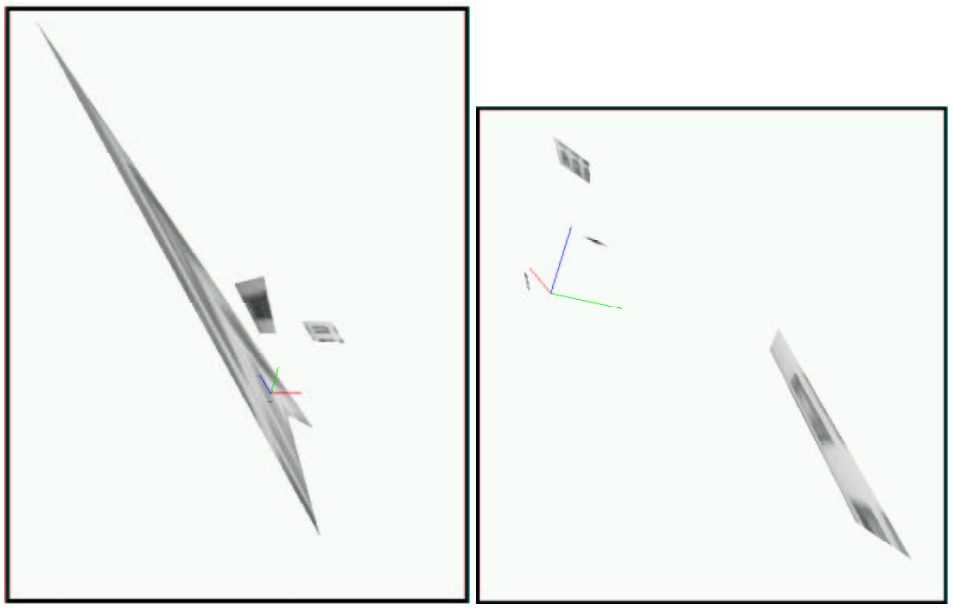


Figure 5.2: (a) linear reconstruction (b) top view

5.7 Conclusions

We have presented an efficient algorithm to recover planar scenes and camera motions. Unlike most previous work, it does not require feature correspondences. The user only need to select several planar patches in the scene to start with, and the system can automatically track the planar patches and recover the planar structure and motion. Since this method relies on image measurement directly, it appears to be more reliable than previous methods which make explicit use of the homographies. Our experiments on real images further confirm this. In practice, by minimizing the SSD error, the algorithm finds reasonable reconstructions even when the initialization from the linear algorithm are far off the ground truth.

5.8 Integration with the Probabilistic Framework

The ultimate goal of a probabilistic framework is to integrate information and to have more reliable results.

In previous chapters, we describe a probabilistic approach to point based tracking and SFM. The thrust is to capture the uncertainty of tracking and SFM with the sampling method and integrate information through time with a probabilistic framework. Similarly this approach can be applied to PSFM. We would need the following items to make a probabilistic PSFM algorithm possible.

- We need to characterize the uncertainty of the homography for each patch. Since a homog-

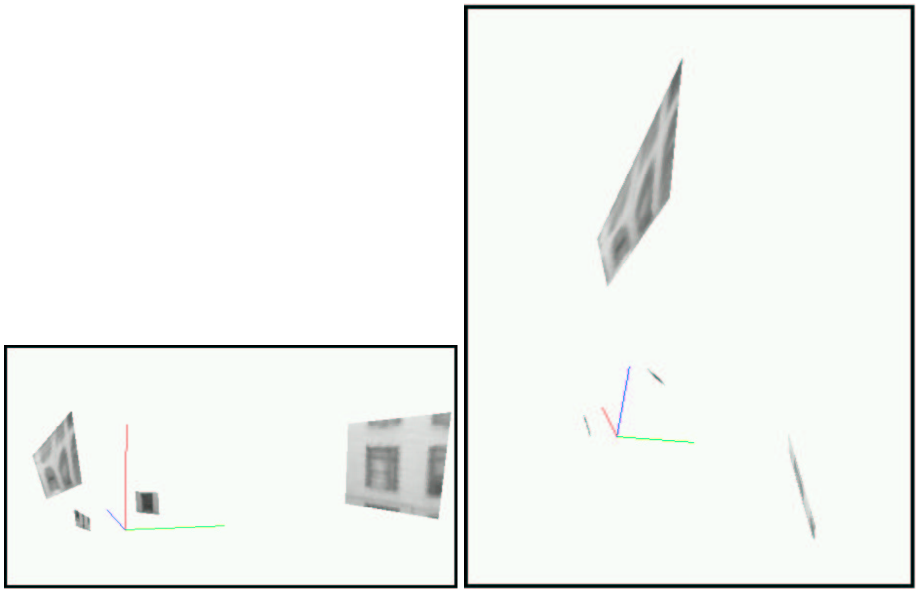


Figure 5.3: (a) reconstruction with Levenberg-Marquard (b) top view

raphy generally has 8 dimensions, it is not a trivial task. One possible way to capture the uncertainty is to use importance sampling. Namely samples are first generated around the homography returned by the tracking, then each sample is weighted by the SSD value. A fair sample set can be produced by resampling the weighted sample set. The resulting sample set represents the uncertainty the homography returned by the tracker.

- Given the sample sets for all the homographies, we solve the PSFM many times. The PSFM results form the sample set representing the uncertainty of current PSFM.
- Similar to the point feature case, the recovered planar structure and camera motion can be integrated through time.

To apply the probabilistic framework to the PSFM is much more expensive than the point case mainly because the planar scene has a much higher dimension space than point scenes. One can possibly combine these two to reduce the computation cost. For example, if the camera motion can be estimated from point features SFM, then the homography for each planar patch has a dimension of $8 - 5 = 3$. The sampling space is dramatically reduced when sampling the uncertainty of the homographies. But a sampling based PSFM remains a very expensive problem.

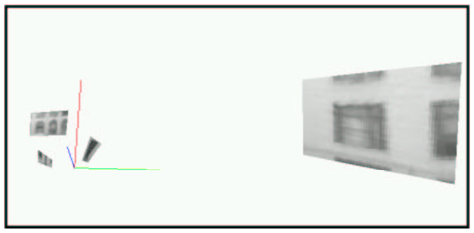
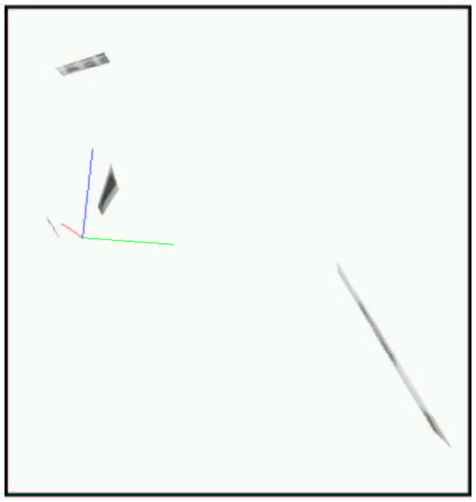



Figure 5.4: (a) reconstruction with gradient descent (b) top view

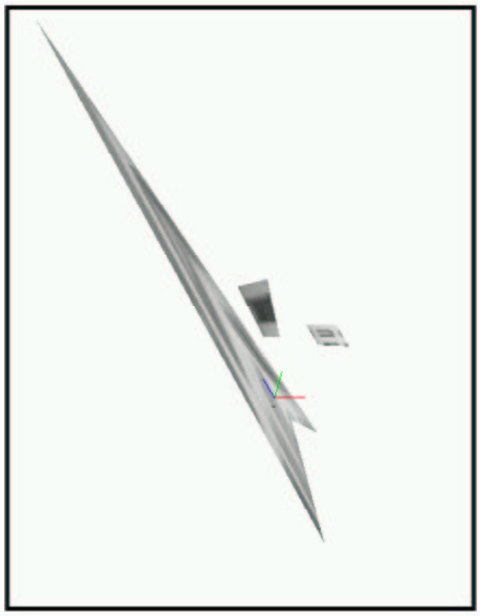


Figure 5.5: a bad reconstruction from the linear algorithm

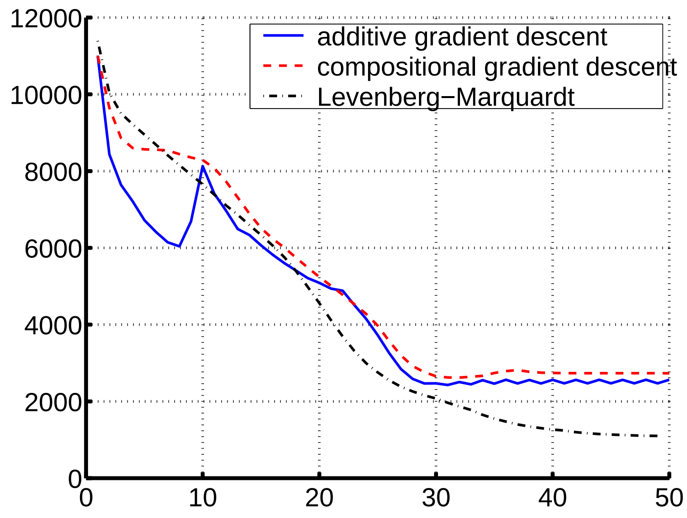


Figure 5.6: SSD error of three algorithms used in experiments

Chapter 6

Applications

6.1 Introduction

In this section, we show experimental results on two types of sequences: (1) omnidirectional image sequences captured on an omnidirectional camera mounted on a mobile robot and (2) image sequences captured with a hand-held camera. Most of the sequences are collected in outdoor urban environments. We want to demonstrate the following results:

(1) robust tracking and SFM for both omnidirectional camera and conventional camera. We show our integrated tracking and SFM algorithm can robustly track multiple features in the scene and perform SFM simultaneously.

(2) our probabilistic filtering scheme does propagate the uncertainty in both tracking and SFM through time. We show the uncertainty of the reconstruction largely decreases with time due to the increased baseline. However the uncertainty could increase if the current image contains degenerate situations. This is different from the Kalman filter case, in which the uncertainty decreases monotonically.

(3) with the reconstruction from SFM, the tracker is able to detect occlusion without tracking to wrong targets. We demonstrate that the tracker can track the target even the actual target is occluded. With the reconstruction the tracker consistently predicts the location of the occluded targets for very long duration of time and captures it once it appears again.

(4) sudden camera motion is naturally handled within the filtering scheme. In case of sudden camera motion, the uncertainty of tracking increases, which means the correspondence set is likely to contain more false matches. As explained earlier, by filtering with previous reconstruction, the false matches are pruned away naturally. The only special treatment for sudden camera motion is to detect it. Once the current image is detected as a sudden camera motion case, the tracking uncertainty is not fused with the SFM prediction, instead, it is used for current SFM directly. The

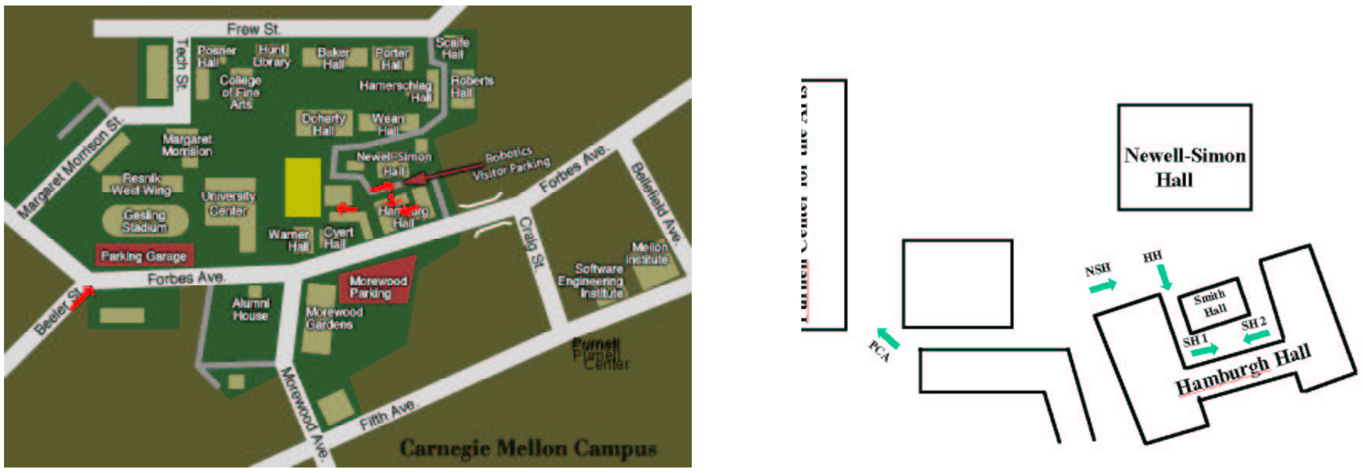
rest of the procedure is the same.

For most of the results, the green rectangles indicate the search region during normal tracking cycles. Red rectangles indicate the search region during recovery from occlusion or abrupt camera motion. The search region during recovery is usually larger than the search region during normal tracking cycle. A yellow rectangle indicates that the target is being tracked normally. If the yellow rectangle is not displayed, the tracker is during a recovery process. The red dots are sample prediction from the SFM. The hand-held camcorder we use to collect data is Sony DC-TRV20. The intrinsic parameters are calibrated with the Intel OpenCV library.

6.2 Robust Tracking and SFM

We show robust tracking and SFM results for several sequences. These sequences include one omnidirectional image sequence and five conventional camera image sequences collected with a hand-held camera. Only snapshots of the resulting sequences are displayed. For results on omnidirectional camera, the images contain eight perspective views unwarped from the original omnidirectional image and the feature tracking is performed on those virtual perspective images.

Since most of the image data we use here are taken around the campus of the Carnegie Mellon University, we enclose a campus map here and highlight the locations where the data collection was done.



(a) campus map (b) data collection pathes

Figure 6.1: The CMU campus map with data collection sites marked

6.2.1 Omnidirectional Sequence: Smith Hall 1 (SH1)

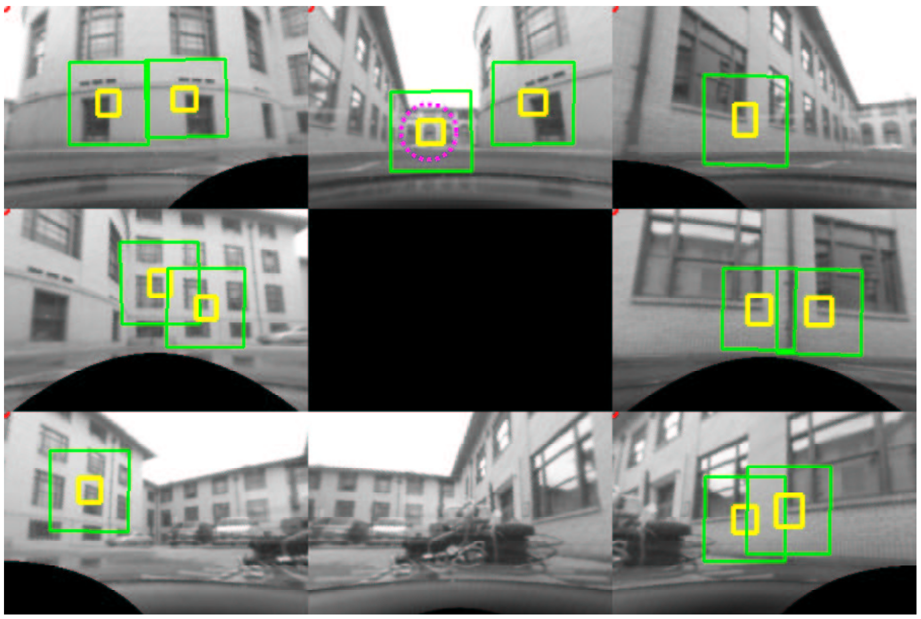
This sequence is collected with an omnidirectional camera mounted on a Pioneer AT mobile robot at the west end of the parking lot between the Smith Hall and Hamburg Hall of Carnegie Mellon University. The robot drives for about 10 meters and makes a point turn. The selected goal (a door) is about 40 meters away, marked with a dotted circle in frame 1. We manually occlude the target with photo editing at some frames. The scene is interesting in that there are several doors similar to the one selected, and we want to demonstrate that with the SFM predictions, the tracker is able to find the original one, instead of being confused by the others, despite large camera motion. Figure 6.2 shows several output images from the whole sequence.

11 features are initialized in frame 1. The target selected for the robot to drive is a door far away (about 40 meters), and is circled with dotted line. From frame 3, SFM starts to predict the target locations. In frame $11, 15$ the robot starts to rotate and the robust tracking and SFM follows it well. In frame 17 the target is artificially occluded and the occlusion is detected. (Note there is no yellow rectangle inside the search box) In frame $19, 21, 23$, the robot continues to rotate and the search region is set according to the prediction from SFM and it follows the occluded target consistently. The target appears again in frame 24, and the tracker is able to capture the original target despite the similar targets nearby.

6.2.2 Omnidirectional Sequence: Smith Hall 2 (SH2)

This sequence is collected with an omnidirectional camera mounted on a Pioneer AT mobile robot at the east end of the parking lot between the Smith Hall and Hamburg Hall of Carnegie Mellon University. The robot drives for about 25 meters and the selected goal is about 40 meters away. The scenario is that the robot first makes a left turn during which the goal is occluded. Then it makes a right turn and the goal appears in the image again. The actual distance driven with the goal occluded is about 15 meters. Figure 6.3 shows several output images from the whole sequence.

In frame 1, 12 features are initialized. The target selected for the robot to drive is a door far away (about 40 meters), and is circled with dotted line. Frame $10, 25$ show the usual multiple feature tracking with template matching method. The SFM is also performed during the initial driving period, but it is not used for prediction until some criteria about the reconstruction is met. In the example shown here, the SFM prediction begins when the average reprojection error is less one pixel. After driving for a certain distance, this criteria is met and the SFM prediction starts (the red dots indicate the SFM prediction), as in frame 26. Frame $35, 40, 70$ are integrated tracking and SFM. The robot makes a left turn and the tracking and SFM prediction match very well. In frame 80, the distant goal is occluded by the building in front. The occlusion is detected and a search is performed in an enlarged region indicated by the red rectangle. In frame $125, 200, 301$, the robot


(a) Frame 1

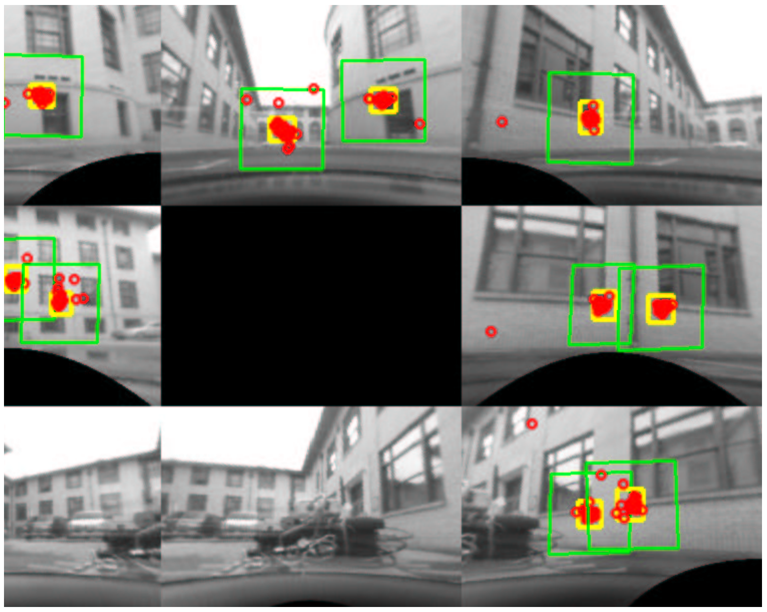
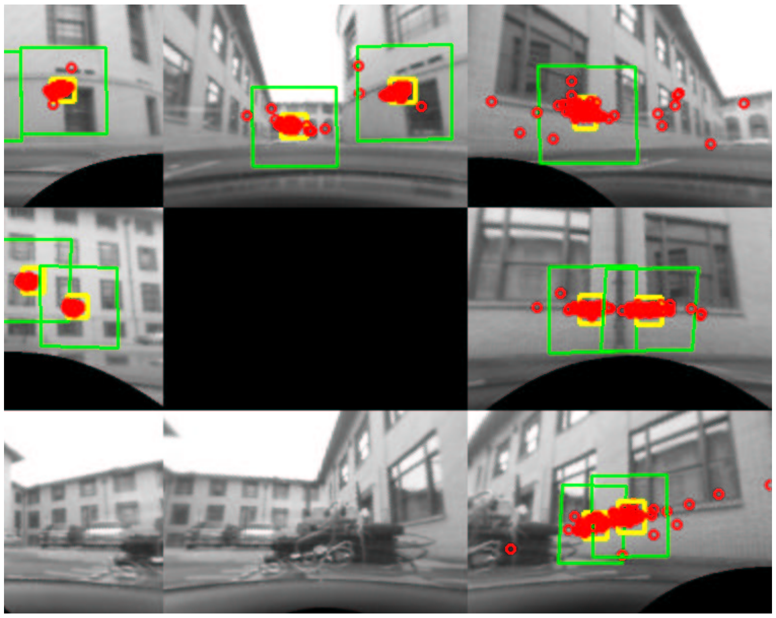

(b) Frame 3

Figure 6.2: Tracking results for omnidirectional sequence SH1


(c) Frame 11

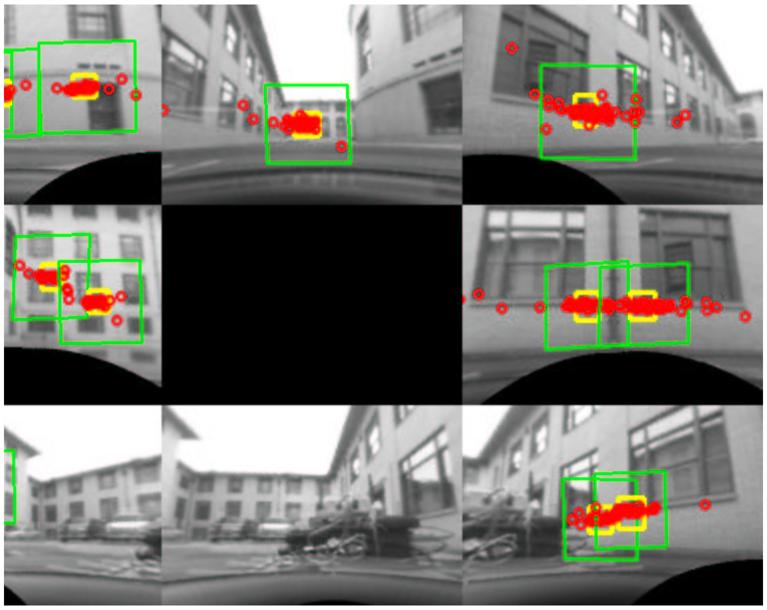
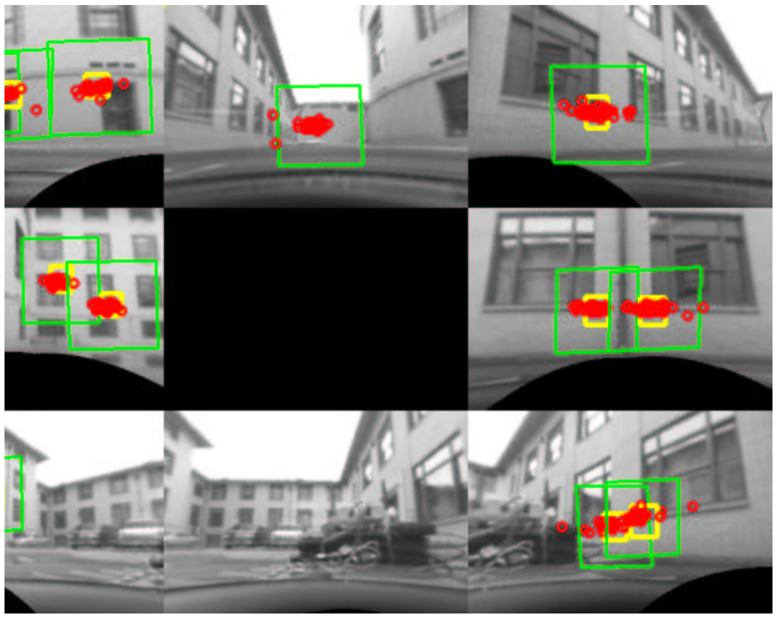

(d) Frame 15

Figure 6.2: Tracking results for omnidirectional sequence SH1


(e) Frame 17

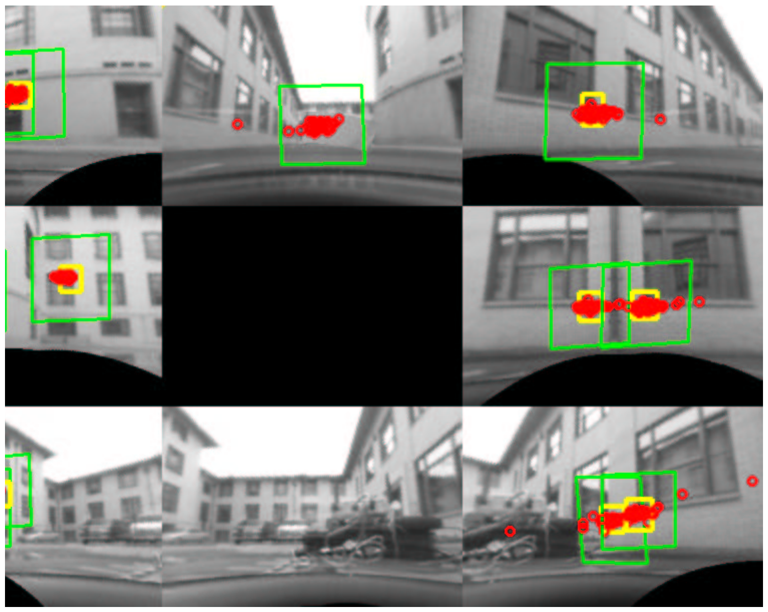
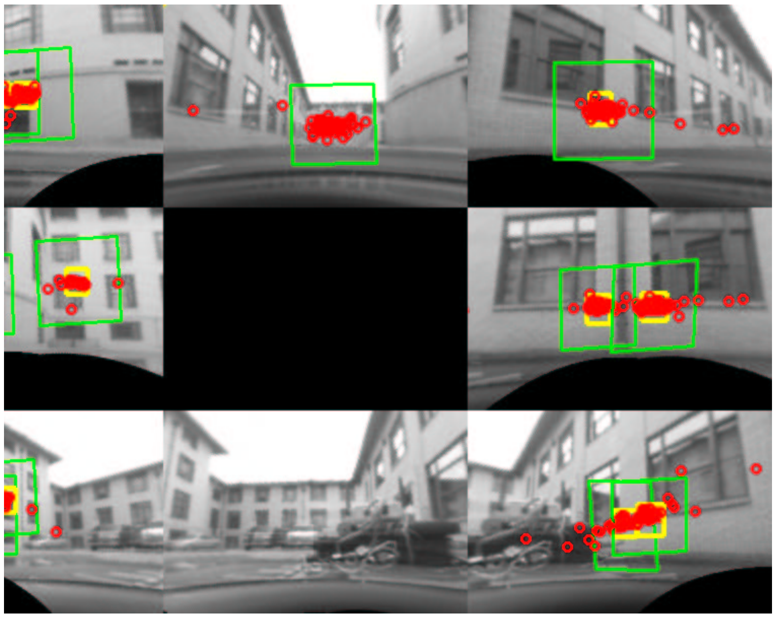
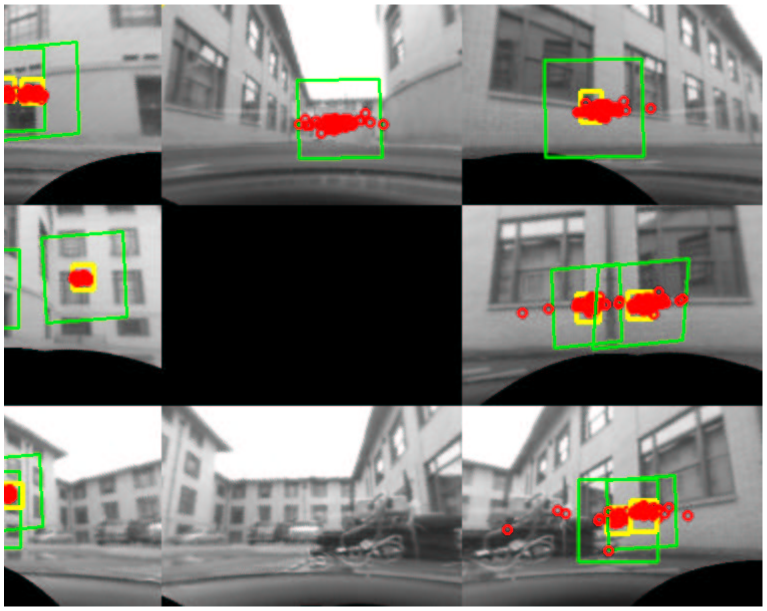

(f) Frame 19

Figure 6.2: Tracking results for omnidirectional sequence SH1

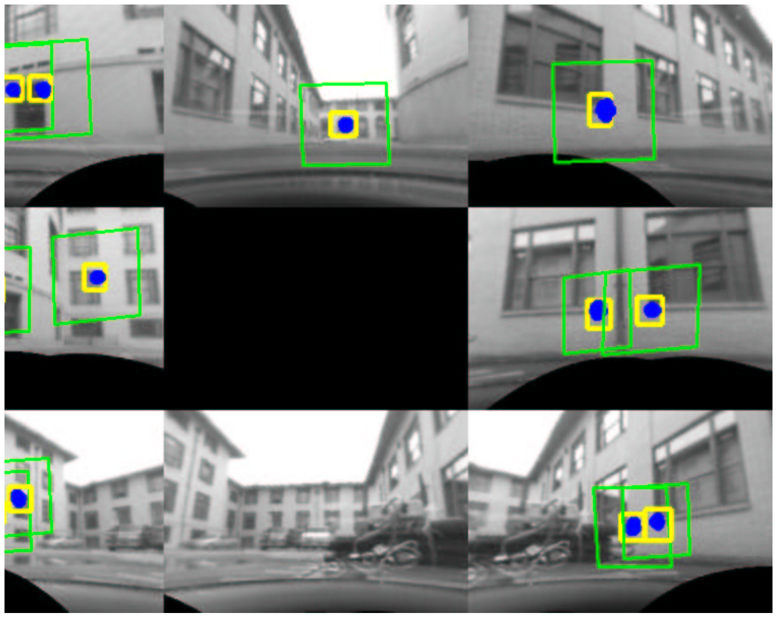


(g) Frame 21



(h) Frame 23

Figure 6.2: Tracking results for omnidirectional sequence SH1


(i) Frame 24

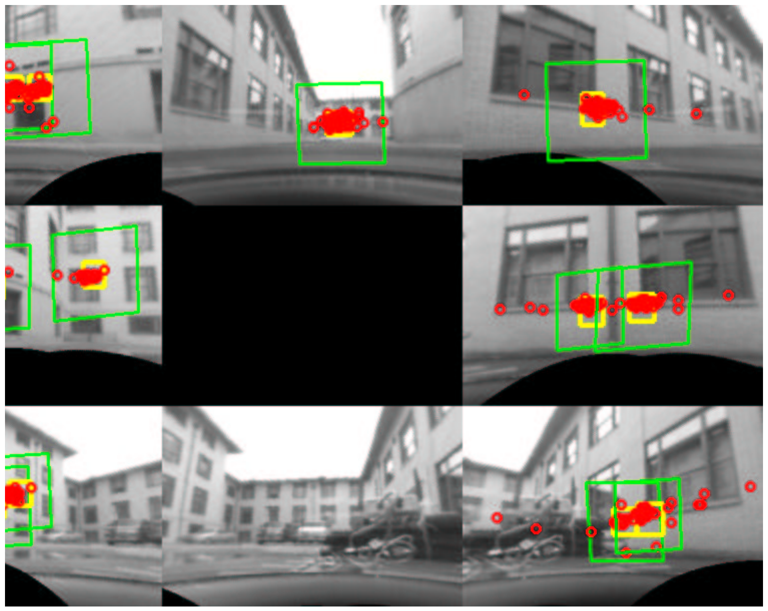
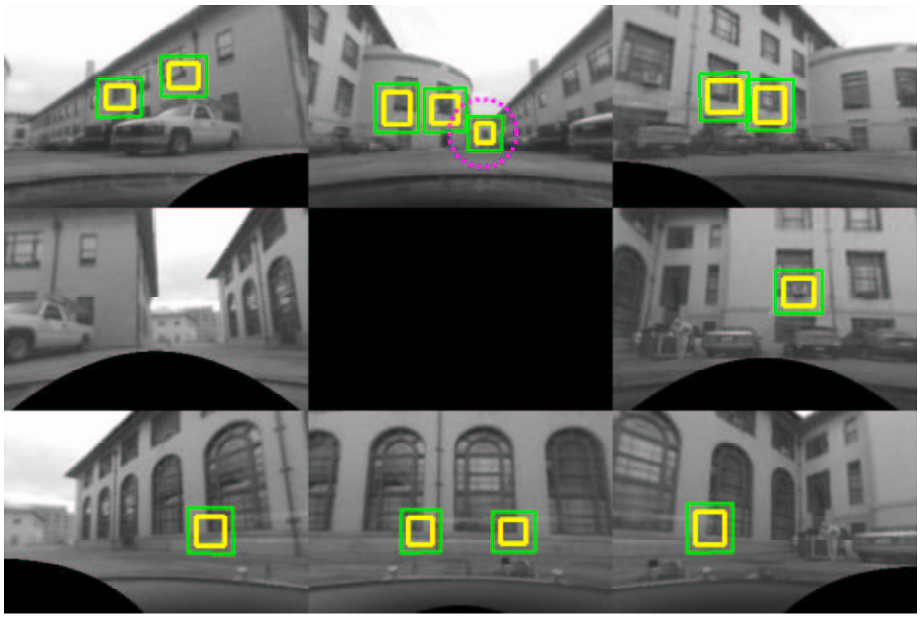

(j) Frame 25

Figure 6.2: Tracking results for omnidirectional sequence SH1


(a) Frame 1

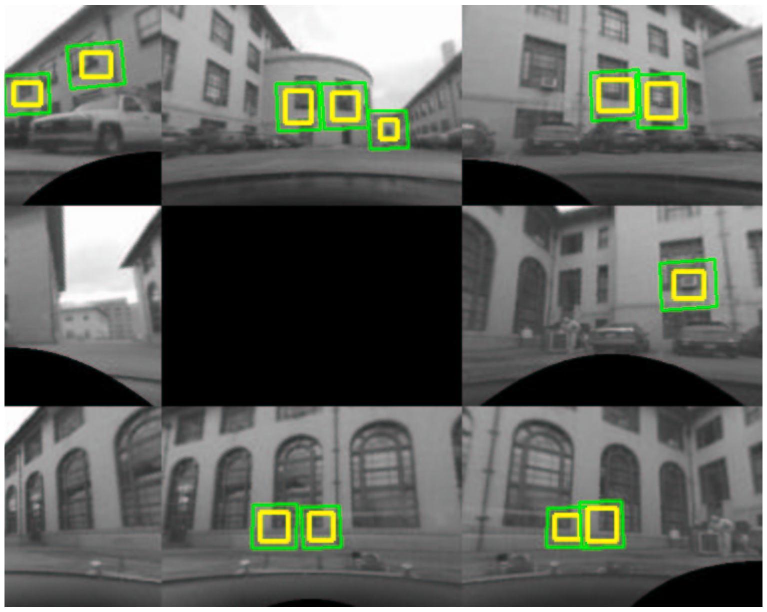
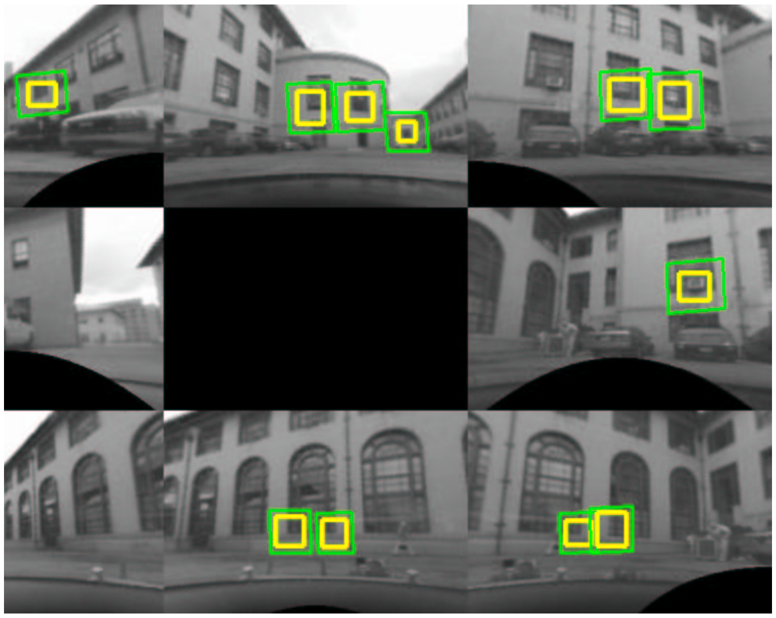

(b) Frame 10

Figure 6.3: Tracking results for omnidirectional sequence SH2


(c) Frame 25

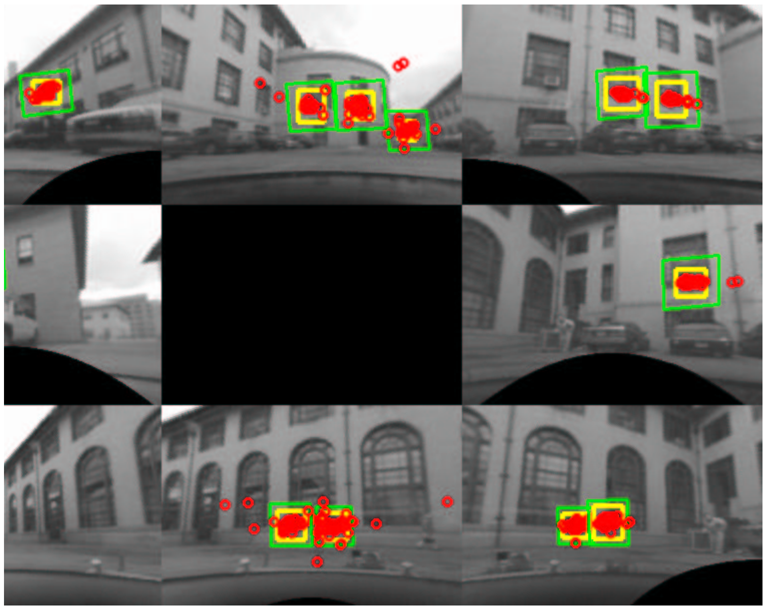
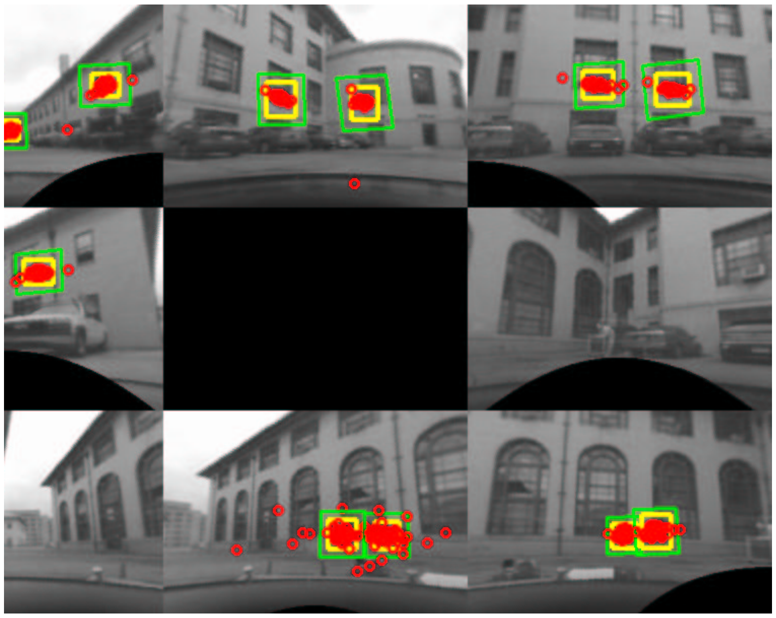
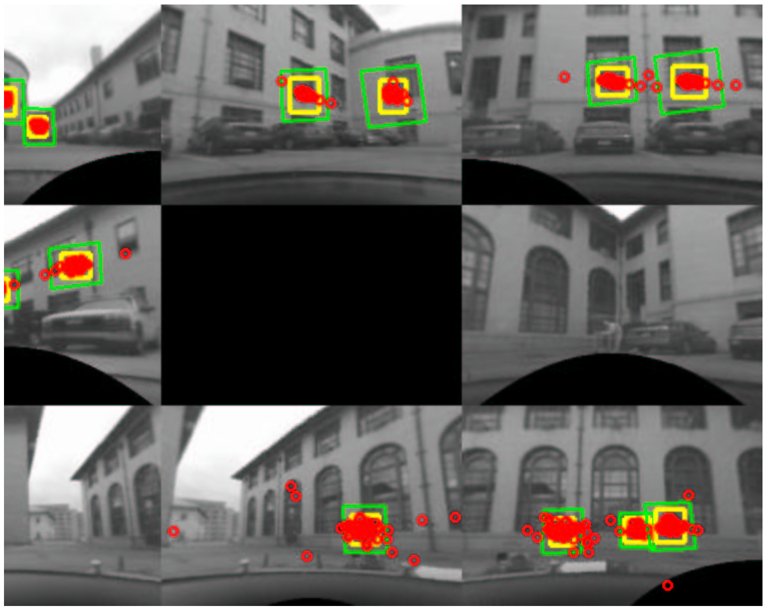

(d) Frame 26

Figure 6.3: Tracking results for omnidirectional sequence SH2

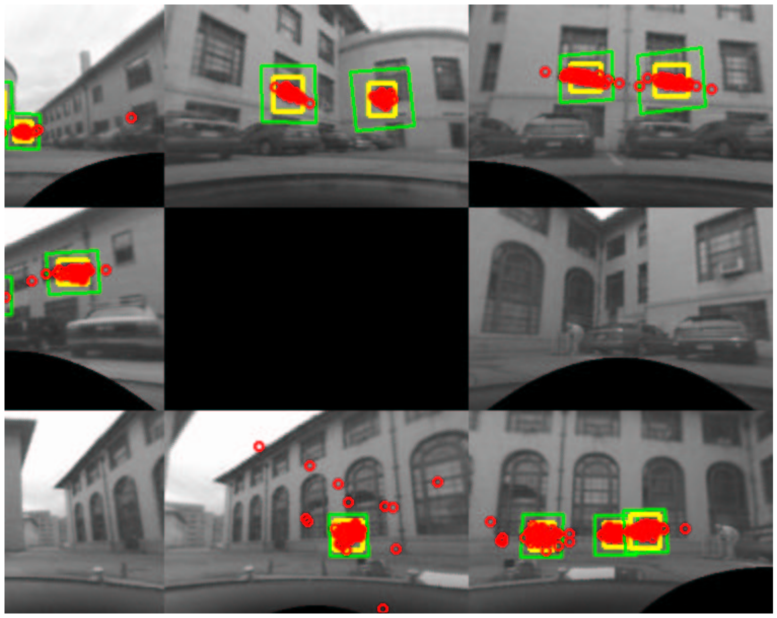


(e) Frame 35



(f) Frame 40

Figure 6.3: Tracking results for omnidirectional sequence SH2


(g) Frame 70

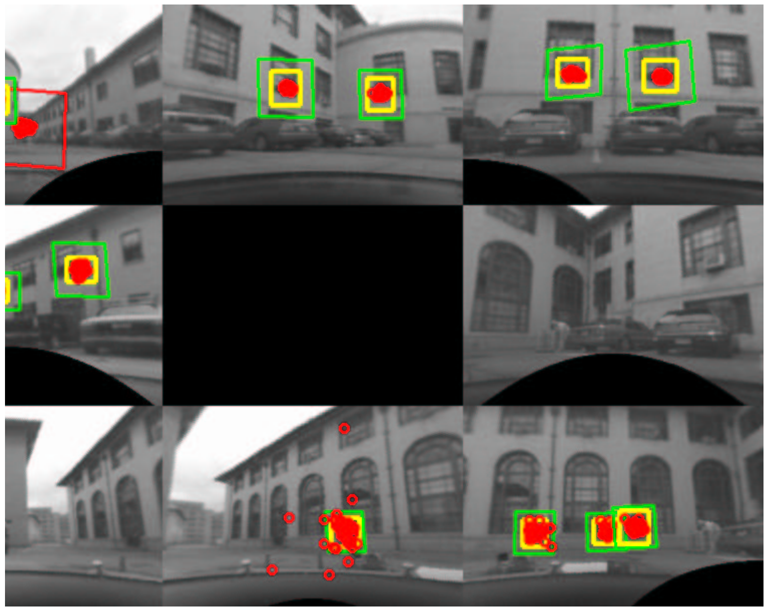
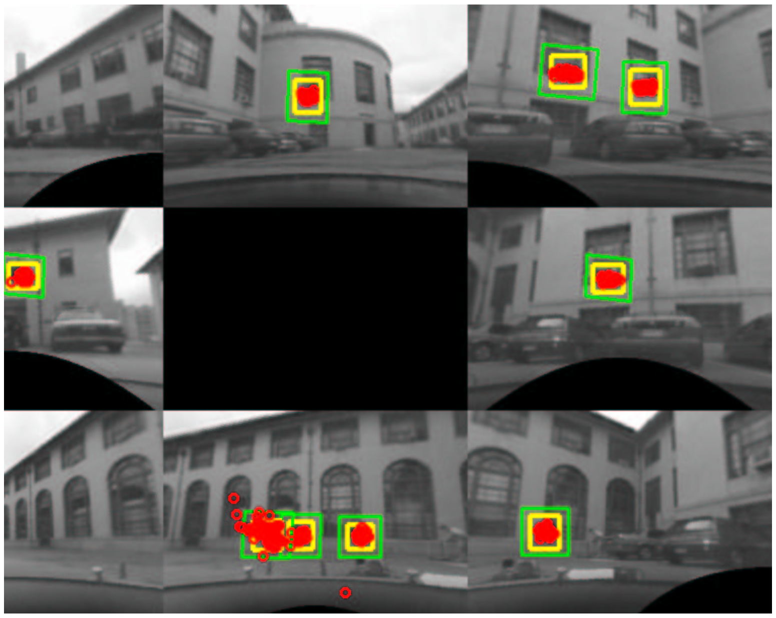

(h) Frame 80

Figure 6.3: Tracking results for omnidirectional sequence SH2


(i) Frame 125

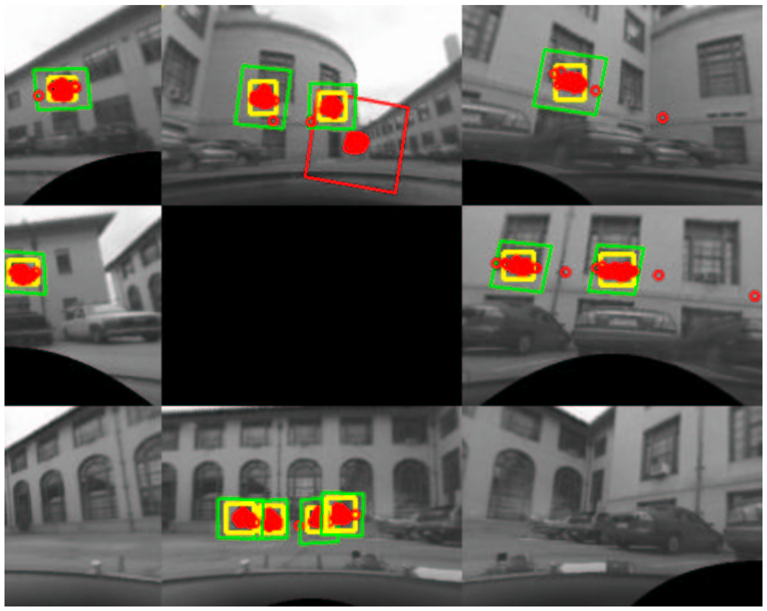
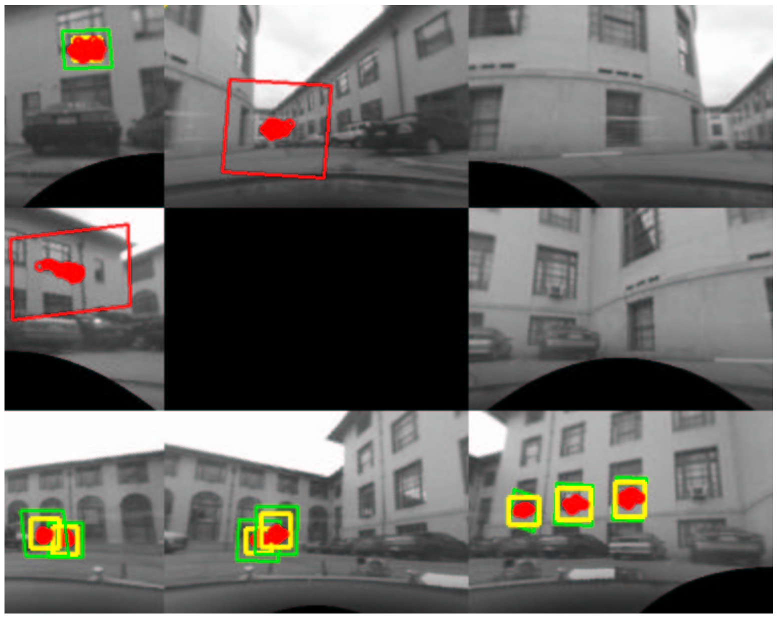
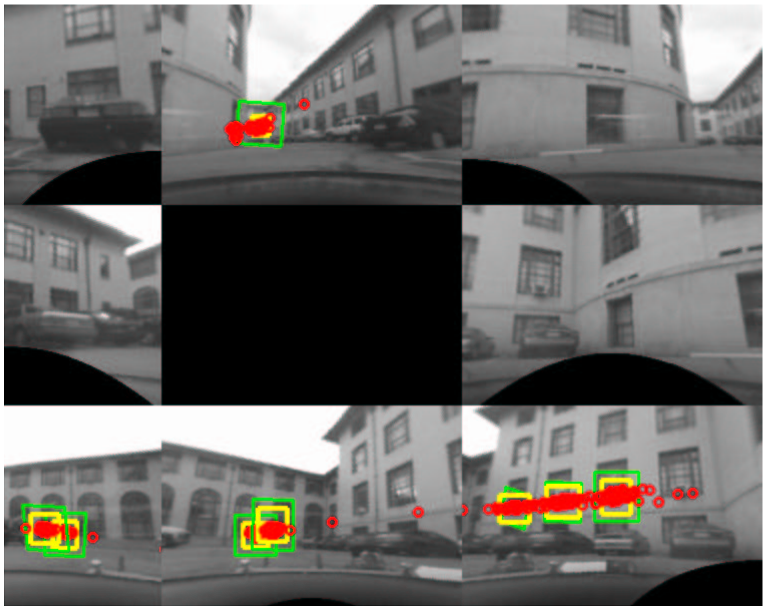

(j) Frame 200

Figure 6.3: Tracking results for omnidirectional sequence SH2

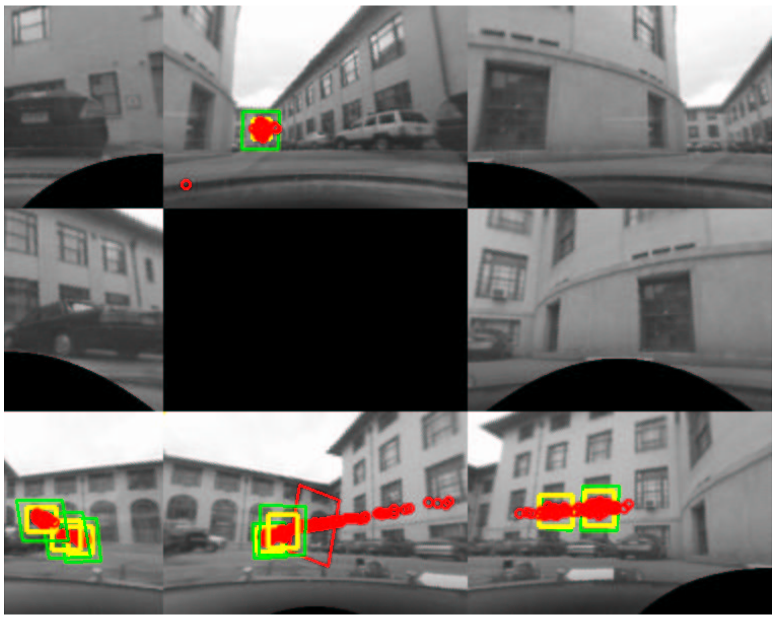


(k) Frame 301



(l) Frame 305

Figure 6.3: Tracking results for omnidirectional sequence SH2



(m) Frame 340

Figure 6.3: Tracking results for omnidirectional sequence SH2

keeps driving forward and turns right. Even when the goal is still occluded, the SFM predictions can consistently predict the location of the goal and guide the tracker to search for it. Note that the amount of driving with the goal occluded is about 15 meters, which is a fairly long distance and the location of the goal moves substantially in the image plane. At frame 305, the goal appears again in the image, and the tracker is able to recapture it with the guidance from the SFM.

6.2.3 Hand-Held Camera Sequence: Hamburg Hall (HH)

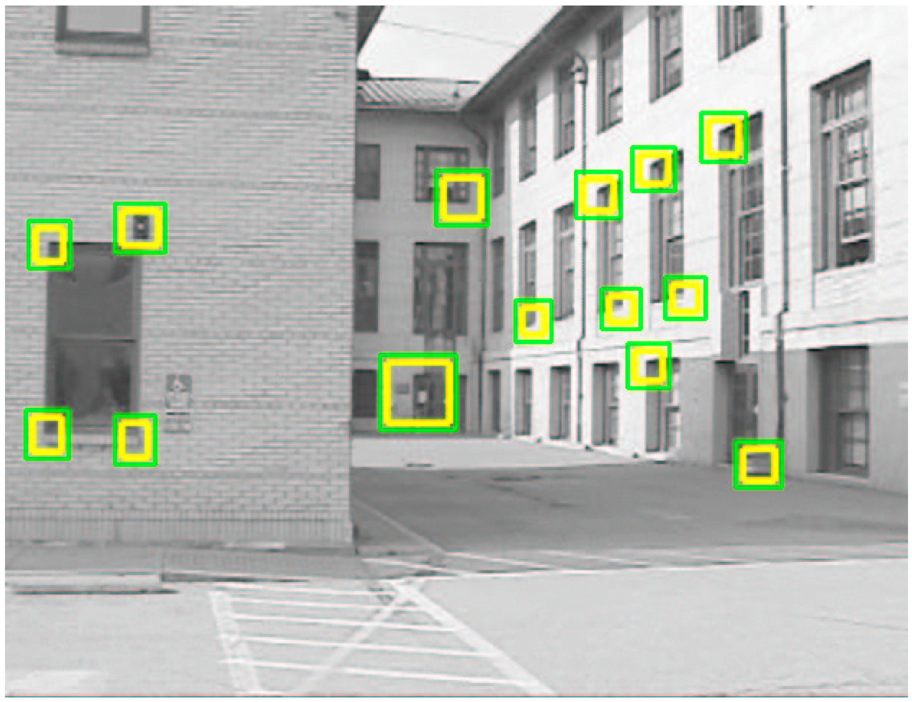
This sequence was collected with a hand-held camera at the parking area between the Smith Hall and Newell-Simon Hall, facing Hamburg Hall. The camera was pointing in the direction of Hamburg Hall while moving forward about 10 meters. The entrance of Hall is selected as the goal and it is about 40 meters away from my initial position. The path is made curvy so that the entrance is occluded for most of the sequence. We show that our integrated tracking and SFM is able to track it consistently. Figure 6.4 shows several output images from the whole sequence.

In frame 1, 14 features are initialized. The target selected is the entrance to Hamburg Hall approximately 40 meters away. Frame 50 shows the usual integrated tracking and SFM. After moving about 5 meters, the goal is occluded, as shown in frame 100. Frame $150, 200, 250, 300, 350$ show the tracking through the occlusion. In frame $403, 409$, the occluded targets appear in the image again, with the SFM prediction, the tracker is able to recover them accurately. Notice that the occluded features have far away from their original locations. Without the SFM prediction, the tracker has to search over a large region and may fail to recover the original targets.

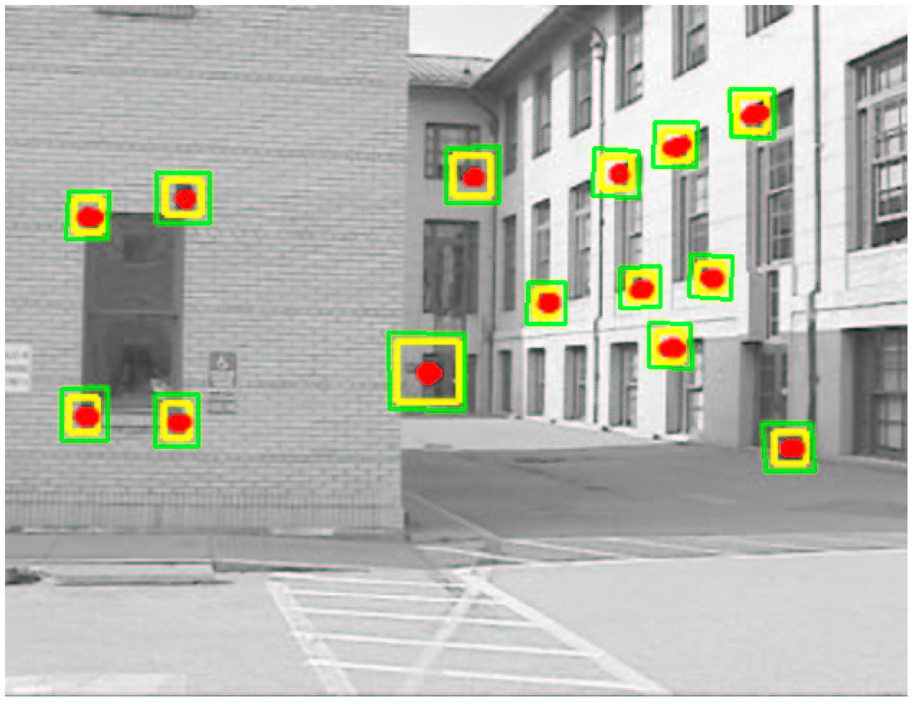
6.2.4 Hand-Held Camera Sequence: Newell-Simon Hall (NSH)

This sequence is collected with a hand-held camera at the parking area between the Smith Hall and Newell-Simon Hall, facing Newell-Simon Hall moving laterally for 15 meters with the camera pointing to the direction of Newell-Simon Hall. The entrance of Newell-Simon Hall is the main goal and it is about 40 meters away from my initial position. The parked automobiles serve as obstructing objects and occlude the goal for an extended period of time. We show our integrated tracking and SFM is able to track the main entrance consistently. Figure 6.5 shows several output images from the whole sequence.

In frame 1, 12 features are initialized. The goal is selected at the entrance to Newell-Simon Hall approximately 40 meters away. Frame $10, 40, 60$ show the usual integrated tracking and SFM. After moving about 5 meters, the goal is occluded by the truck, as shown in frame 70. The occlusion is detected and a search over larger region is started. Frame $90, 120, 160, 190, 230, 260, 280$ show the tracking through the occlusion. Due to abrupt camera motion, image quality or going out of the field of view, some features are lost during the tracking, but they are often recovered promptly. Frame

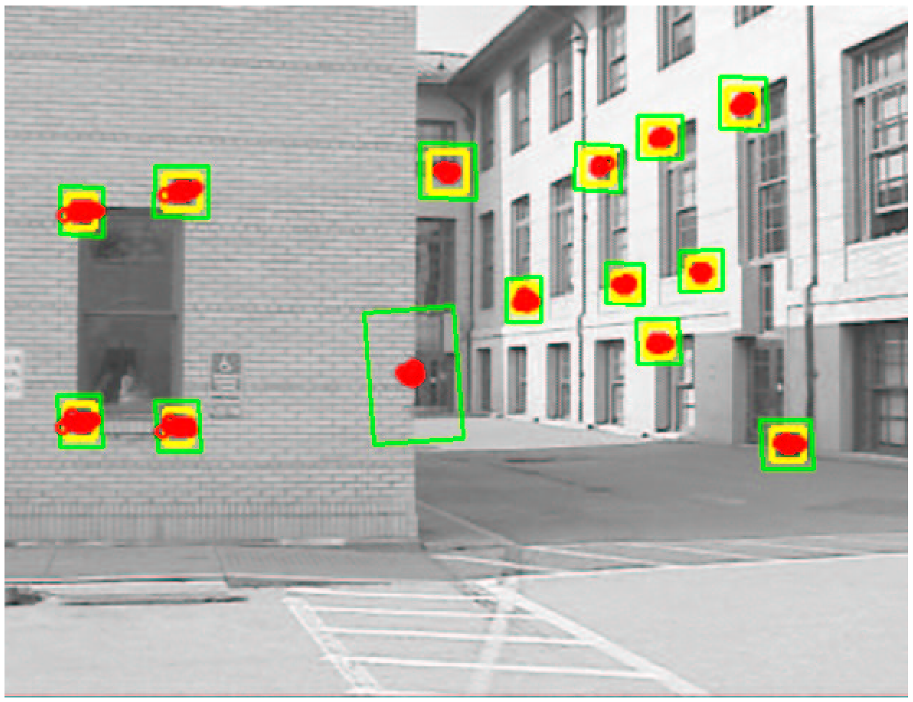


(a) Frame 1

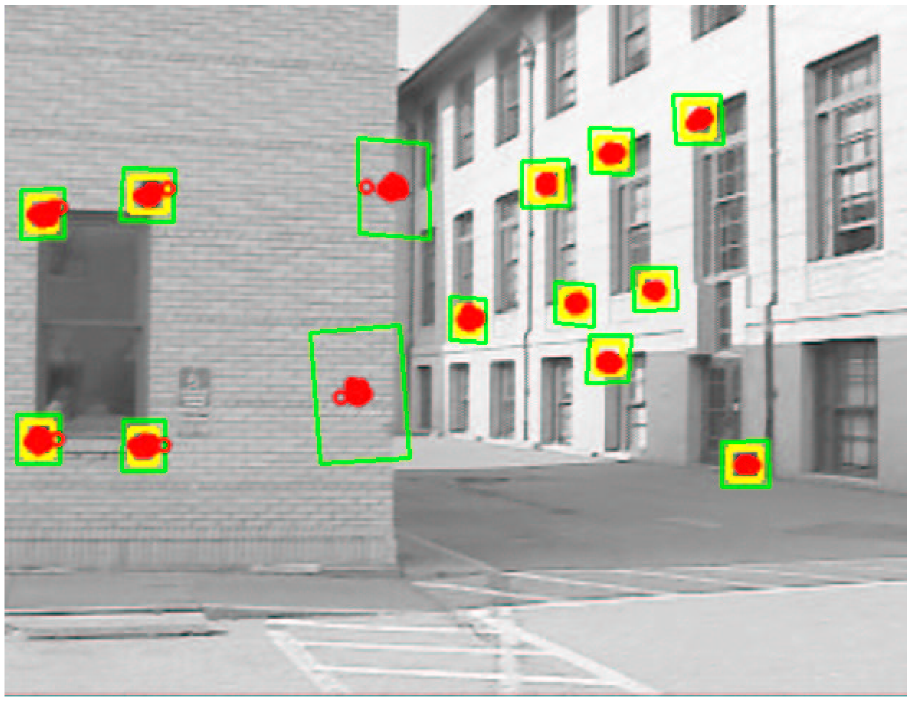


(b) Frame 50

Figure 6.4: Tracking results for hand-held sequence HH

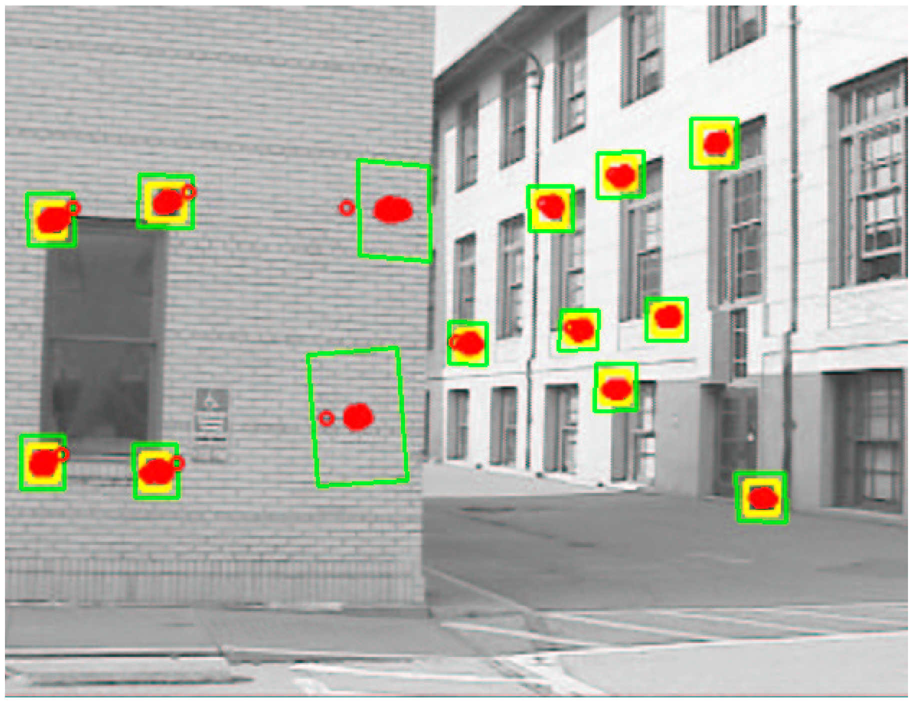


(c) Frame 100

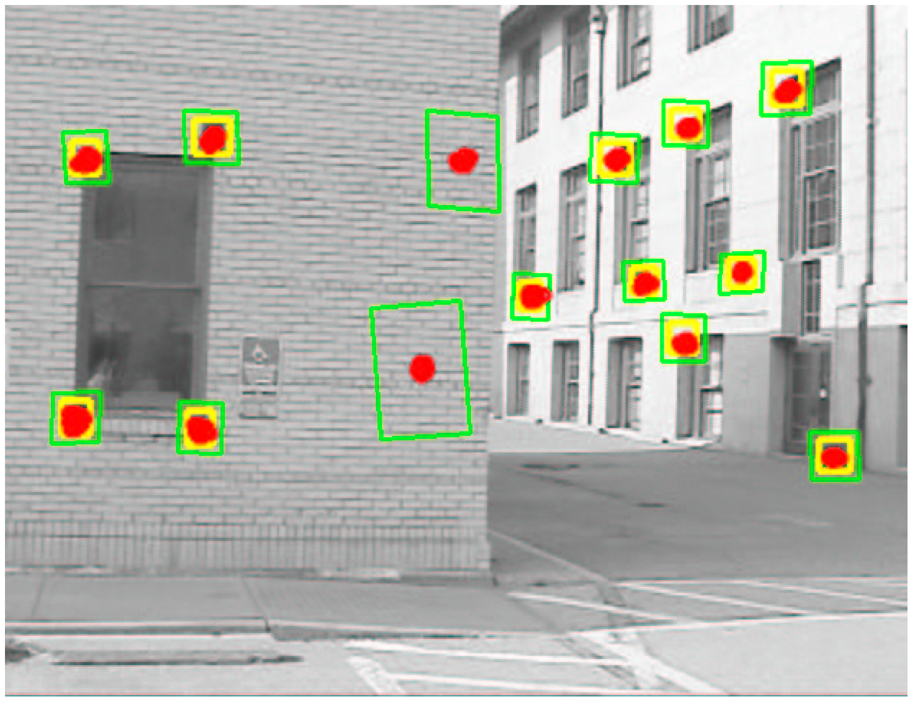


(d) Frame 150

Figure 6.4: Tracking results for hand-held sequence HH

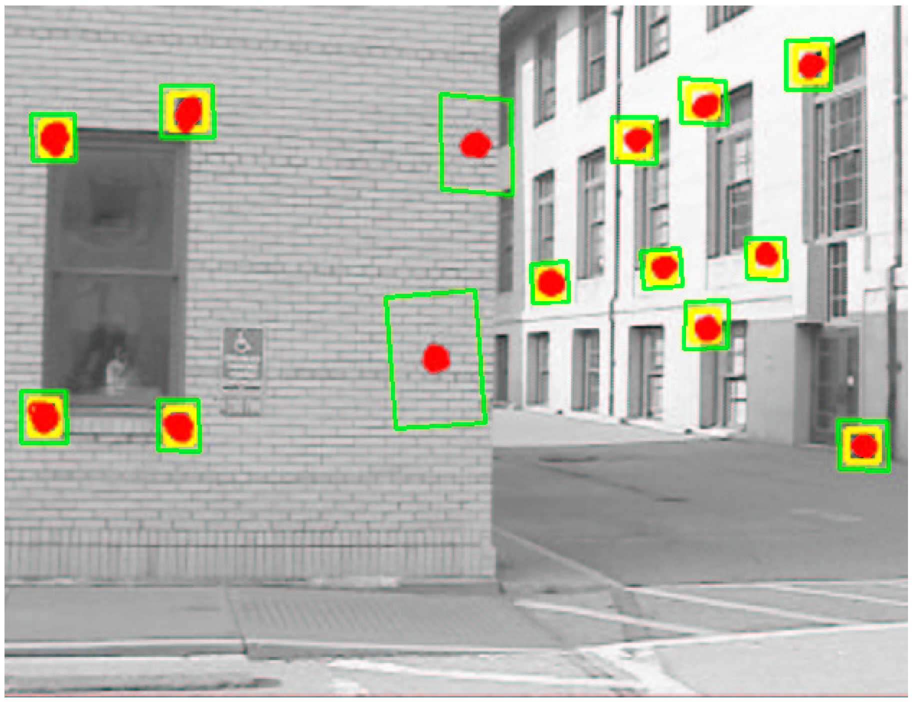


(e) Frame 200

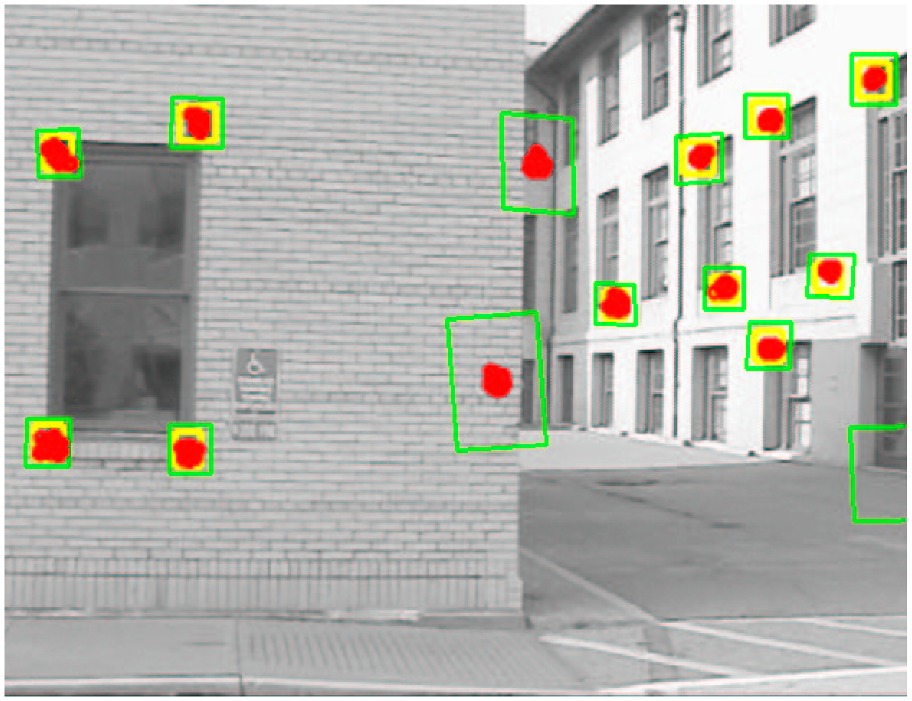


(f) Frame 250

Figure 6.4: Tracking results for hand-held sequence of HH

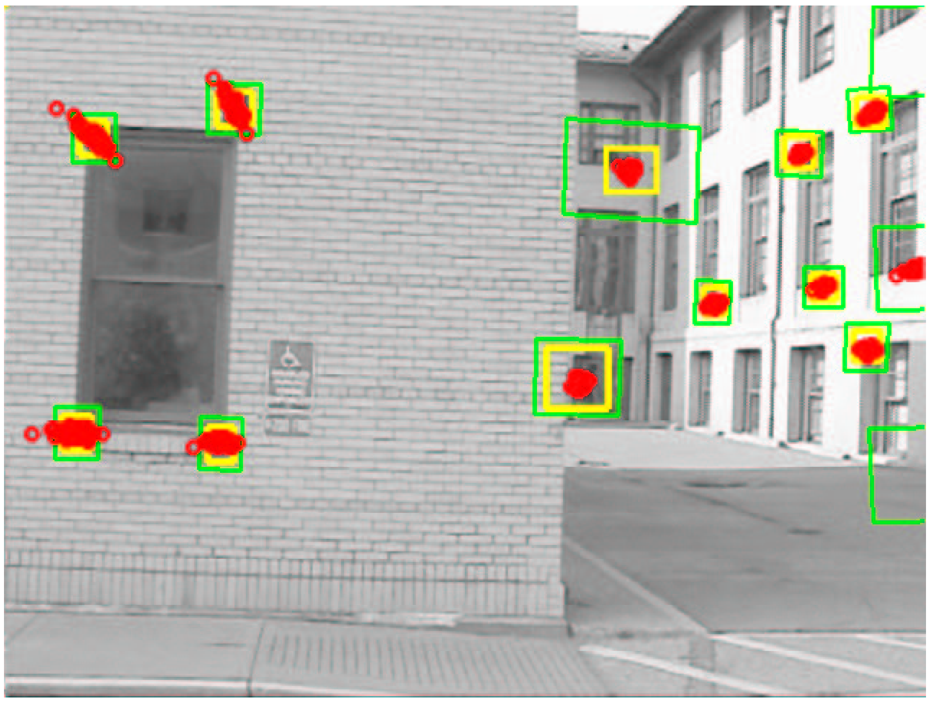


(g) Frame 300

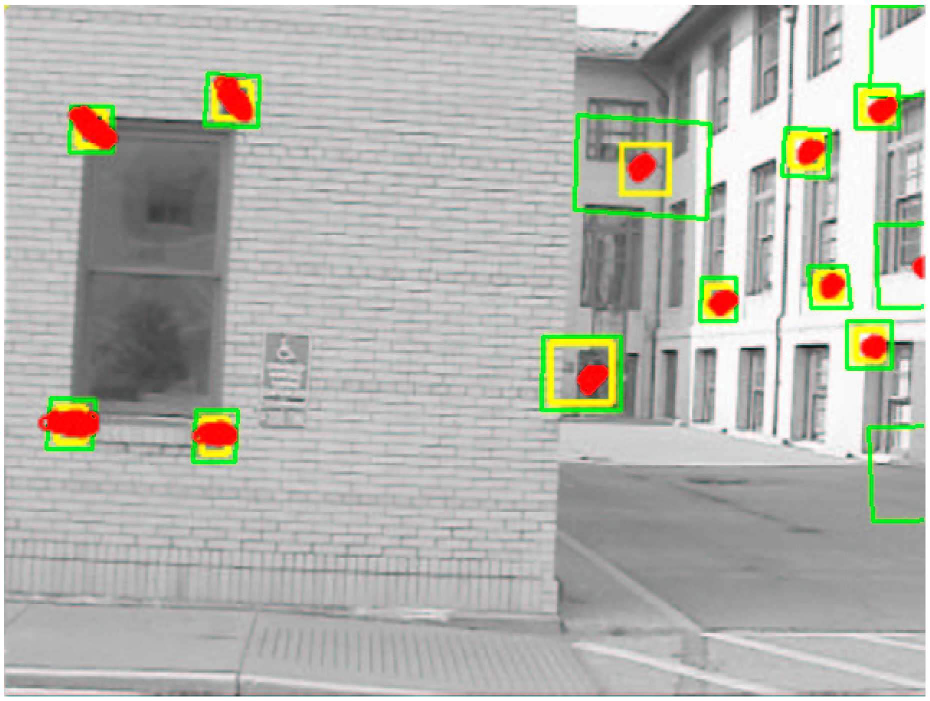


(h) Frame 350

Figure 6.4: Tracking results for hand-held sequence HH

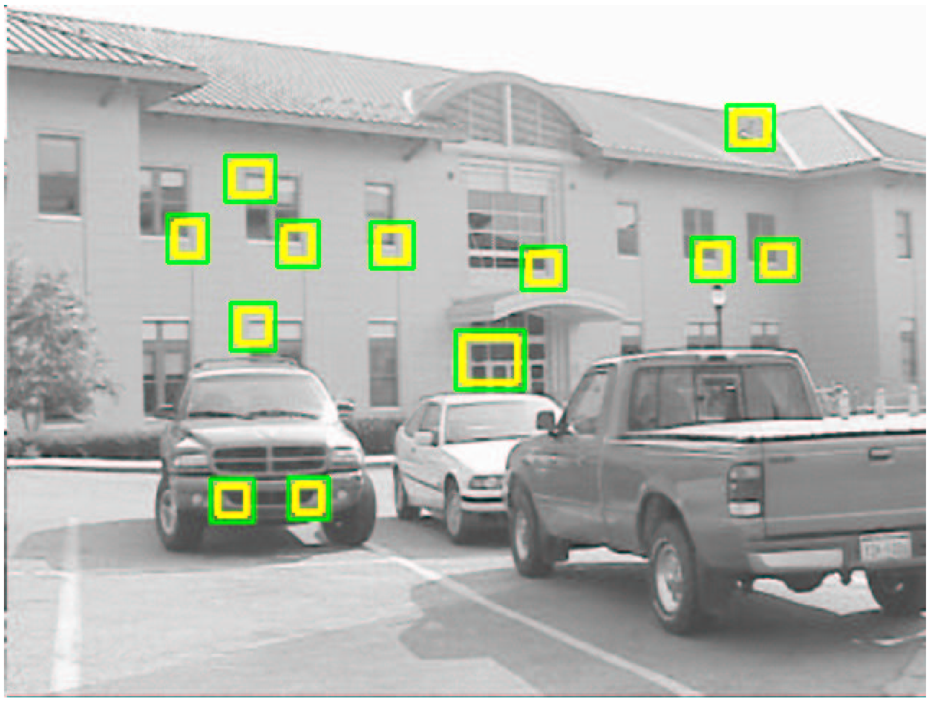


(i) Frame 403

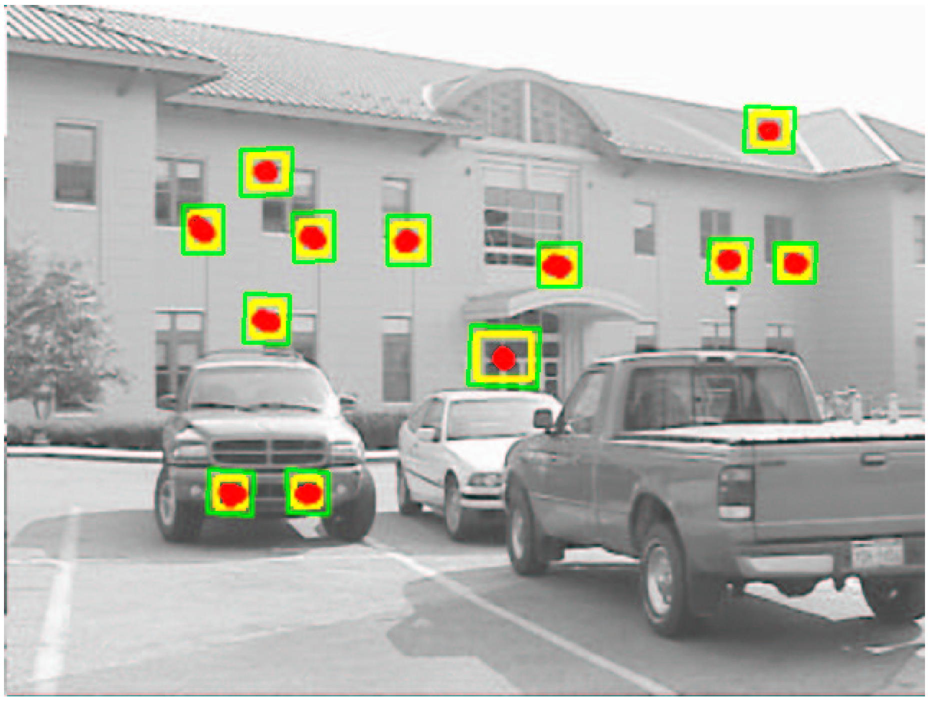


(j) Frame 409

Figure 6.4: Tracking results for hand-held sequence HH

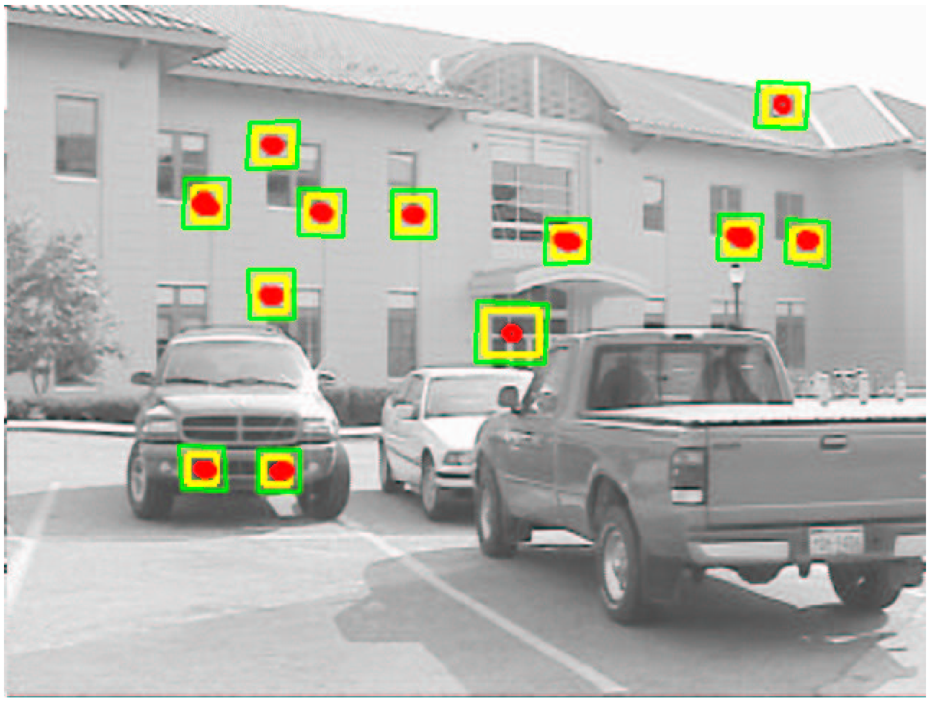


(a) Frame 1

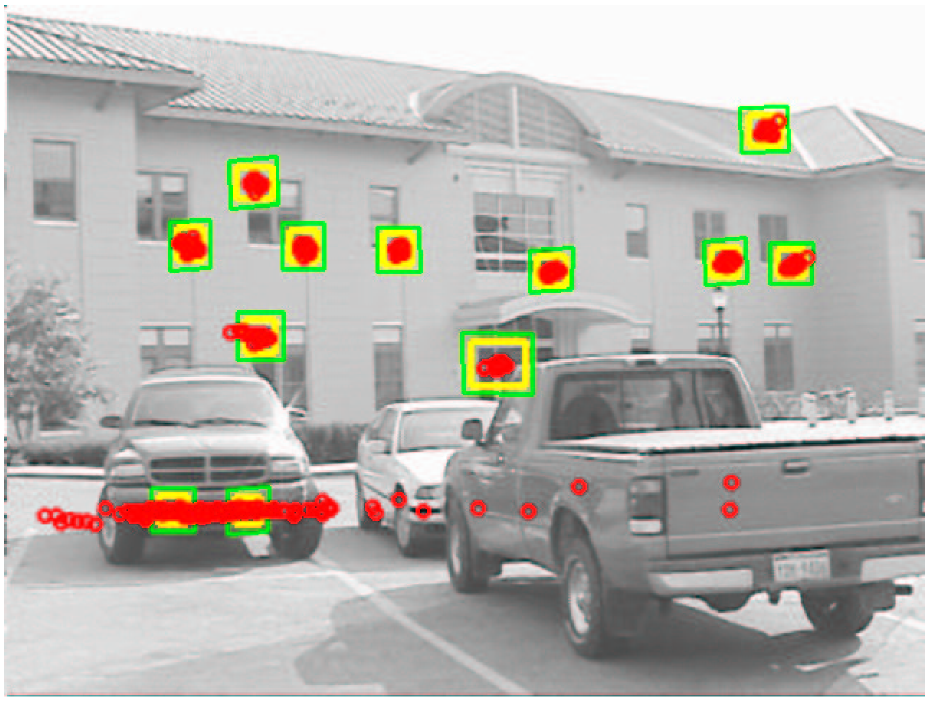


(b) Frame 10

Figure 6.5: Tracking results for hand-held sequence NSH

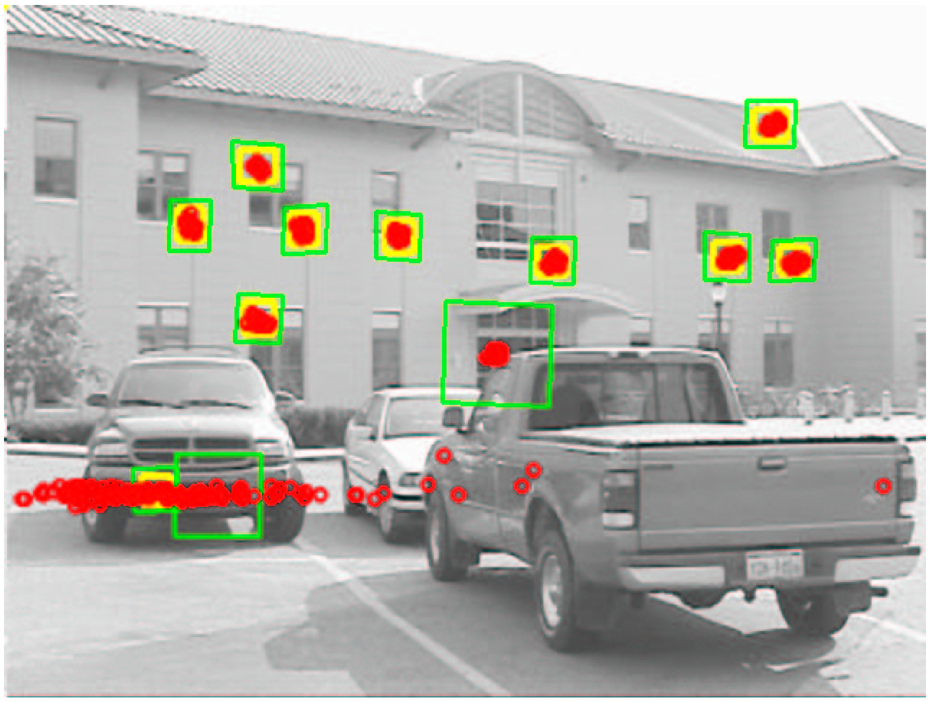


(c) Frame 40



(d) Frame 60

Figure 6.5: Tracking results for hand-held sequence NSH


(e) Frame 70

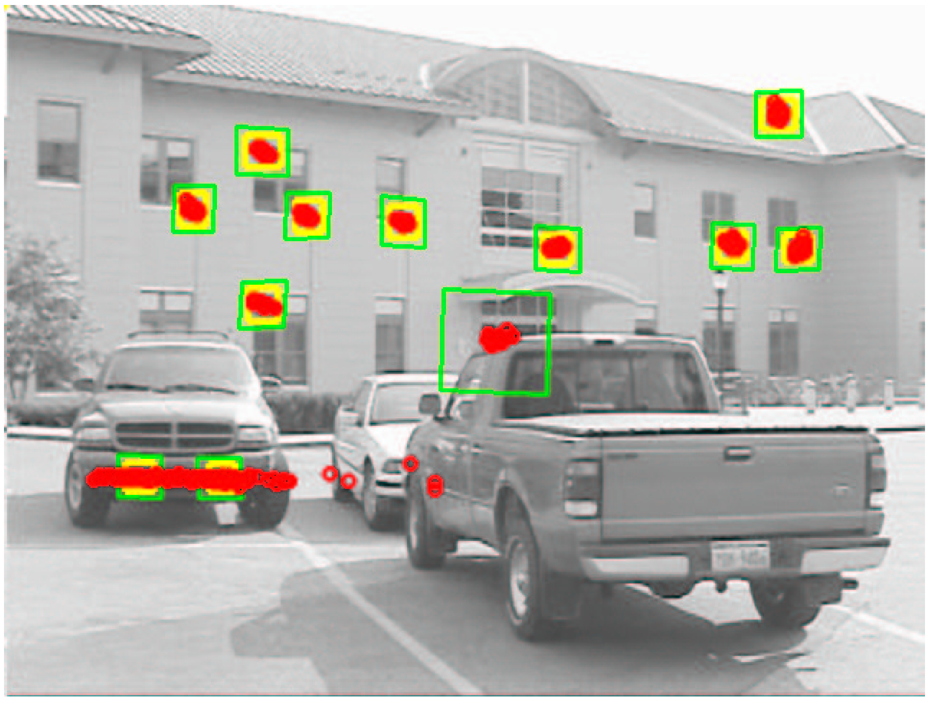
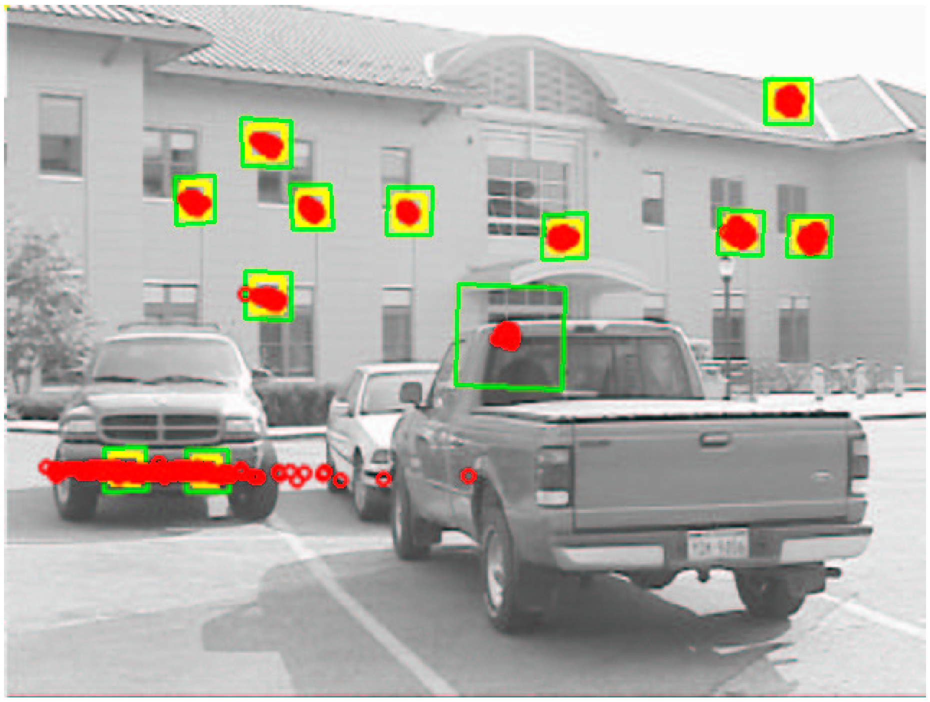
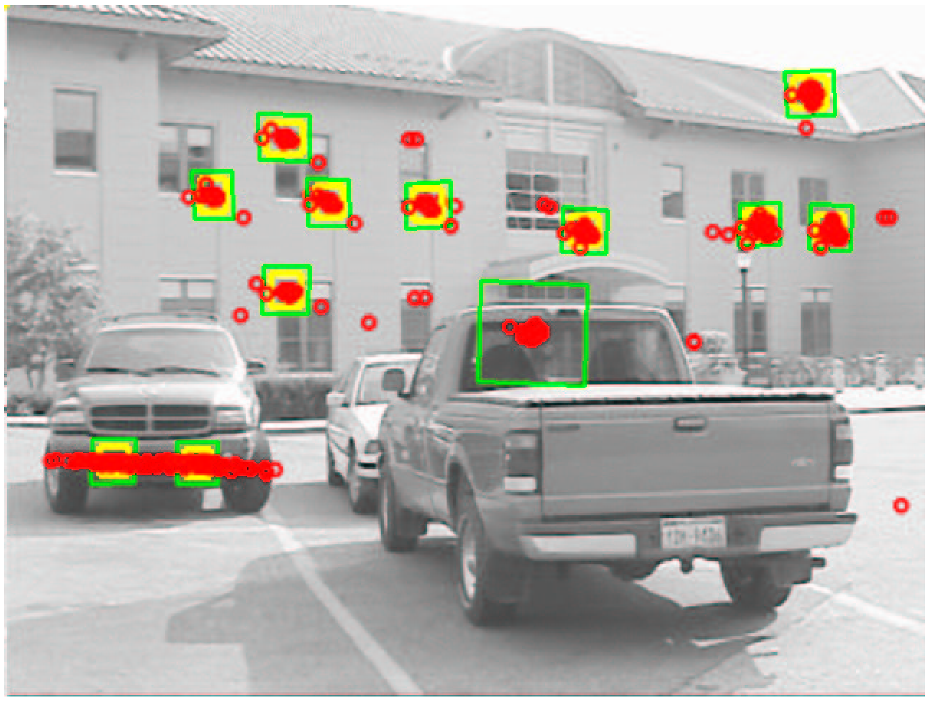

(f) Frame 90

Figure 6.5: Tracking results for hand-held sequence NSH

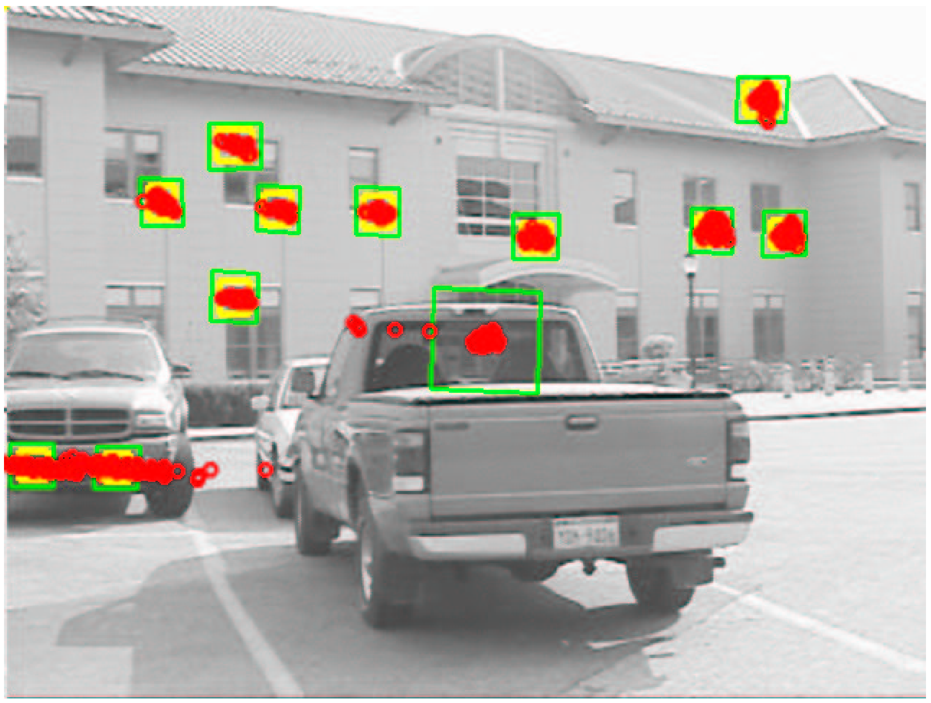


(g) Frame 120

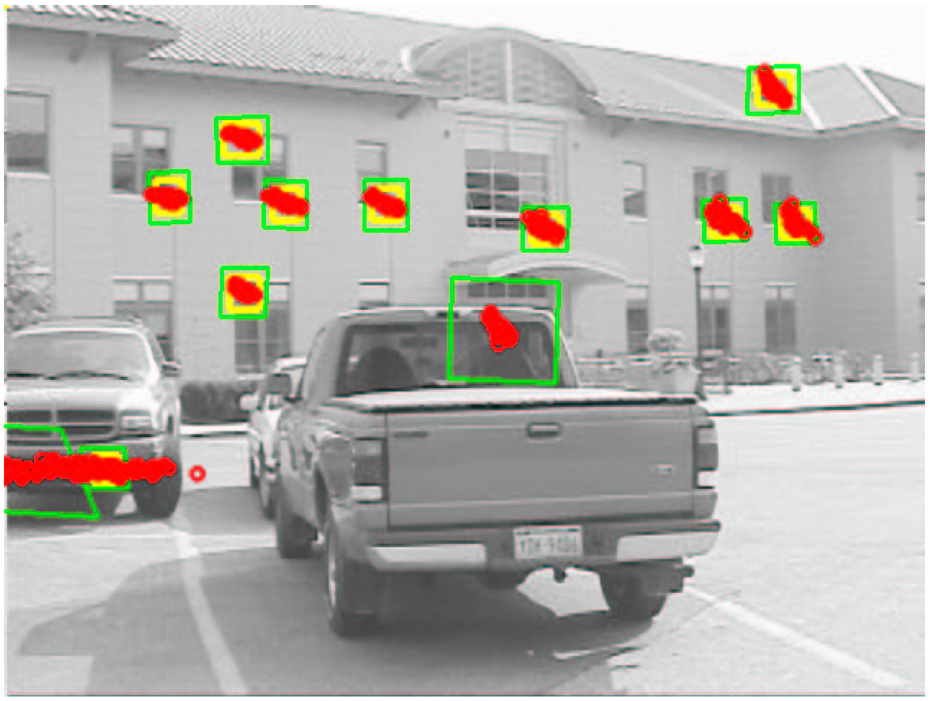


(h) Frame 160

Figure 6.5: Tracking results for hand-held sequence NSH

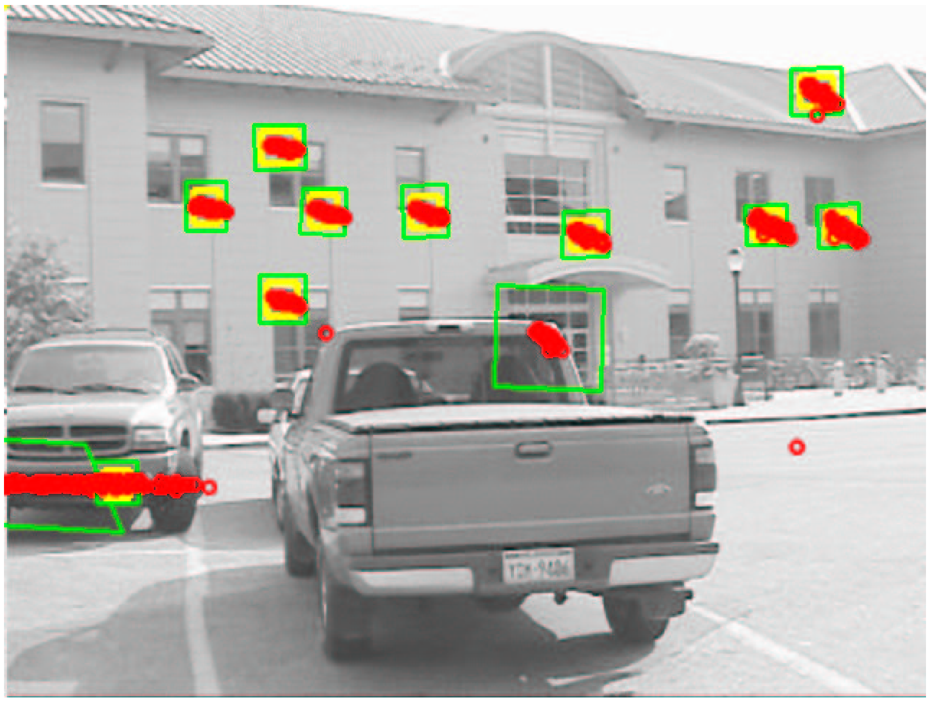


(i) Frame 190

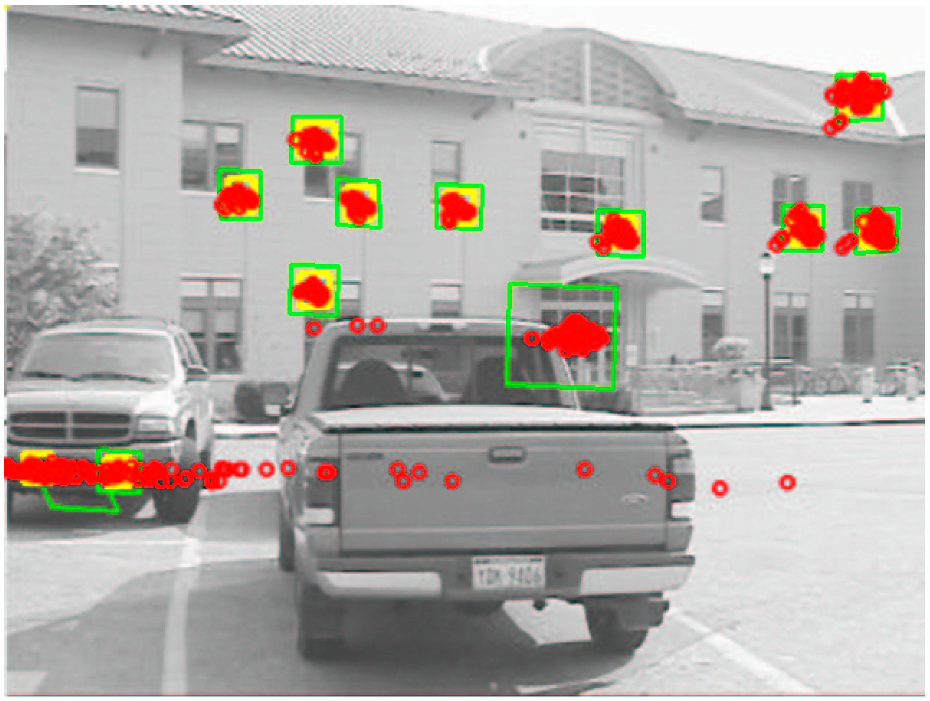


(j) Frame 230

Figure 6.5: Tracking results for hand-held sequence NSH

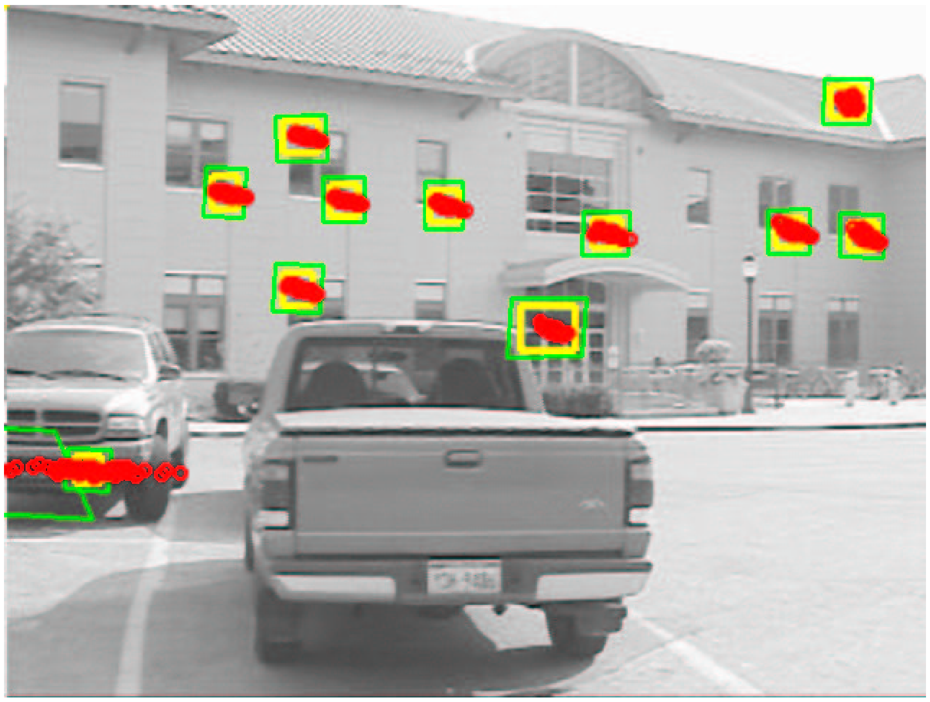


(k) Frame 260



(l) Frame 280

Figure 6.5: Tracking results for hand-held sequence NSH



(m) Frame 290

Figure 6.5: Tracking results for hand-held sequence of NSH

290 shows the target is reacquired once it appears again in the image.

6.2.5 Hand-Held Camera Sequence: Purnell Center for the Arts (PCA)

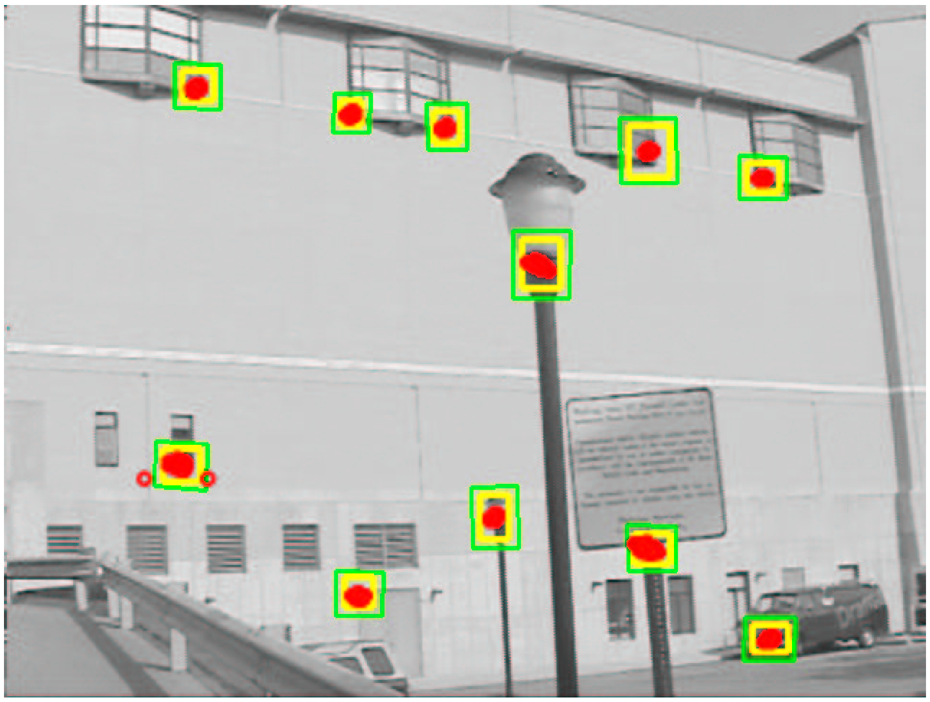
This sequence is collected with a hand-held camera at the parking area behind the Purnell Center for Arts. The idea is to show various occlusion and scale change during the tracking.

In frame 30, 11 features are shown being tracked with the SFM predictions. Frame 70, 90 show the usual integrated tracking and SFM. Note that the light is about to occlude another feature at the background. Frame 100 shows the corner of the window is occluded by the light. The occlusion is detected and the SFM prediction locks at the correct location. At frame 120, another feature, the front of a parked van, is occluded by the sign. Again, both occlusions are detected and predicted by the SFM predictions. At frame 140, the feature on the parked van is recovered, while the window corner is still occluded by the light. In frame 160 the van is occluded again, while the window corner is about to appear again. In frame 170, the window corner appears again, and it is captured by the tracker. Due to the large parallax, the closest feature (on the sign) is lost and the van is still being occluded. The uncertainty of the prediction increases, resulting the samples spread over a relatively large region. In frame 180, the feature on the sign is captured, and the uncertainty of the prediction decreases, that is, they are concentrated at the right locations. Frame 200 shows the usual integrated tracking and SFM. The feature on the van is outside the field of view and is predicted by the SFM samples. This sequence is interesting because that due the special camera trajectory, the occluded feature appears from the same side of the occluding target. This is not easy to deal with if a Condensation type approach is taken. Since Condensation type approach needs the dynamics of the target. Once the target is occluded, its dynamics is no longer observable, and therefore is usually assumed to the same speed before it gets occluded. Under this assumed dynamics, the hypotheses will expect the target to appear from the other of the occluding target, which never happens in this example. By keeping SFM in the loop, our method is quite flexible dealing with all kinds of occlusions, and the occluded target is always being predicted quite accurately according to its 3D locations.

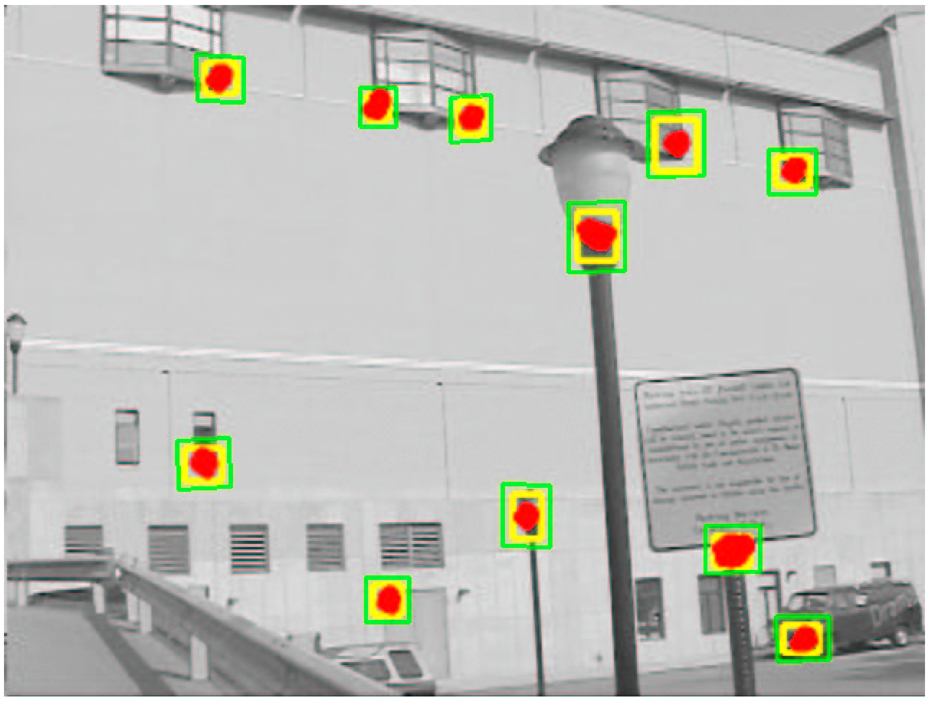
6.2.6 Hand-Held Camera Sequence: Garage Entrance at Beeler Street

This sequence is collected with a hand-held camera on the Beeler Street walking toward the Carnegie Mellon University west campus garage. The garage is about 35 meters away, with traffic on the road. The purpose of the sequence again, is to show the robustness of the integrated tracking and SFM algorithm. Figure 6.7 shows snapshots from the whole sequence.

In frame 30, 13 features are shown being tracked with the SFM predictions. Frame 70, 110, 160 show the usual integrated tracking and SFM. Frame 170 shows the one corner of the garage door

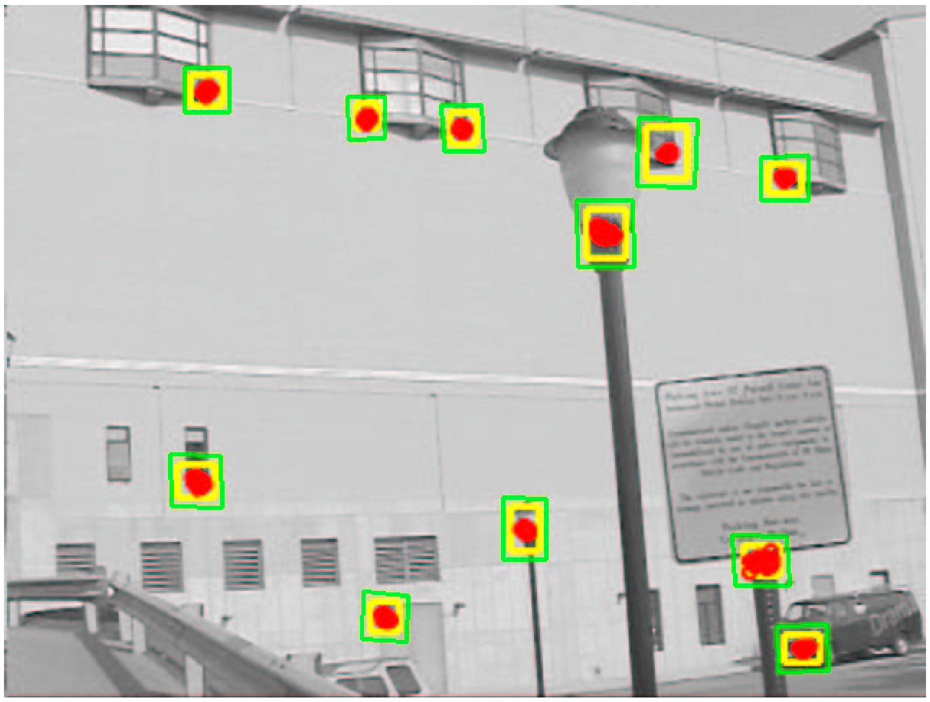


(a) Frame 30



(b) Frame 70

Figure 6.6: Tracking results for hand-held sequence PCA


(c) Frame 90

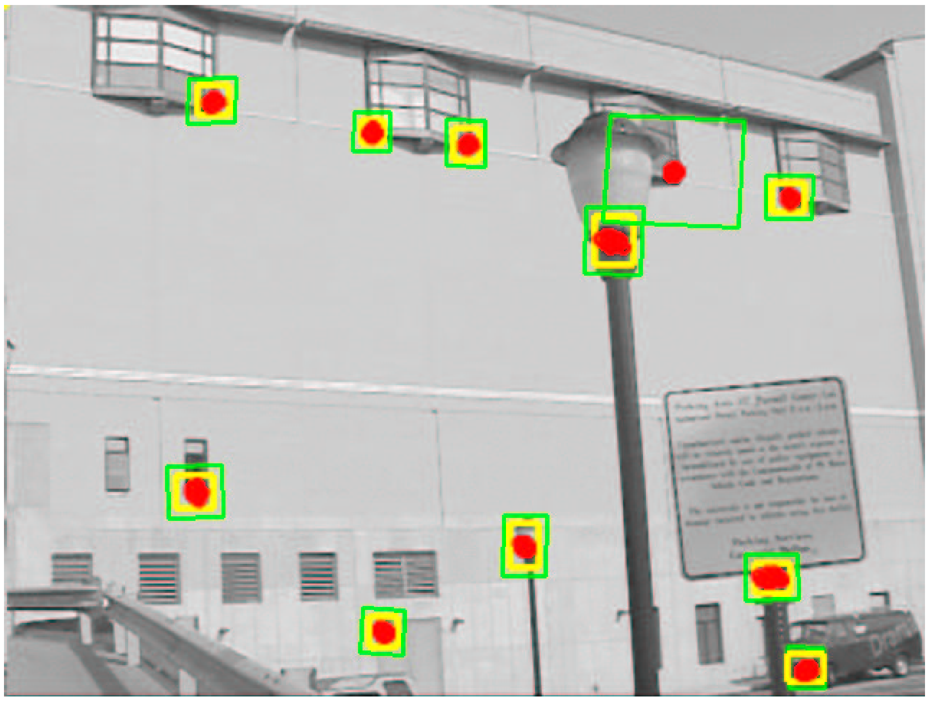
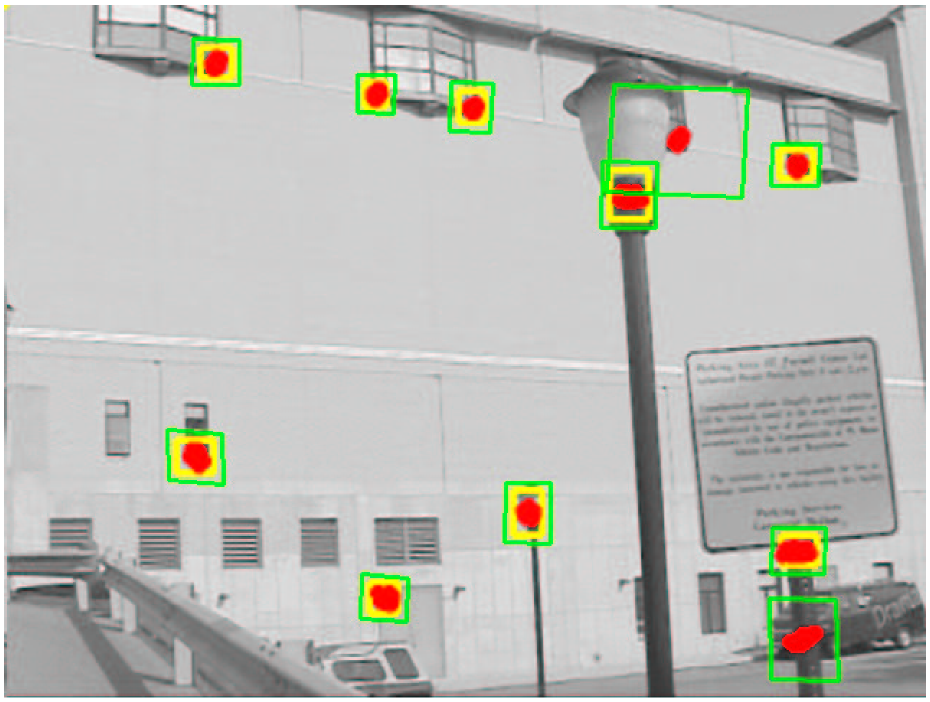
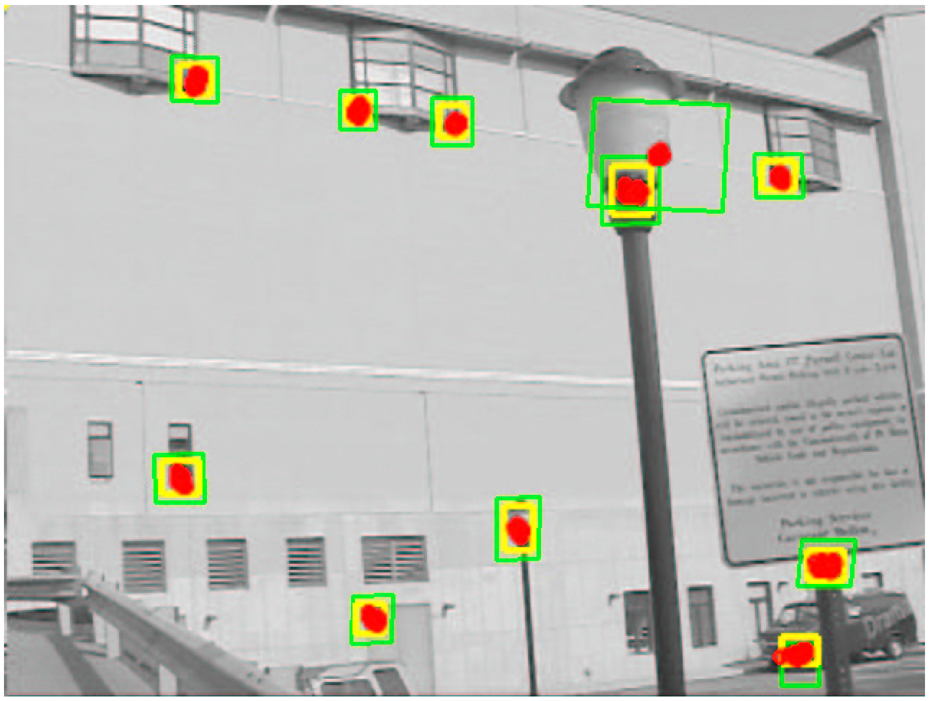

(d) Frame 100

Figure 6.6: Tracking results for hand-held sequence PCA

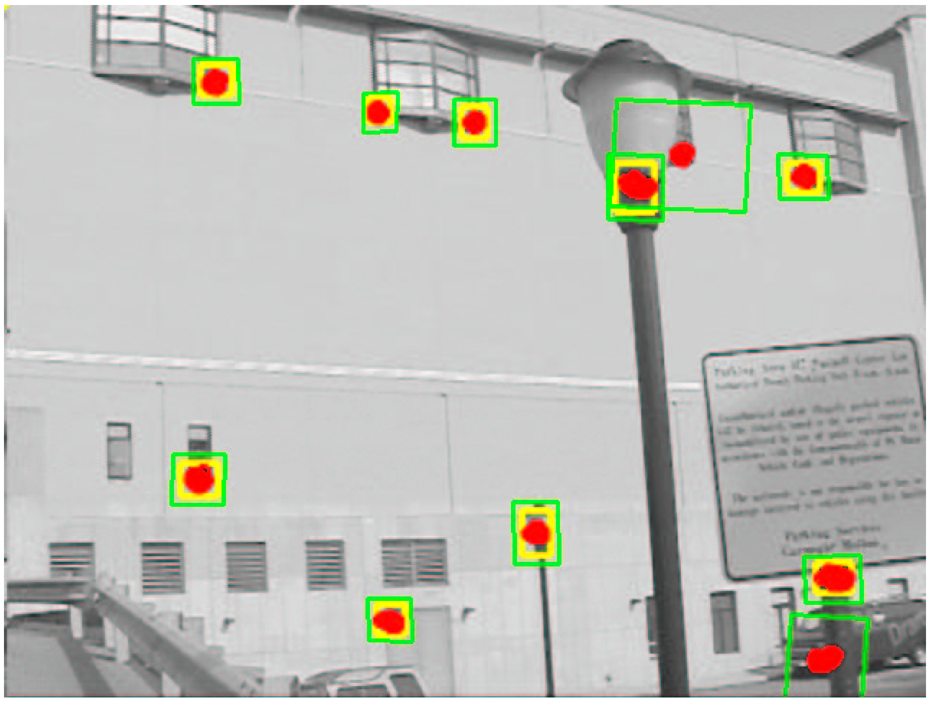


(e) Frame 120



(f) Frame 140

Figure 6.6: Tracking results for hand-held sequence PCA


(g) Frame 160

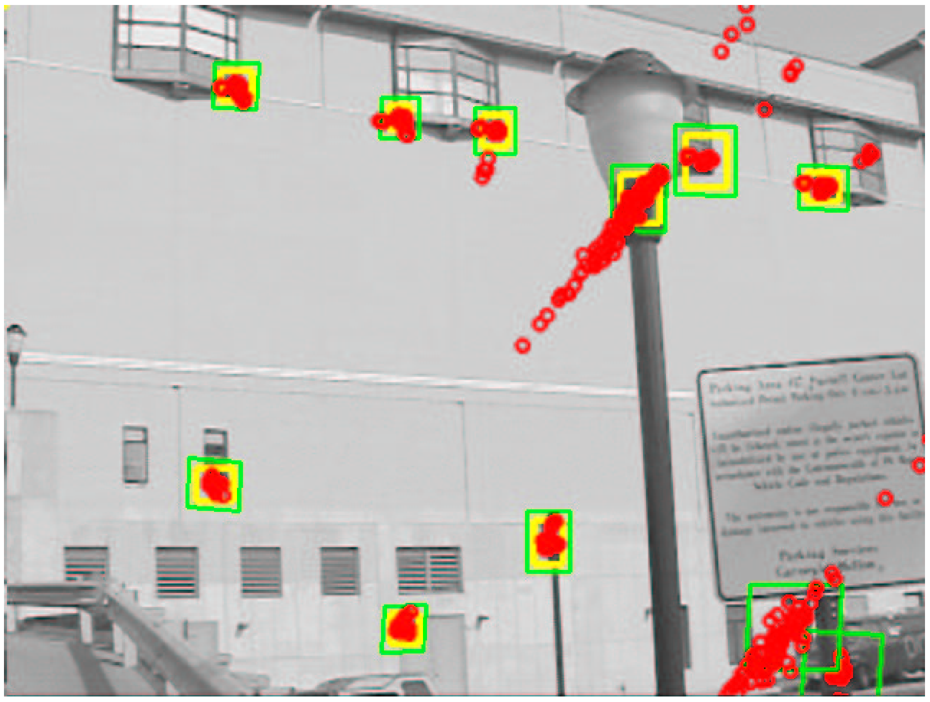
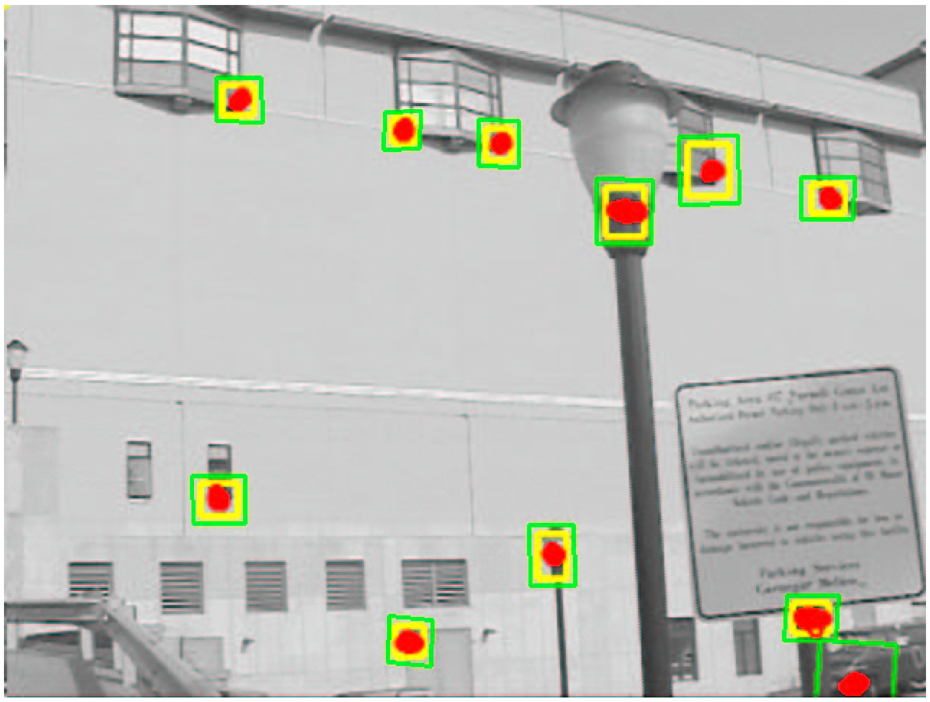
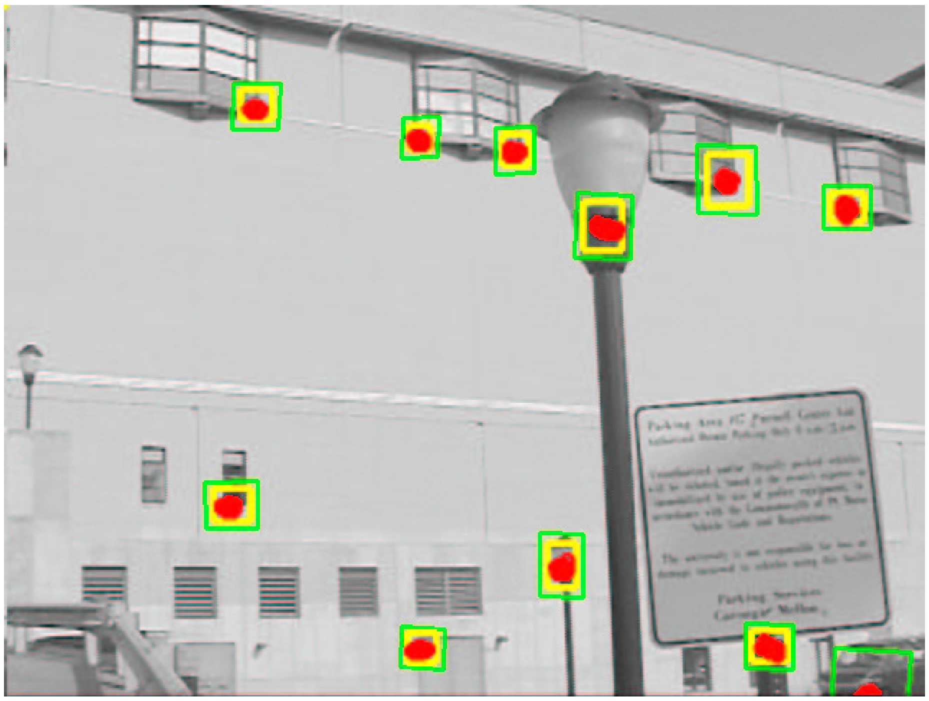

(h) Frame 170

Figure 6.6: Tracking results for hand-held sequence PCA

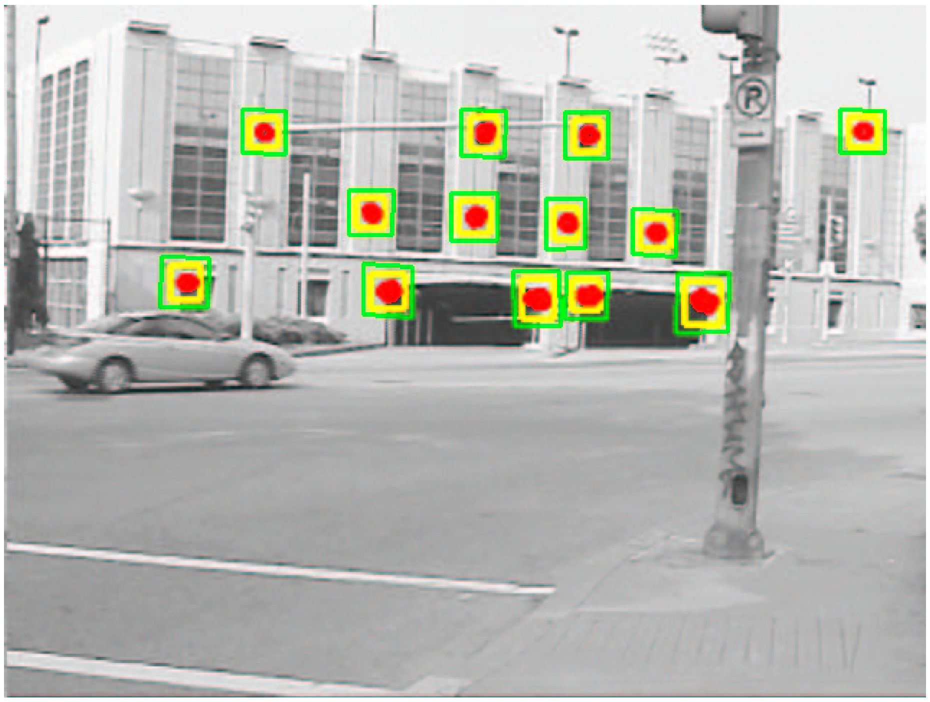


(i) Frame 180

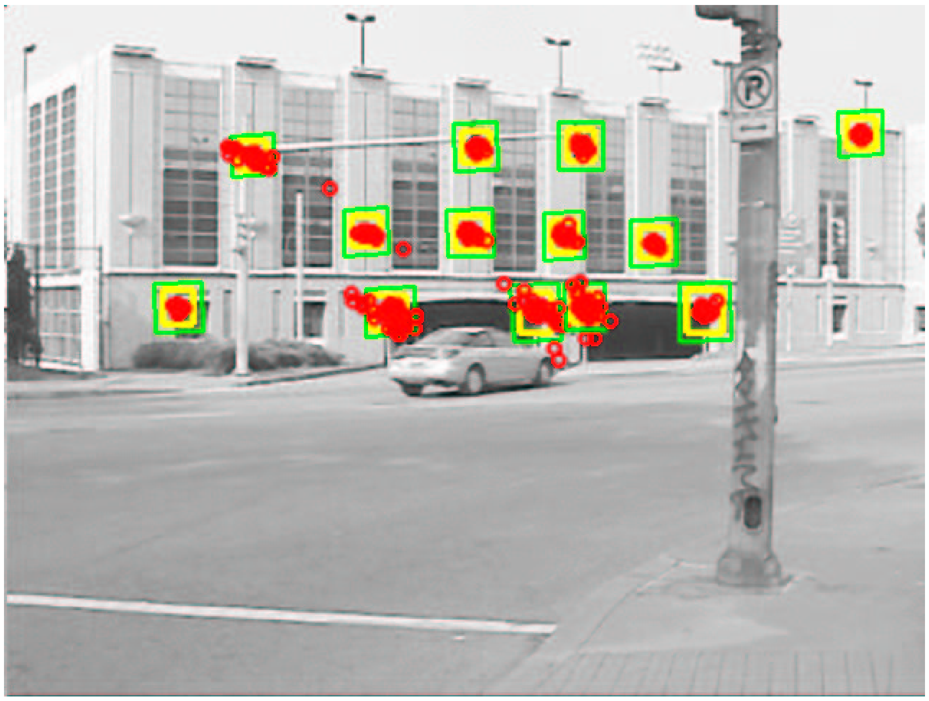


(j) Frame 200

Figure 6.6: Tracking results for hand-held sequence PCA

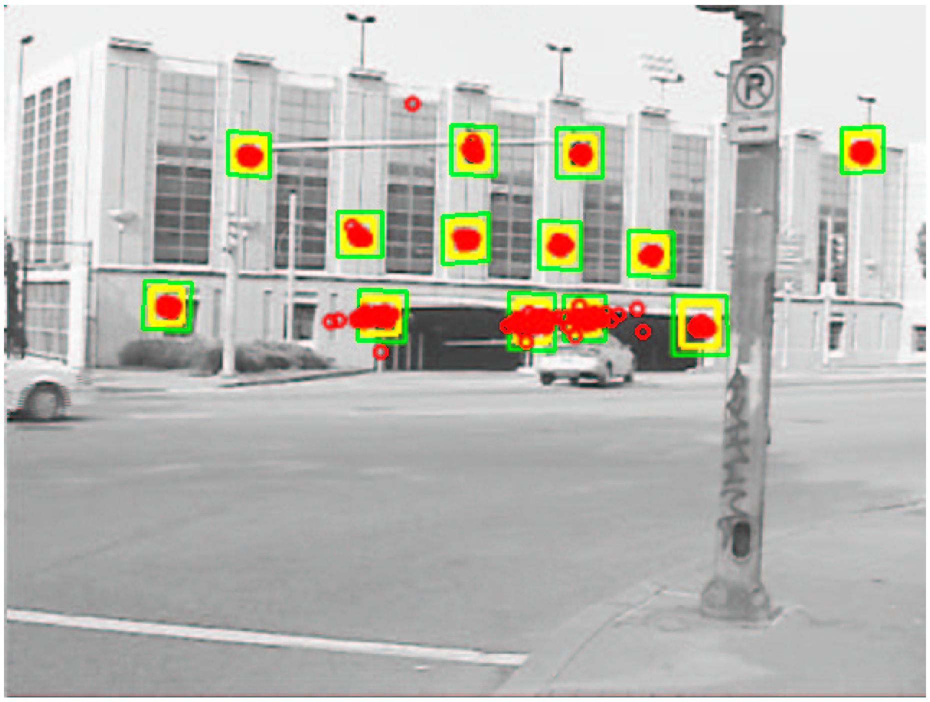


(a) Frame 30

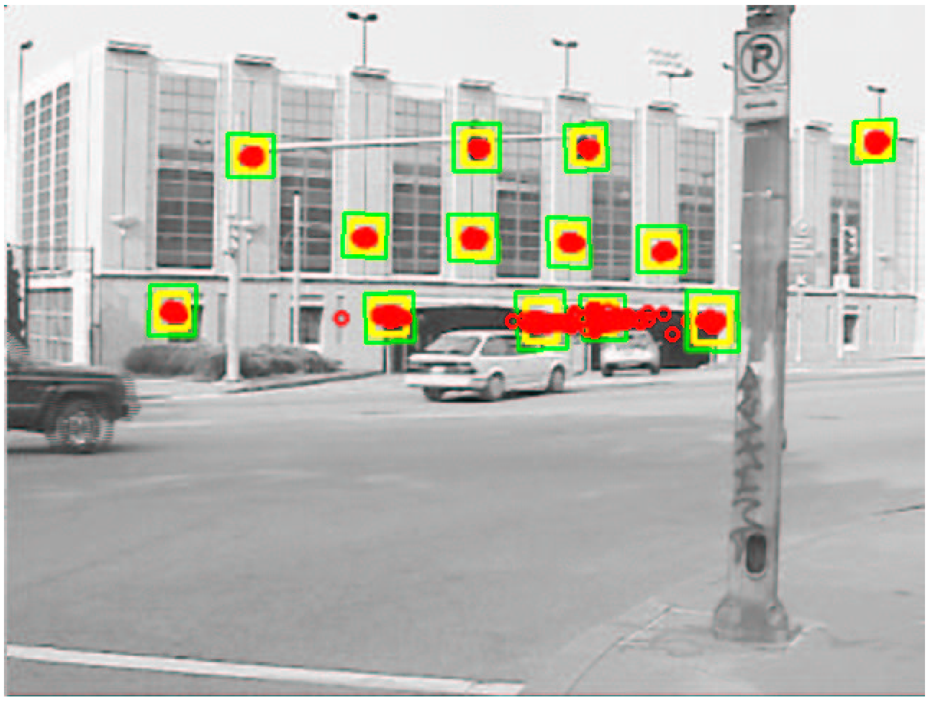


(b) Frame 70

Figure 6.7: Tracking results for hand-held sequence of CMU garage

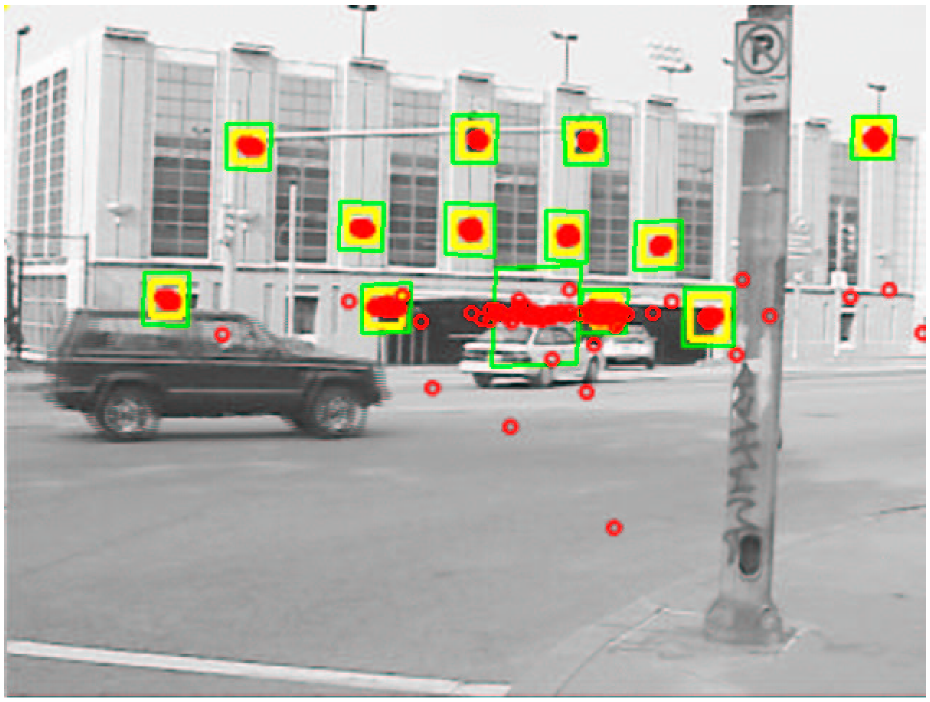


(c) Frame 110

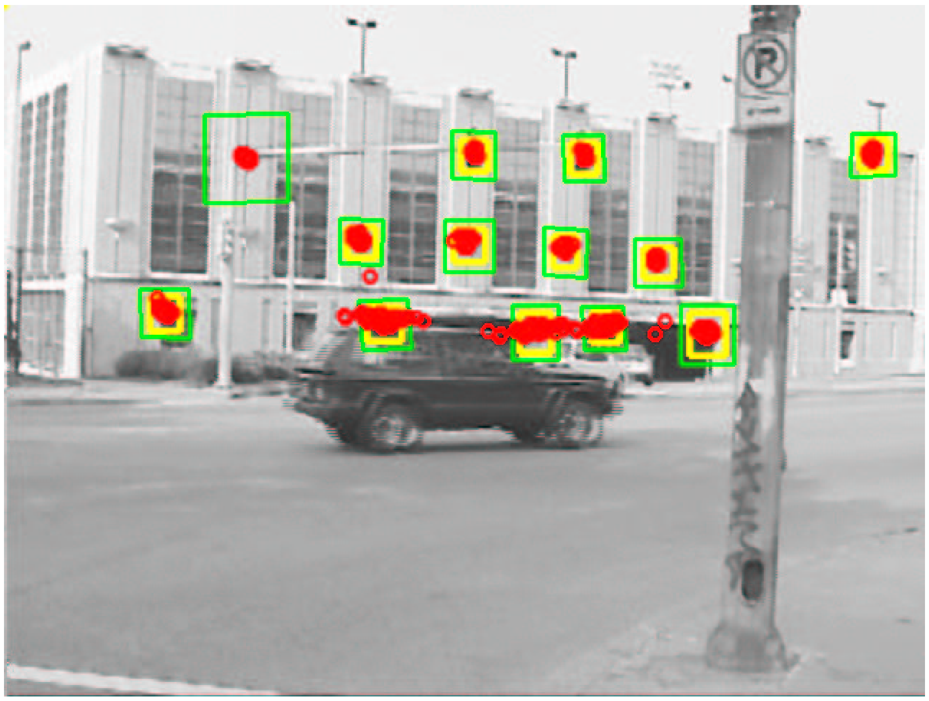


(d) Frame 160

Figure 6.7: Tracking results for hand-held sequence of CMU garage

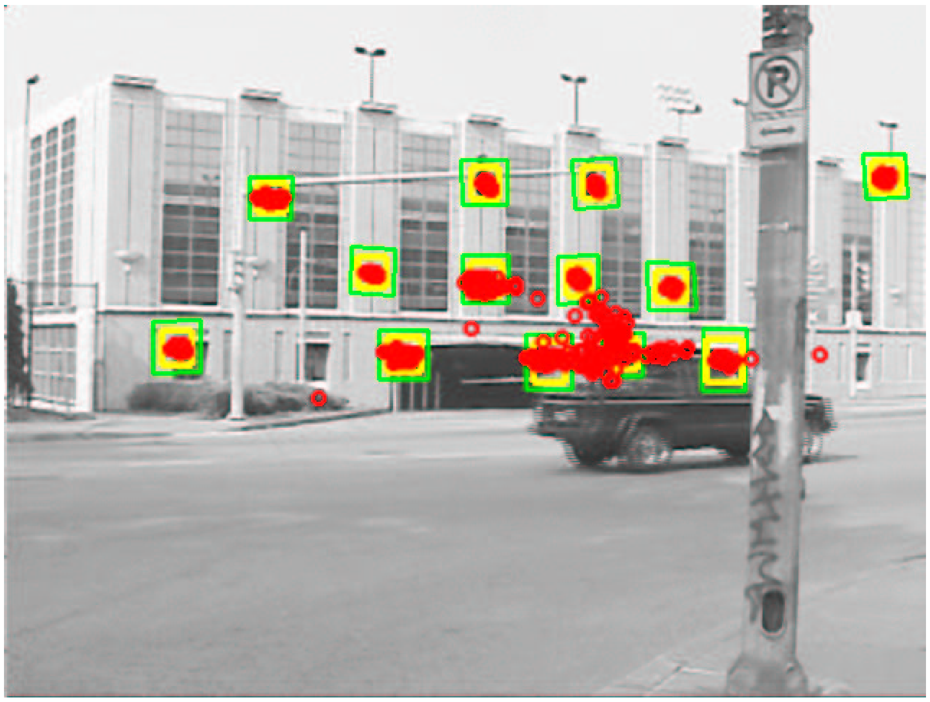


(e) Frame 170



(f) Frame 180

Figure 6.7: Tracking results for hand-held sequence of CMU garage


(g) Frame 190

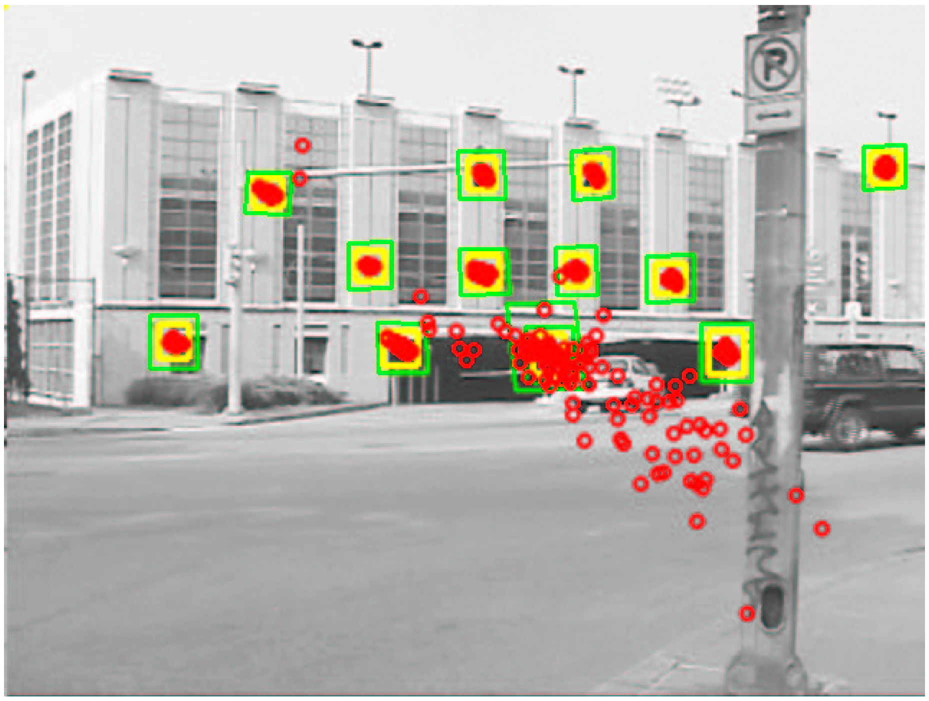
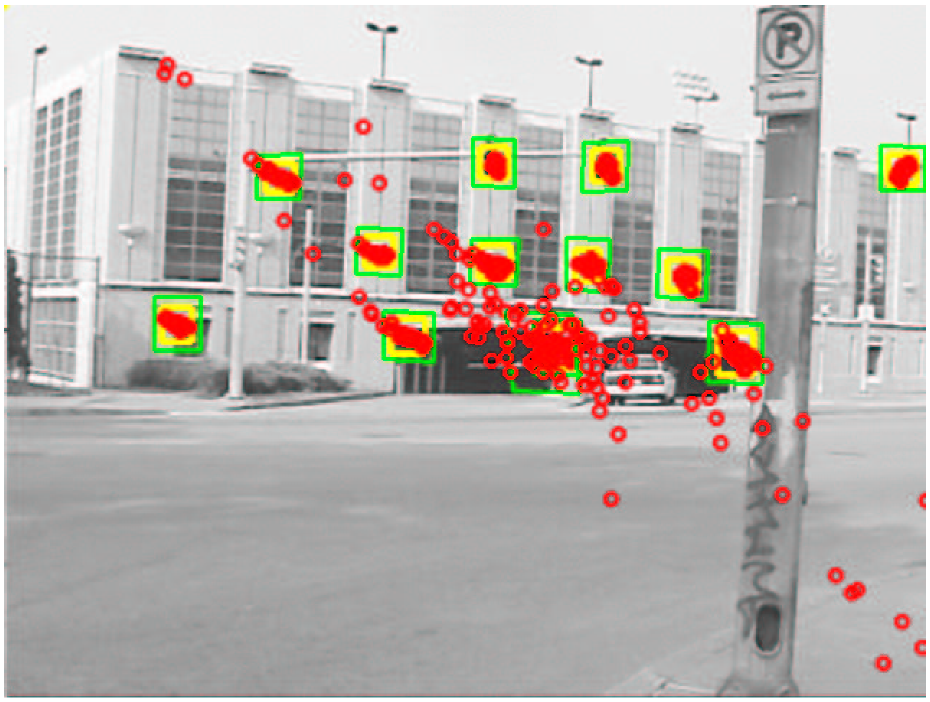
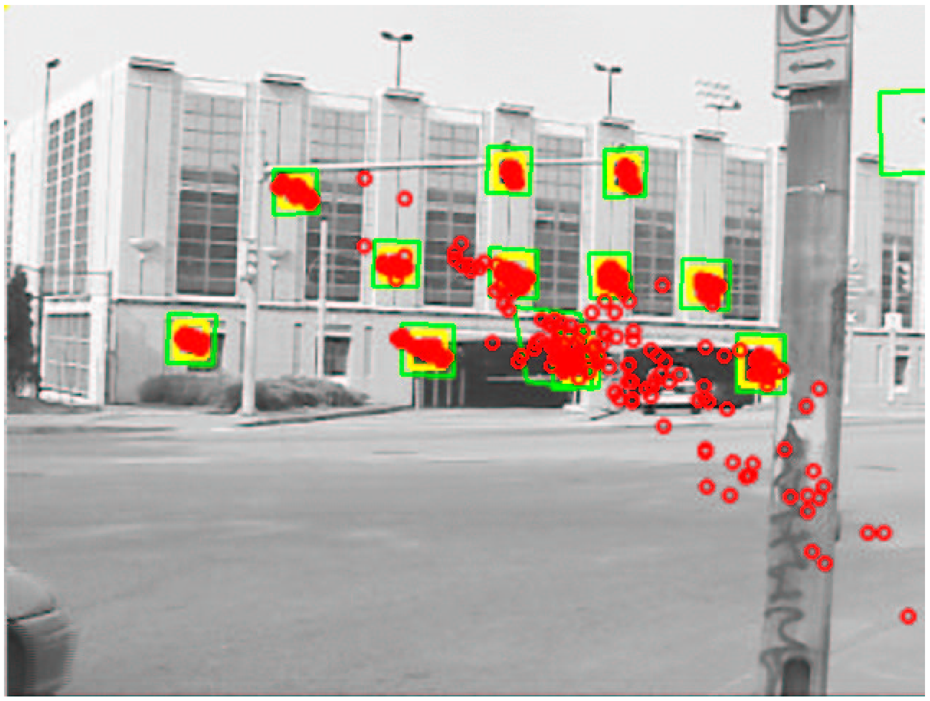

(h) Frame 200

Figure 6.7: Tracking results for hand-held sequence of CMU garage

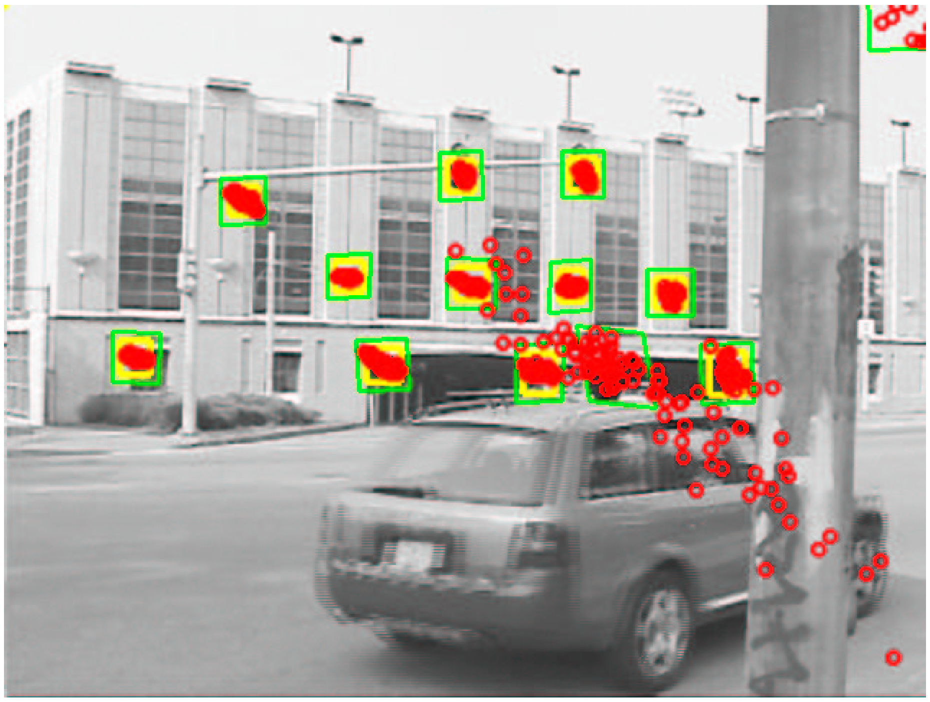


(i) Frame 230

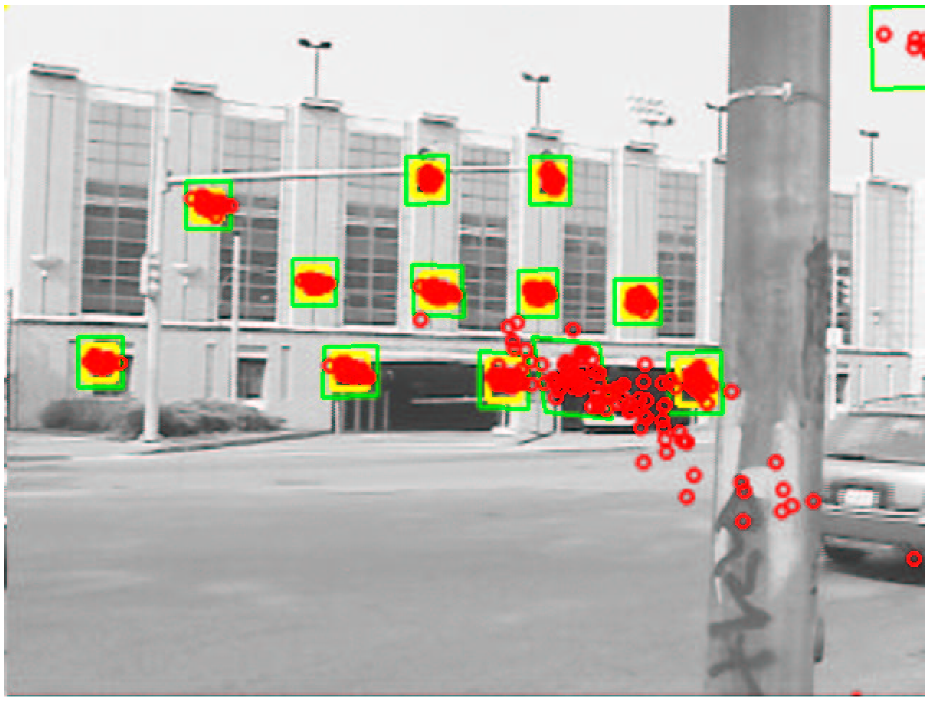


(j) Frame 280

Figure 6.7: Tracking results for hand-held sequence of CMU garage



(k) Frame 380



(l) Frame 400

Figure 6.7: Tracking results for hand-held sequence of CMU garage

is occluded by the incoming traffic. The occlusion is detected and the SFM prediction locks at the correct location. At frame 180, the corner feature is recovered. Frame $190, 200$ show a jeep passing by and causing occlusions to several features. After it moves away, one corner point is lost and the SFM prediction is uncertain. Frame $230, 280, 380, 400$ show the subsequent frames. Most of the features are being tracked well. One feature is lost due to going out of field of view. The corner feature on the garage door is predicted approximately correct (see the green search window is located at the right location) but the tracker fails to recover from it, due to the appearance matching score is not high enough. We conjecture that one reason for the SFM prediction to be so uncertain can be due to the fact that most of the features are on a plane, which is a degenerate case for SFM. In this case, the SFM result is naturally uncertain and it reflects on the feature predictions.

6.2.7 Hand-Held Camera Sequence: Lab Scene

This sequence is collected with a hand-held camera in an indoor environment. The camera motion is made quite jerky intentionally, only to show how our method copes with abrupt camera motions. Figure 6.8 shows snapshots from the whole sequence.

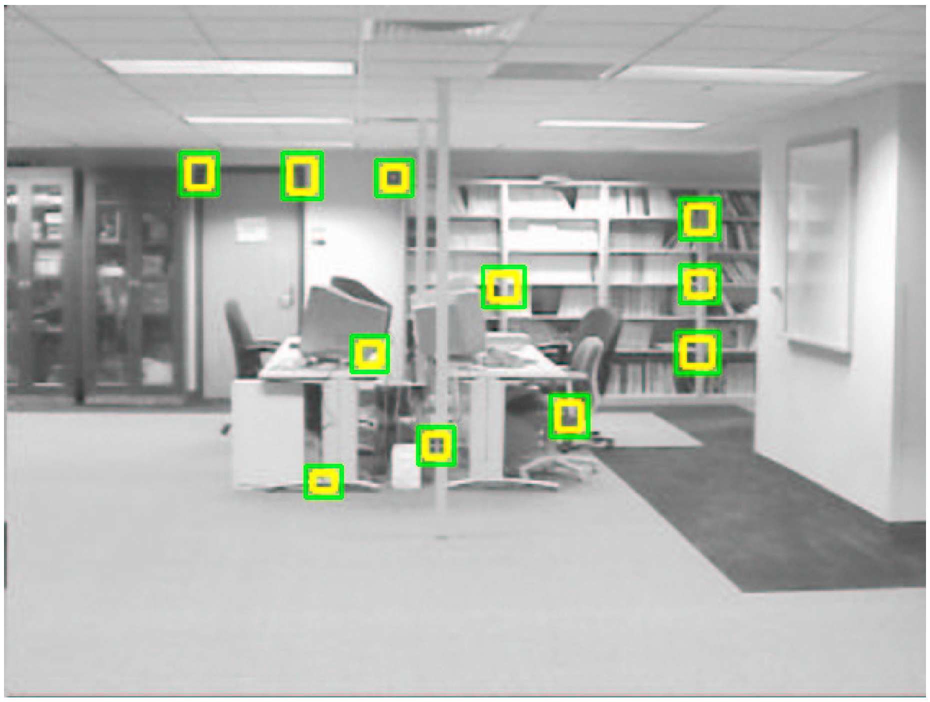
In frame 1 11 features are selected. Frame $10, 20, 30, 40, 60, 70, 80$ show the robust tracking and SFM. Note there is substantial sudden camera rotation in the sequence. Frame 100 shows several features are lost due to the vigorous camera motion, but they are recovered in later frame 110. Frame $160, 190, 250, 270, 280, 290$ show the tracking and SFM for the rest of the sequence. It can be seen that our method works in presence of continuous camera jittering.

6.3 Uncertainty Analysis

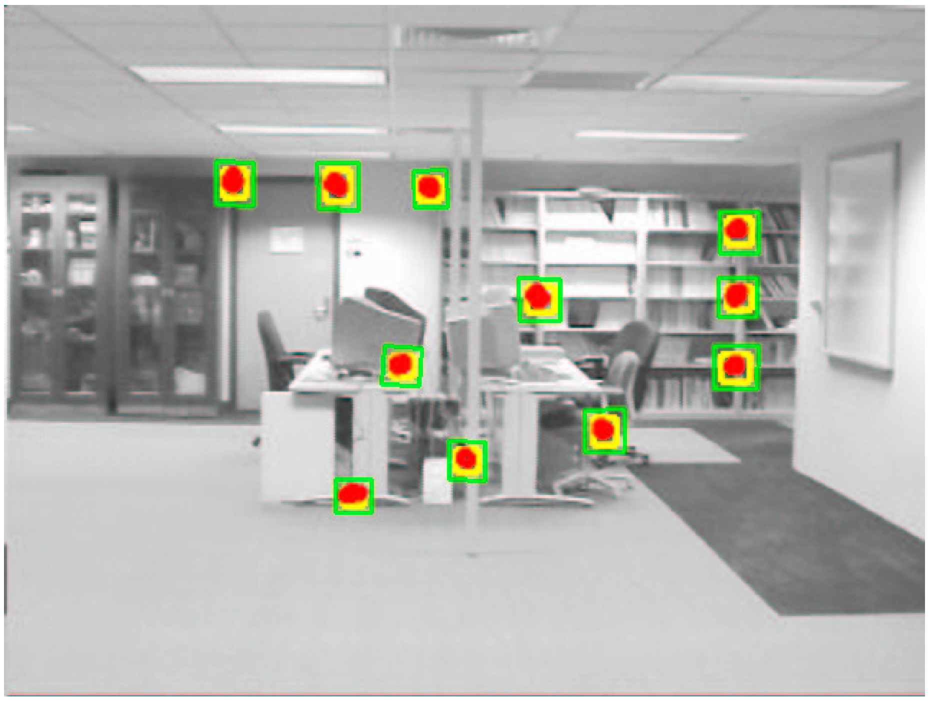
In this section, we analyze the uncertainty of both tracking and reconstruction. We show that the tracking uncertainty is reduced after fusing with SFM predictions. And the uncertainty of SFM is also reduced by fusing with previous reconstructions. Since we need one scalar for the uncertainty analysis, the variance is used as a simple measurement.

6.3.1 Tracking

Figure 6.9 shows the tracking uncertainty for one sequence HH1. It can be seen that the uncertainty of the tracking is reduced after fusing with the SFM results. The uncertainty is measured by the variance of the feature correspondences in the sample set. To emphasize the difference, a log scale is used for plotting the variance. One may expect that, when the SFM result becomes accurate, it should reduce the tracking error further. However, we fuse the tracking uncertainty with SFM predictions, which depends on the camera motion dynamics. With uncontrolled camera motion,

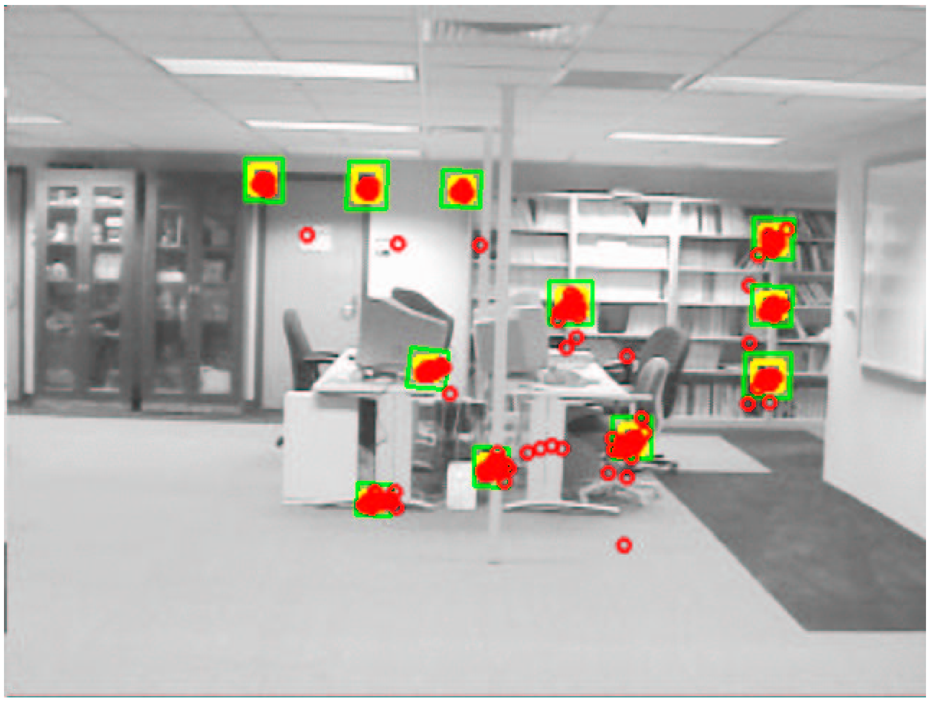


(a) Frame 1

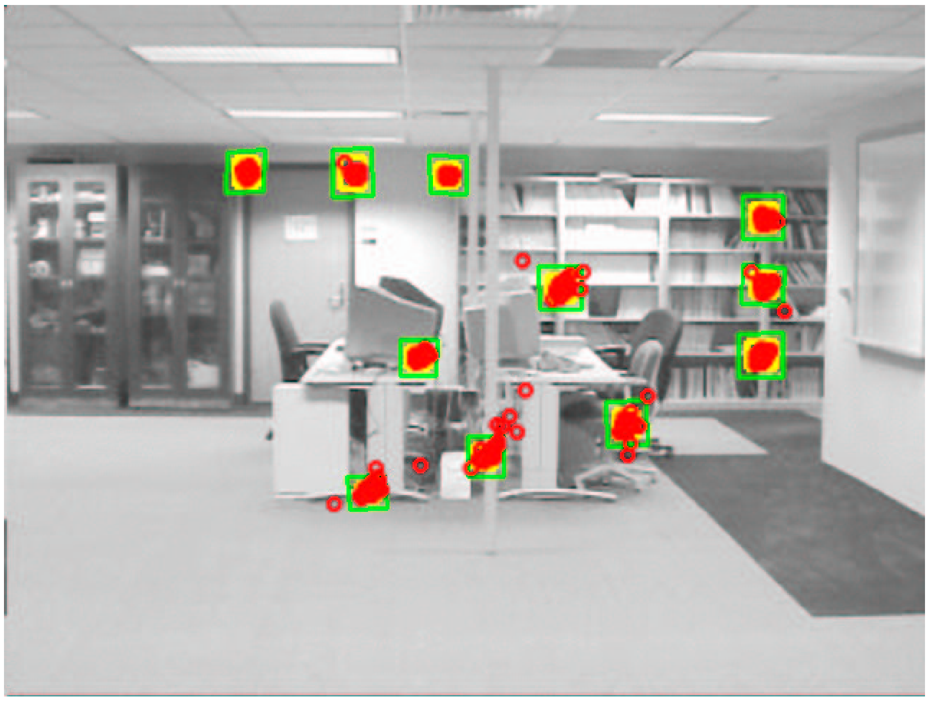


(b) Frame 10

Figure 6.8: Tracking results for sequence with large sudden motion

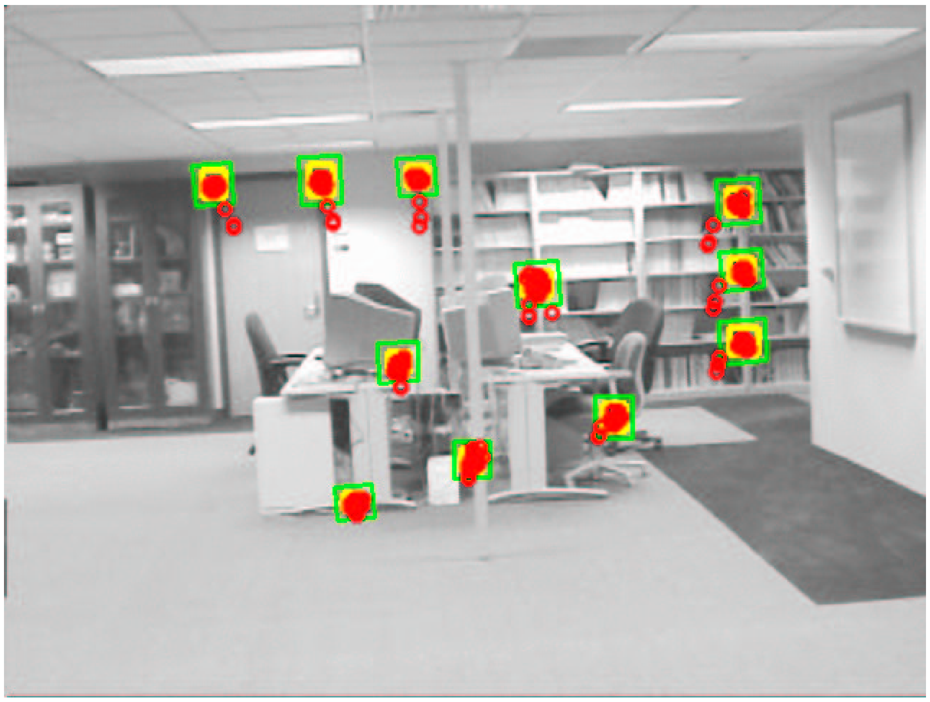


(c) Frame 20

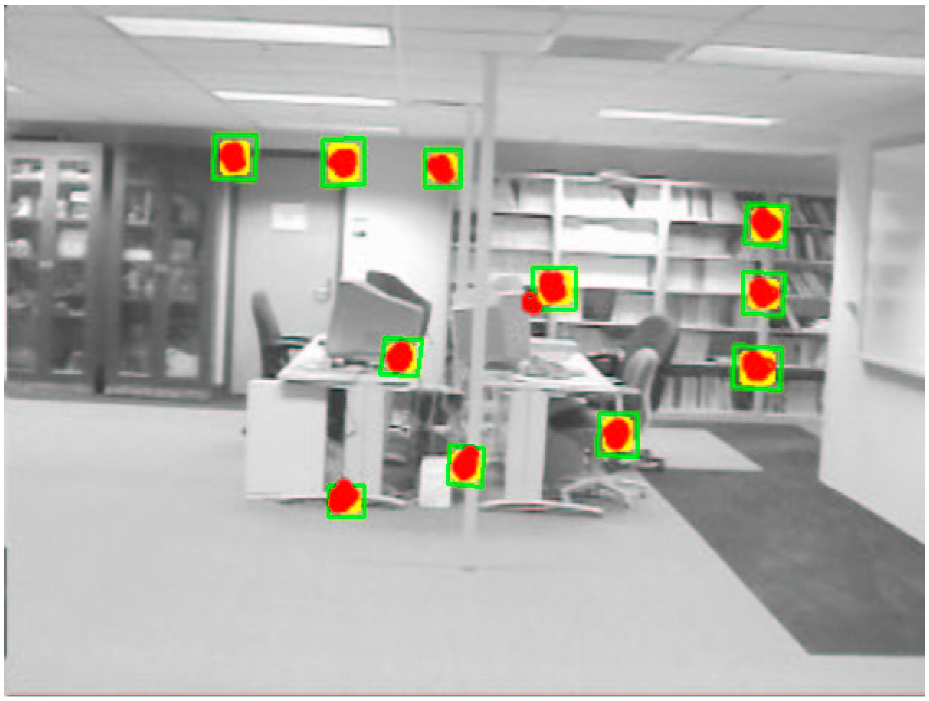


(d) Frame 30

Figure 6.8: Tracking results for sequence with large sudden motion

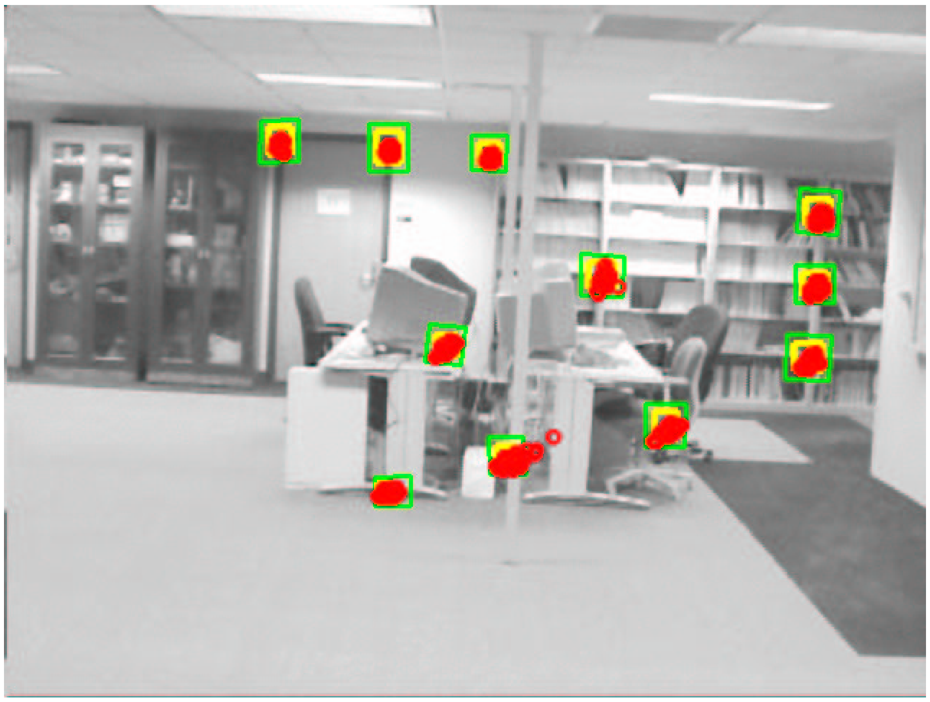


(e) Frame 40

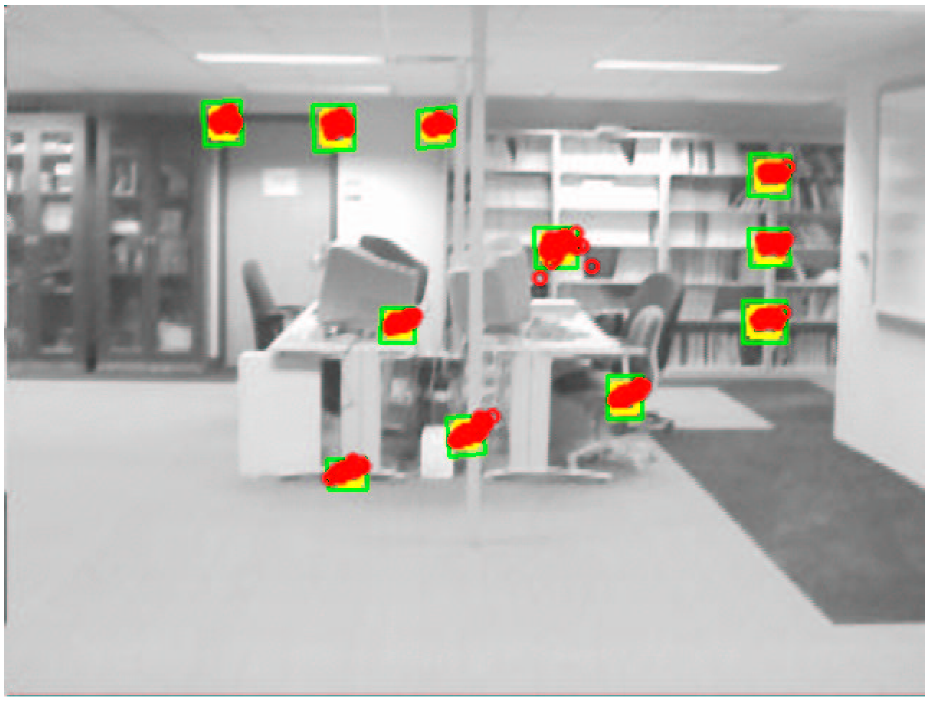


(f) Frame 60

Figure 6.8: Tracking results for sequence with large sudden motion

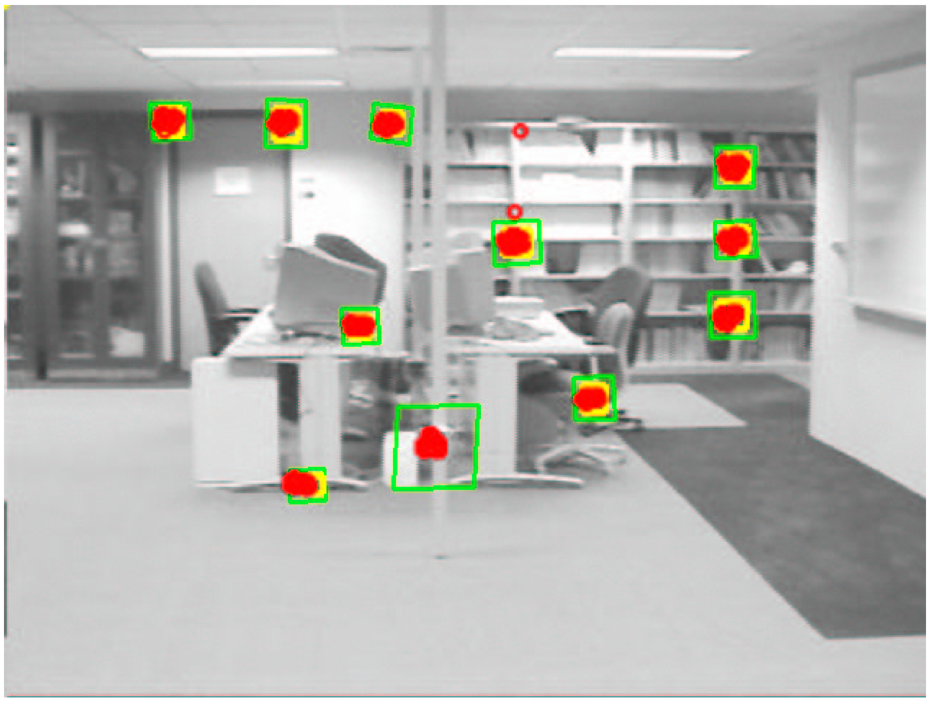


(g) Frame 70

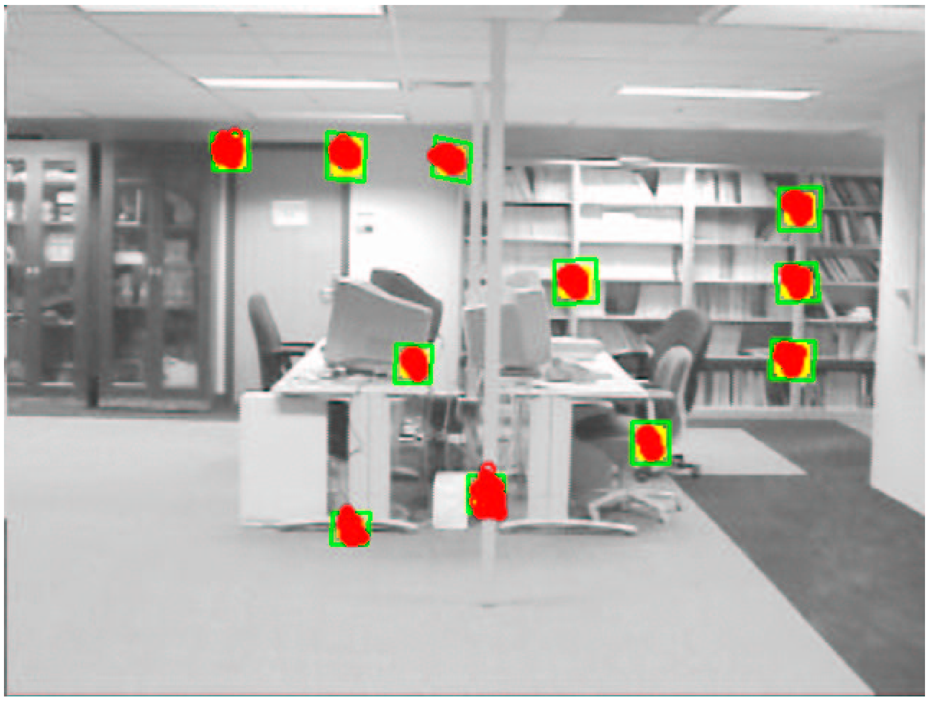


(h) Frame 80

Figure 6.8: Tracking results for sequence with large sudden motion

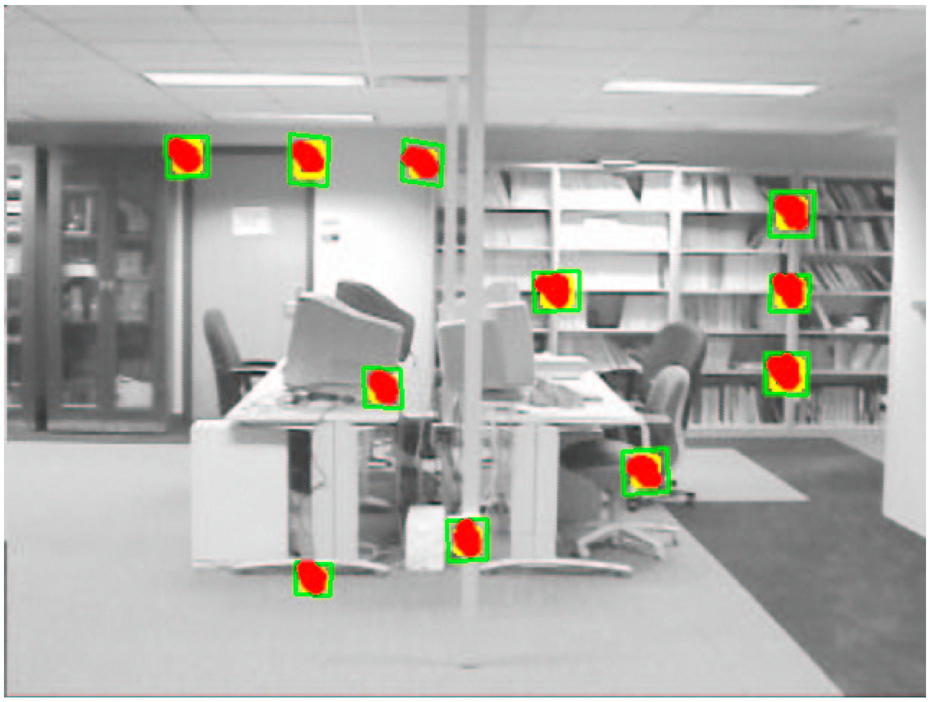


(i) Frame 100



(j) Frame 110

Figure 6.8: Tracking results for sequence with large sudden motion


(k) Frame 160

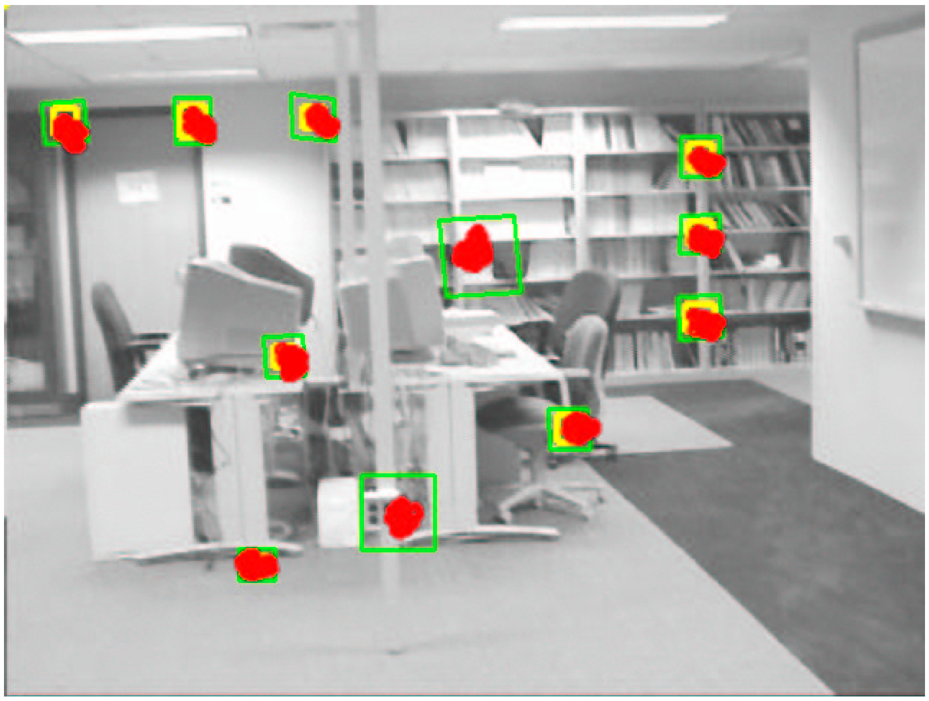
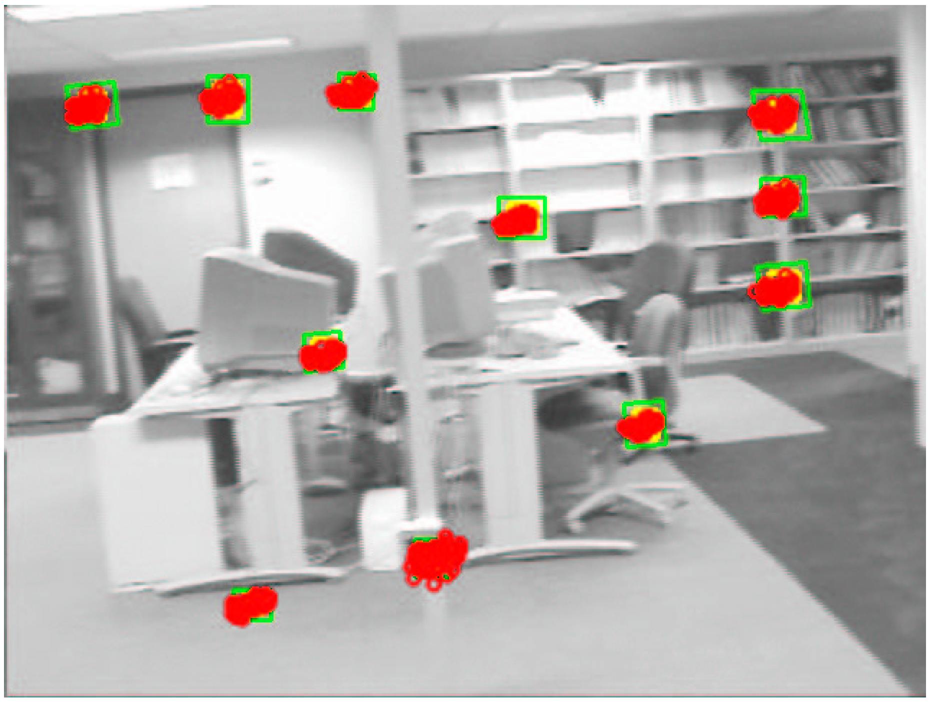

(l) Frame 190

Figure 6.8: Tracking results for sequence with large sudden motion


(m) Frame 250

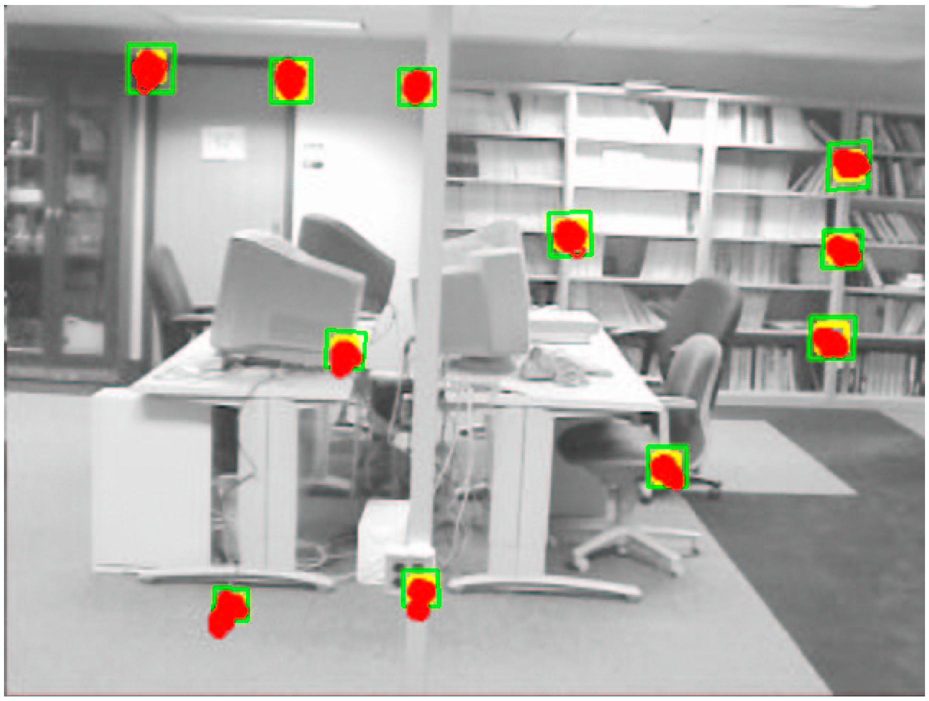
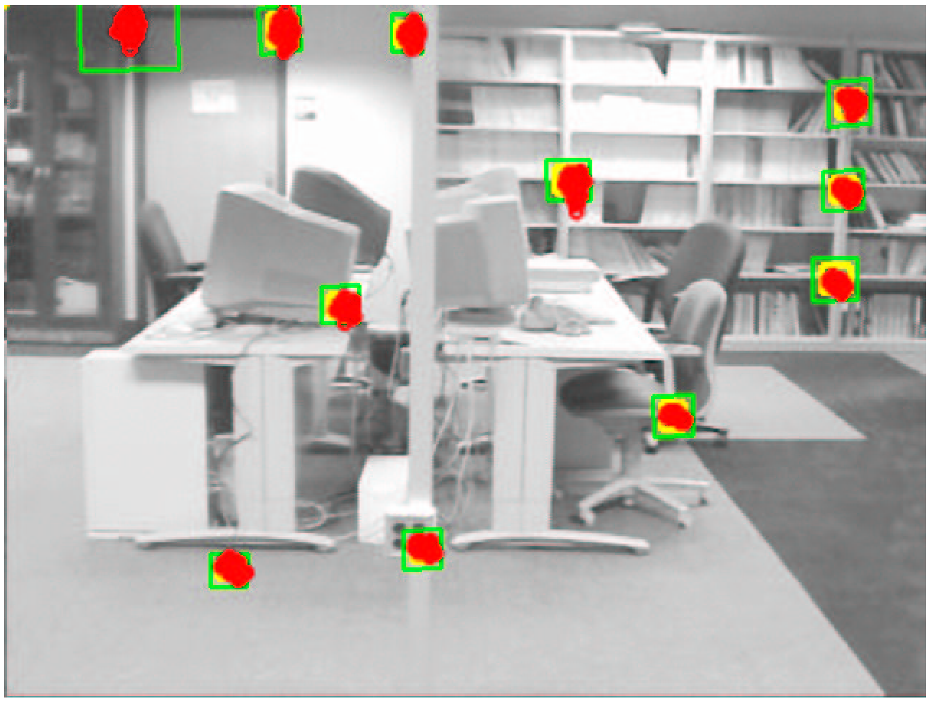
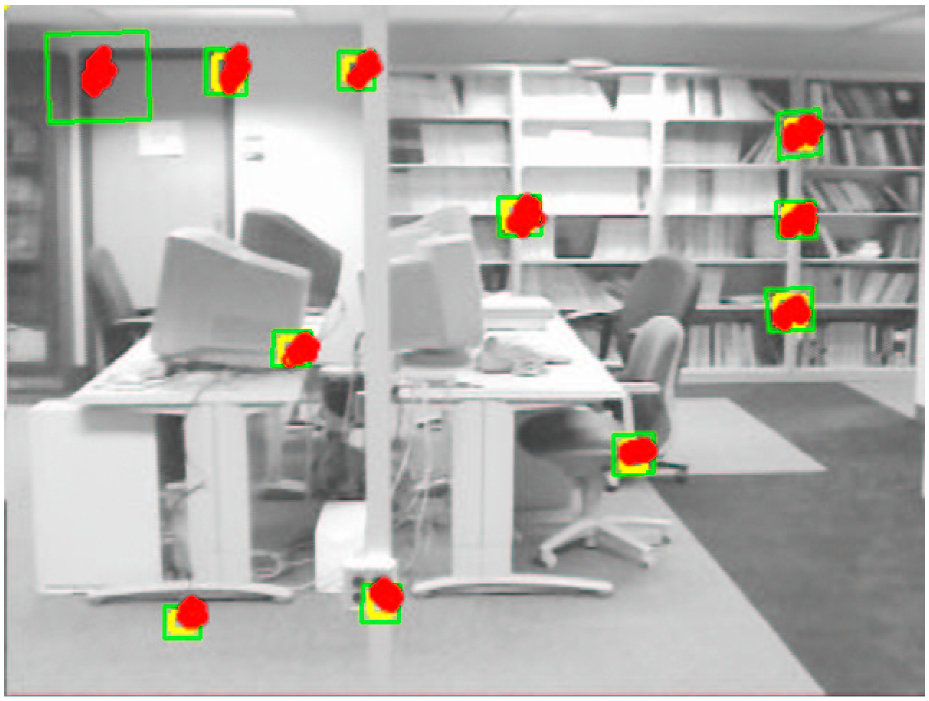

(n) Frame 270

Figure 6.8: Tracking results for sequence with large sudden motion



(o) Frame 280



(p) Frame 290

Figure 6.8: Tracking results for sequence with large sudden motion

the camera motion is quite uncertain and this uncertainty reflects in the SFM predictions. In other words, we can not precisely predict the feature correspondences from current SFM along.

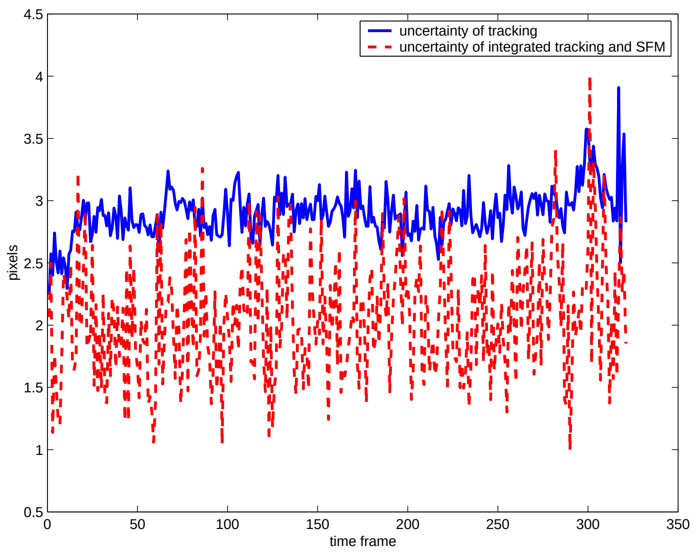


Figure 6.9: The tracking uncertainty over time

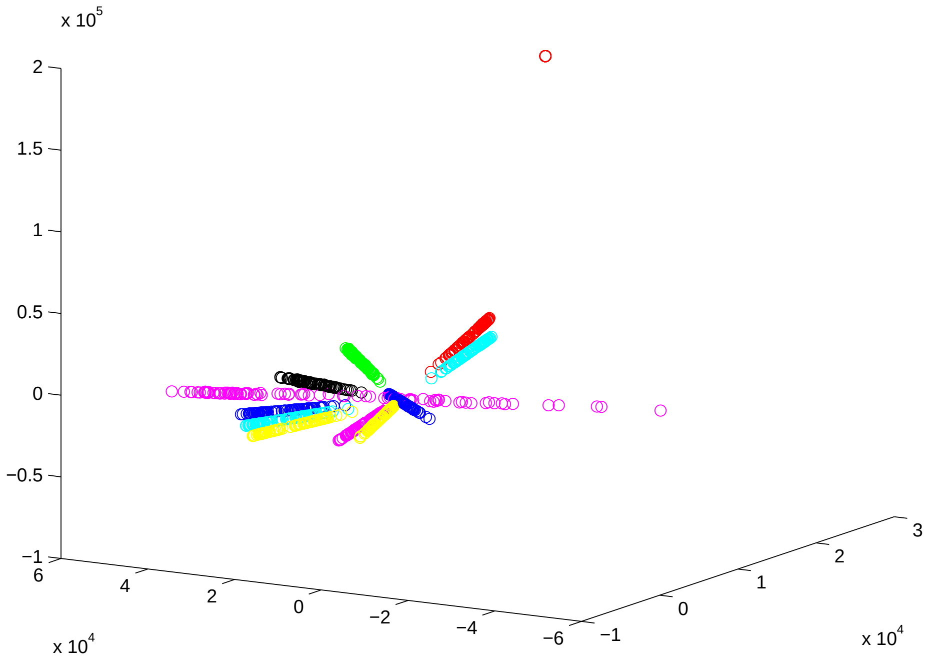
6.3.2 Reconstruction

Figure 6.10 show two sample sets for reconstructions at different time frames. It can be seen that the uncertainty is smaller at frame 600 than at 350. The distributions are long tailed and may not be well captured by covariance matrices.

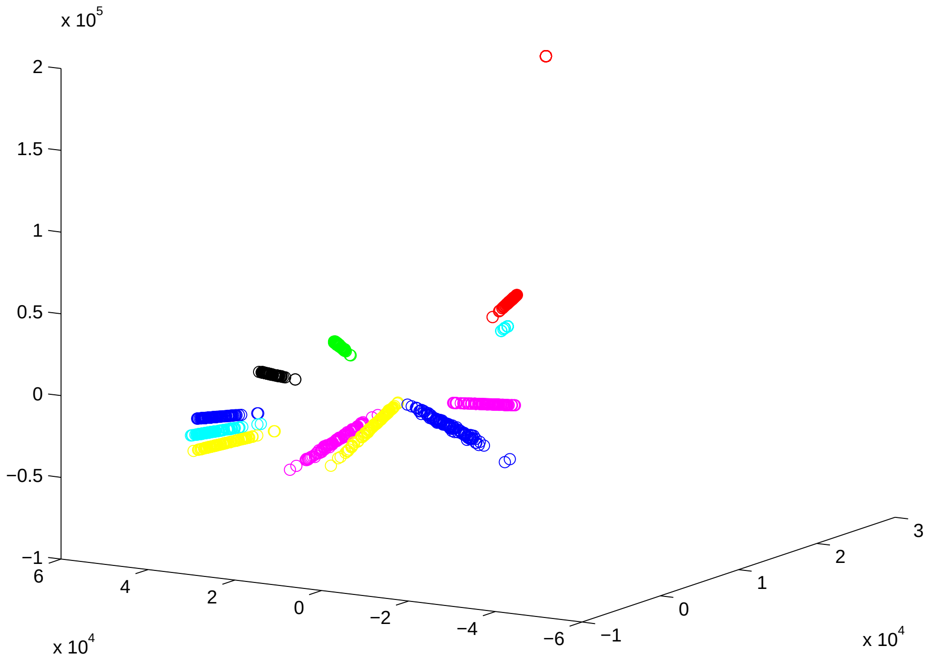
Given a sample set $X = (\mathbf{x}_i, i = 1, ..., N)$ for the reconstruction, the uncertainty of the reconstruction can be approximated by $\sum_{i=1}^{N} Var(\mathbf{x}_i)$, where Var can be defined as the trace of the covariance matrix for the vector $\mathbf{x}$. Note that the variance is used here only as a simple method to measure the scarceness of the points in the reconstruction.

Figure 6.11 shows the reconstruction uncertainty for one sequence HH1. It can be seen that the uncertainty of the reconstruction decreases over time. The uncertainty is reduced after the current SFM result is fused with the reconstruction at previous time frame.

Reprojection error is often used as a measurement for the quality of SFM. Figure 6.12 shows the reprojection error over time for several test sequences. The red dash line indicates the tracking status. The possible status are (0: no fusion; 1 fusion started; 2 target get occluded; -1 target recaptured; 3 large camera motion). Several facts are noticed. The reprojection error has been fairly small. Targets frequently get occluded and then recaptured. Some of the detected occlusion are due to some factors other than true occlusion: (1) temporary large appearance change which



(a) reconstruction sample set at frame 350



(b) reconstruction sample set at frame 600

Figure 6.10: Sampled reconstructions of the Omni-Smith 2 sequences at two different time frames

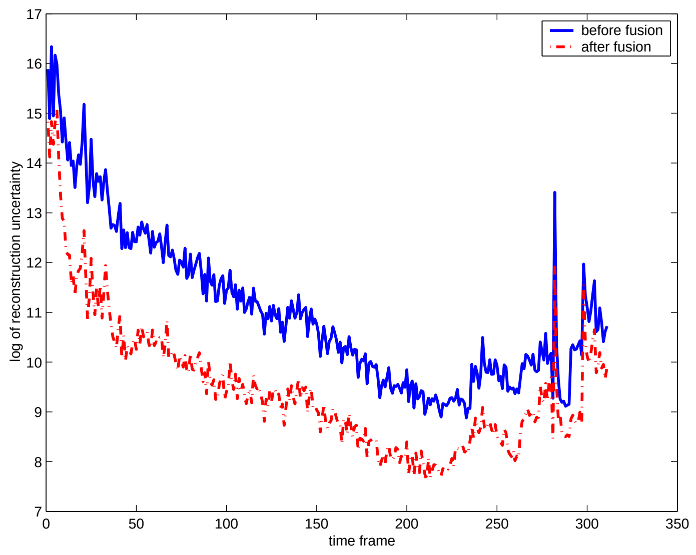


Figure 6.11: The reconstruction uncertainty over time

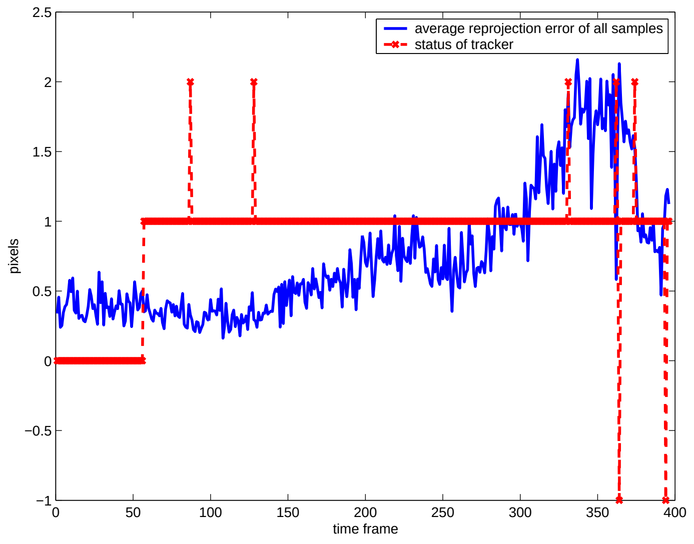
causes the tracker to fail, due to image quality, motion blur etc.; this case usually can be resolved in a later frame (2) large motion causes the target to move beyond the search region; in this case, the tracker relies on the SFM predictions to guide the search at later frames. The occlusion status shown in Figure 6.12 indicates new target is being occluded in that time frame. Sudden camera motion usually results in increased reprojection error, but the error goes back to usual level later on, as shown in the *office* sequence.

6.4 Rotation Compensation

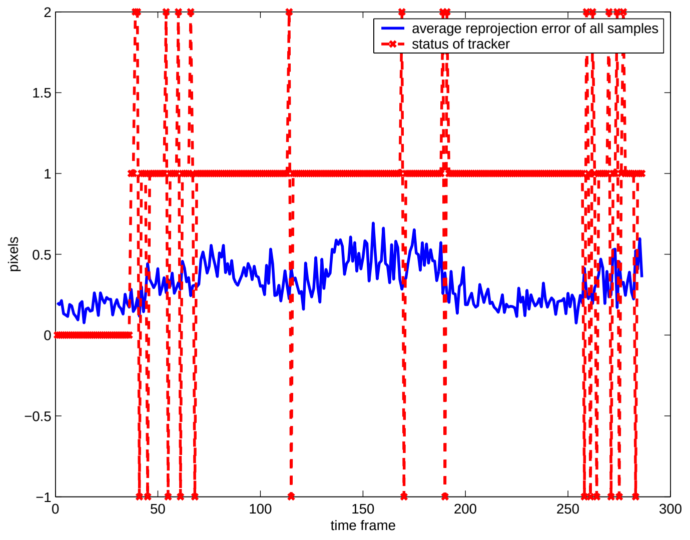
To verify the camera motion is estimated correctly, we simulate the stabilization of the sequence by compensating the estimated rotation. With a calibrated camera (as in our case), it only involves a homography transformation to the original sequence. We align all the image with respect to the original frame. This naive stabilization is applied to the lab sequence, which has a lot of camera jittering. Figure 6.13 shows several frames of the stabilized scene. The image on the left is the original frame and the aligned image is shown on the right.

6.5 Conclusion

In this chapter, we demonstrate the robust tracking and SFM results with various experiment settings, omnidirectional camera and conventional camera, outdoor and indoor. In particular, we show

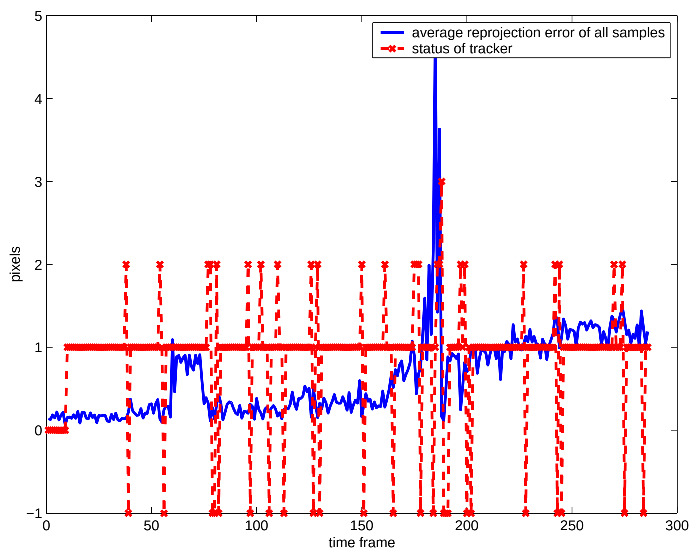


(a) sequence HH



(b) sequence NSH

Figure 6.12: Reprojection error coupled with tracking status

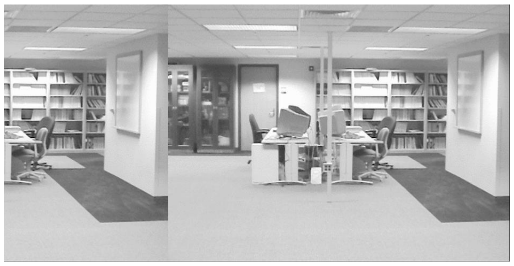


(c) sequence Office

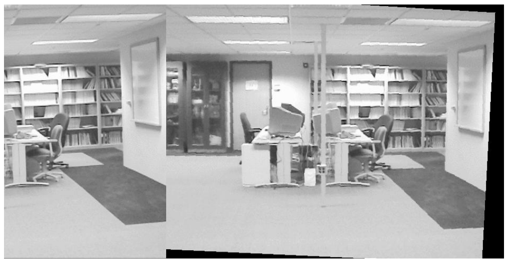
Figure 6.12: Reprojection error coupled with tracking status

that the tracking is made robust against occlusion and large camera motion, by integrating with SFM. Some of the sequences are taken with a hand-held camera, without controlling camera motion. We actually intentionally increased the level of vibration during the recording procedure.

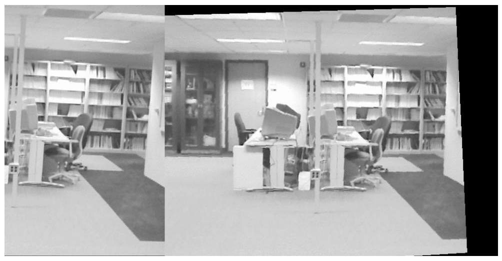
Faithfully representing the uncertainty is the key in our system performance. With the sampling method, we are able to fully characterize the uncertainty in both tracking and SFM under both good and extreme conditions. A unique feature of our system is that at any time of the process, the system is fully aware of the degenerate conditions in the data, such as occlusion and sudden camera motion, and takes proper actions against them. The resulting system can automatically perform tracking and SFM under rather challenging conditions and we believe it is one step closer to a fully automatic robust SFM module, which can find applications in various scenarios.



(a) Frame 1

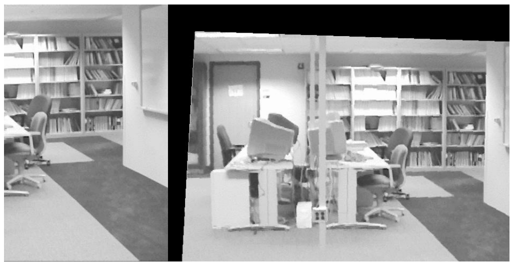


(b) Frame 43

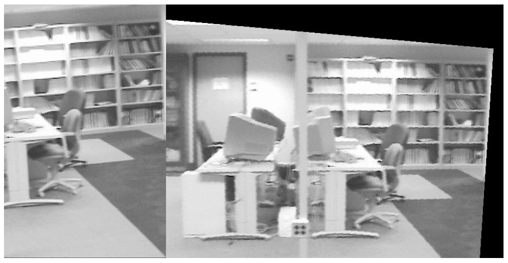


(c) Frame 74

Figure 6.13: Compensating the estimated rotation



(d) Frame 197



(e) Frame 255

Figure 6.13: Compensating the estimated rotation

Chapter 7

Conclusions and Future Work

Tracking and SFM are important topics for visual navigation and other applications. Even though both the tracking and SFM have made significant progress recently, it is still difficult for them to be applied in real applications. Namely, tracking with only appearance suffers from occlusion and ambiguity, while SFM is still considered hard to use reliably in practice.

In this thesis, I analyze the uncertainty of both tracking and SFM, and explore the idea of using sampling methods to characterize the uncertainty in both tracking and SFM. Sampling is a powerful method to capturing the uncertainty of an estimator. The particle filter provides a natural vehicle to propagate the sample based uncertainty through time. Both advantages are exploited in our integrated tracking and SFM algorithm. We show that with the sampling method, we are able to capture the uncertainty of the process under extreme conditions, which is often beyond the reach of conventional approaches. Since we can handle those extreme conditions, which usually cause the failures for conventional approaches, our tracking and SFM system exhibits improved performance in terms of robustness. Sampling provides a simple and general tool for uncertainty representation. With the increasing available computing power, we believe this type of approaches should find more applications in practice.

From a more practical standpoint I believe our system is more reliable and easy to use compared to many previous SFM systems. It is my ultimate goal to develop a SFM system that can be easily used on a robot platform. To reach that goal, the following problems have to be addressed, which are outside the scope of this thesis.

- Automatic feature selection. There exists working algorithms to perform corner detection, interesting point selection, etc. To better serve the purpose of visual navigation, a more sophisticated approach can be developed.
- Large scale driving. In this thesis, we are limited to the range at which most features are

visible throughout the sequences. There is no fundamental limit to extend our approach to larger scale. The issues include how to maintain the structure of features, such as insertion and deletion.

- 3D visual servoing. We start out this research with visual servoing in mind. If we can estimate the 3D structure of scene, 2D visual servoing is no longer needed. The problem becomes how to design a controller given a 3D structure with sample based uncertainty representation.

Appendix A

Structure from Motion with Omnidirectional Cameras

A.1 Introduction

In this chapter, a structure from motion algorithm is developed for catadioptric omnidirectional camera, along with its uncertainty analysis against correspondence noise.

Although the underlying geometry for SFM is well studied, most of the research focus on using conventional cameras. In this chapter, we develop the epipolar constraint for catadioptric omnidirectional camera and present a two-frame SFM algorithm based on it. In Section A.4, we compare the geometric property of the omni-directional camera and conventional camera with experiment results. We show that, in some situations, the omnidirectional SFM can achieve better results than the conventional one even though the omnidirectional image has a lower resolution.

Generally speaking, most existing SFM algorithms can be divided into two categories according to the criterion functions that they seek to optimize. The first one is to parameterize both the camera motion and scene structure, then estimate them by minimizing the criteria based on the distances of the image projections (as in [82] and [1]). The other approach is to develop the criteria based on epipolar constraints. See [96] and [87] for a complete review. The observation is that the image projections are related by the fundamental matrix (essential matrix, for calibrated cameras) which is only a function of the camera motion. Since the later tries to recover fewer parameters given the same data, presumably it is easier and that is the approach we adopt here.

The epipolar constraint can take either a differential form or a normal form. J. Gluckman and S. Nayar [34] addressed the problem of recovering the ego motion from an omnidirectional camera using the differential form epipolar constraint. As it is noted in [**?**],there is a fundamental trade-off between the differential form and normal form. With the differential form, motion between two frames is assumed small so that the differential form epipolar constraint still holds, thus the correspondence problem is easier. On the other hand, with the normal form epipolar constraint, the motion can be arbitrarily large, which complicates the determination of the correspondence, but large motion also means large flow magnitude, which is very desirable for the optimization to be robust against the measurement noise in optic flow. For a robot moving at normal speed with a normal frame grabber, the differential form epipolar constraint usually does not hold in real situations. Therefore we developed our omnidirectional SFM based on the normal form epipolar constraint. To solve the correspondence problem, we can use a multiple target tracking process which finds correspondences over time.

A.2 Two-step omnidirectional SFM Algorithm

We present the algorithm by first introducing the epipolar constraint for an omnidirectional camera. In [80], the epipolar geometry is studied for panoramic cameras with hyperbolic mirrors. In this paper, we develop the epipolar constraint for catadioptric cameras which is simple to compute due to its special geometric property.

If P is a point in the scene, and m_1, m_2 are its projections on two images, then the epipolar constraint holds:

$$m_1^T F m_2 = 0$$

and F is the fundamental matrix. If the camera is calibrated, then m_1 and m_2 are normalized image coordinates, and F is the essential matrix and can be decomposed as $[t]_\times R$ where $[t]_\times$ and R encode the translation and rotation of the camera motion.

Now let us look at the epipolar constraint for omnidirectional cameras as shown in Figure A.1. Here we take a catadioptric omnicam with a parabolic mirror as an example. See Nayar [62] for more details on this type of omnicam.

P is a point in the scene. P_1 and P_2 are its projections on two omnidirectional mirrors. (u_1, v_1) and (u_2, v_2) are its corresponding image coordinates on the two omnidirectional images. We construct the local coordinate systems $X_1Y_1Z_1$ and $X_2Y_2Z_2$ as shown in Figure A.1 then the local coordinates of P_1 and P_2 can be computed as in equation A.1 according to [62]. For other types of omnicam with different mirror shapes, the appropriate projection equation should be used.

$$P_i = \begin{bmatrix} \frac{h^2-u_i^2-v_i^2}{2h} \\ u_i \\ v_i \end{bmatrix} \quad i = 1, 2 \tag{A.1}$$

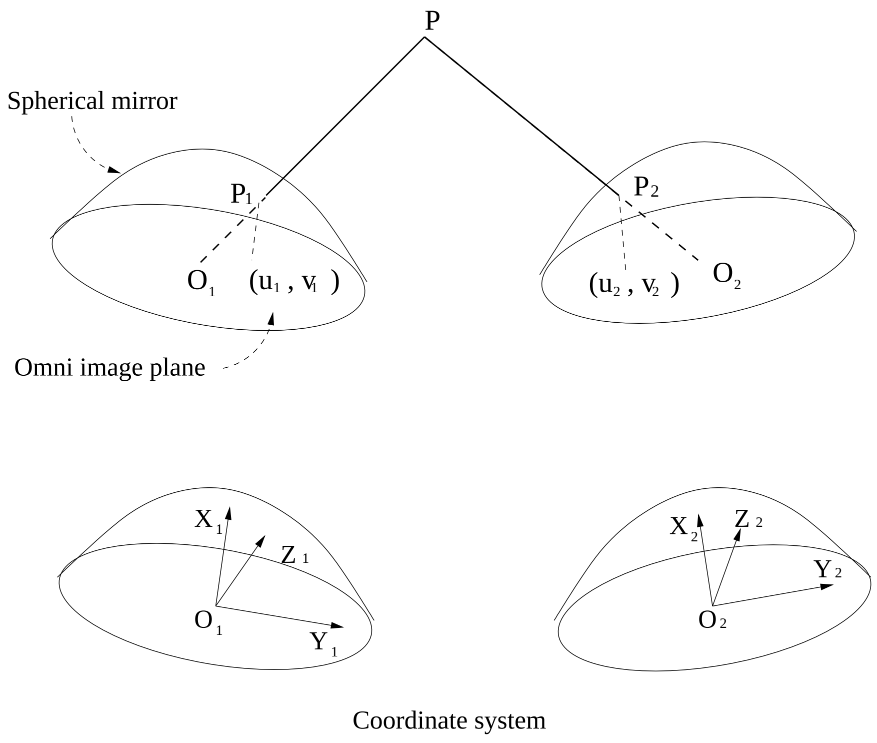


Figure A.1: epipolar geometry of omnidirectional camera

Observing that P, P_1, P_2, O_1 and O_2 are coplanar, then

$$\overline{O_2O_1} \times \overline{O_2P_1} \cdot \overline{O_2P_2} = 0$$

which is equivalent to

$$O_1^2 \times P_1^2 \cdot P_2 = 0 \tag{A.2}$$

where O_1^2 and P_1^2 are the coordinates of points O_1 and P_1 in coordinate system $X_2Y_2Z_2$. Assuming the rigid motion between $X_1Y_1Z_1$ and $X_2Y_2Z_2$ can be described by rotation matrix R and translation vector t, then

$$O_1^2 = R \cdot O_1 + t = t$$

$$P_1^2 = R \cdot P_1 + t$$

Substituting into equation A.2, we get

$$P_2^T E P_1 = 0 \tag{A.3}$$

where $E = [t]_\times R$ is the essential matrix encoding the motion parameters. Given multiple correspondence points, E can be estimated by minimizing the epipolar errors. In [96] Zhang gives a comprehensive review of estimating E for conventional cameras. Since we are more interested in the motion parameters than E itself, we choose to parameterize E with the underlying motion parameters $(t_x, t_y, t_z, q_0, ..., q_3)$, and then recover those parameters directly by minimizing the epipolar errors. (t_x, t_y, t_z) is the translation vector and $(q_0, ..., q_3)$ is the quaternion representing rotation matrix R. Since it naturally enforces the rank constraint of E, it achieves better result than simply estimating E.

We present now the two-step omnidirectional SFM algorithm. Suppose we have a set of point correspondences: $(m_i, m'_i), i = 1, ..., n$, where $n \geq 7$, where $m_i = [u_i, v_i]$ are the image coordinates in the omni image. We can can compute P_i and P'_i according to equation A.1. Like what is usually done for a conventional camera, we construct the epipolar constraint for each correspondence pair and stack them together into one linear system.

$$Uf = 0$$

where

$$U = [\mathbf{u}_1, ..., \mathbf{u}_n]^T$$

and $\mathbf{u}_i$ and f are vectors constructed by stacking columns of matrices $\mathbf{P_i}$ and E, respectively.

$$\mathbf{P}_i = P_i {P'}_i^T$$

.

Step 1: estimate E with linear least square by solving

$$Uf = 0$$

where U is a $n \times 9$ matrix and f is 9×1 vector containing the 9 elements of E.

Step 2: parameterize f as $f(t_x, ..., q_3)$. Use the estimated f in step 1 as a starting point, perform Levenberg-Marquard to solve the following equation:

$$Uf(\mathbf{t}, \mathbf{q}) = \mathbf{0} \tag{A.4}$$

In practice, to avoid the trivial solution, we actually minimize

$$Uf(\mathbf{t}/\|\mathbf{t}\|, \mathbf{q}/\|\mathbf{q}\|)$$

To verify the algorithm, we first test it with synthetic data. A $100m \times 40m \times 15m$ rectangle area is constructed. We simulate the robot moves in a predefined 3D curve as shown in Figure A.2. The circles represent 3D points (50 in total) randomly selected around this rectangle area and the solid curve is the trace of the robot which moves at a speed of 0.5m/s for 20 seconds. We simulate an omnidirectional camera to estimate the motion of the robot every one second. We simulate the tracking uncertainty by adding a Gaussian noise to the computed optical flow. For the results shown in Figure A.3, the Gaussian noise is chosen to be $N(0,1)$. Figure A.3 shows the seven motion parameters estimated by the linear step and nonlinear step along the path. As we can observe, the linear step follows the true values, while the nonlinear step achieves more accurate results in presence of noise.

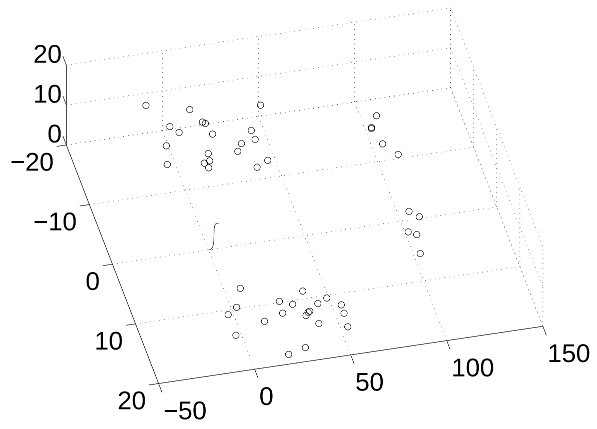


Figure A.2: trace of a robot moving in the scene

A.3 Uncertainty analysis

Since the correspondence measurements are always corrupted with noise, we want to model how the uncertainty of the correspondences propagates into the estimated motion parameters. Following the implicit theorem as in [96], it is possible to derive an analytic solution to compute the uncertainty of the estimation. We summarize the results as follows.

Solving equation A.4 with LM is equivalent to minimize the following criterion function:

$$C(\mathbf{x}, \mathbf{p}) = \|U(\mathbf{x}) f(\mathbf{t}, \mathbf{p})\| \tag{A.5}$$

where **x** is the correspondence measurement and **p**, the motion parameters $(\mathbf{t}, \mathbf{q})$. We can compute the covariance matrix of **p** by equation A.6. Further details can be found in [96].

$$\Lambda_{\mathbf{p}} = \mathbf{H}^{-1}\frac{\partial\Phi}{\partial\mathbf{x}}\Lambda_{\mathbf{x}}\left(\frac{\partial\Phi}{\partial\mathbf{x}}\right)^T\mathbf{H}^{-T} \tag{A.6}$$

where

$$\Phi = 2\left(\frac{\partial f}{\partial\mathbf{p}}\right)^T U(\mathbf{x})^T U(\mathbf{x})f(\mathbf{p})$$

and

$$\mathbf{H} = \frac{\partial\Phi}{\partial\mathbf{p}}$$

An approximation to the covariance matrix is also presented by Zhang [96]

$$\Lambda_{\mathbf{p}} = \frac{2S}{n-k}\mathbf{H}^{-T} \tag{A.7}$$

where S is the value of the criterion function C at the minimum, k is the number of parameters, i.e. 7 here, and n is the number of correspondences. This approximation shows that the uncertainty of the estimated motion parameters $(\mathbf{p})$ depends on the condition number of the Hessian $\mathbf{H}$.

The above analytic solutions provide ways to estimate the uncertainty of the estimated parameters, but they are only valid near the true minimum. The results would be meaningless if the optimization ends up at some wrong local minimum. To cope with this problem we use bootstrap to estimate the uncertainty in our synthetic experiments. Bootstrap is a statistical tool for the computation of covariance matrices, bias and confidence regions for specified estimators. It is a more general alternative to the traditional covariance propagation technique. The book of Efron [26] contains an good introduction. Bootstrap does not require the analytic solution but is data driven and computationally more expensive. It is more appealing here since the covariance analysis is only valid around the true minimum while with bootstrap we statistically evaluate cases when the minimum is not properly found. Bootstrap is also simple to use, we assume Gaussian noise in the correspondence data and create a bootstrap set. We run the two-step SFM for many times (200) and compute the covariance matrix of the recovered motion parameters as its uncertainty measurement. We use bootstrap to generate the results in the next sections.

A.4 Comparison with conventional camera

Obviously omnidirectional camera sacrifices image resolution in exchange for a wider field of view. One obvious advantage of the 360 degree horizontal field of view is that we can track feature points for longer distance with less constraint to the robot motion. Furthermore, one would wonder if this larger field of view would help solve the SFM problem too.

From equation A.6 we can see that the actual uncertainty of the SFM results depends on both the configuration of the scene points and camera motion itself. Assuming **i.i.d.** noise in the flow measurement, a better result would be expected when the flow magnitude is larger due to the increased SNR. If we perform SFM by looking at objects at short range, the conventional camera is no doubt preferred to the omnicam since it generates larger flow with its higher resolution. But when we look at objects far away, SFM becomes more difficult, since either the flow magnitude is very small or the bas-relief ambiguity arises due to the weak perspective effect. The omnidirectional camera bypasses this difficulty by looking at objects at its sides. The actual range beyond which an omnidirectional camera would outperform a conventional camera in SFM depends on the resolution of the images and actual environment. In the experimental setting as in Figure A.2, we find the omnidirectional SFM is better when the objects in front are more than 50 meters away. Figure A.4 shows two flow fields generated by a conventional and an omnidirectional camera respectively.

A.5 Experiment results

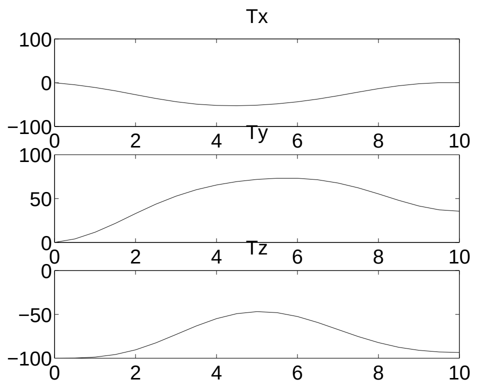
In this section, we present some results on the uncertainty of the motion estimation with respect to flow magnitude and noise level in flow data. The basic observation is that motion estimation generally improves when flow magnitude increases and that the motion estimation degrades when noise increases in the flow magnitude. We also compare results from the omnidirectional SFM with the conventional SFM for a specific situation to verify our discussion in section A.4.

Experiment 1: We simulate the robot with an omnidirectional camera moving 10 meters forward in the rectangle area which is described in section A.2. We assume we can keep tracking all the points along the path and we perform omnidirectional SFM every half meter. The flow data is corrupted with a Gaussian noise, whose variance is set to 1 in the following experiment. We compare the recovered parameters with the ground truth. The error metric for (t_x, t_y, t_z) is chosen to be the angle between the true translation direction and recovered translation direction. The error metric for $(q_0, ..., q_3)$ is the angle between the true quaternion and the recovered one. Absolute mean error and standard deviation are computed for both translation and quaternion after running the algorithm for 200 times. Figure A.5 shows that motion parameters and reconstructed structure improve along the path, which is what we expect to see. In figure A.5(c) A.5(d), the linear method becomes less accurate on rotation estimation along the path.

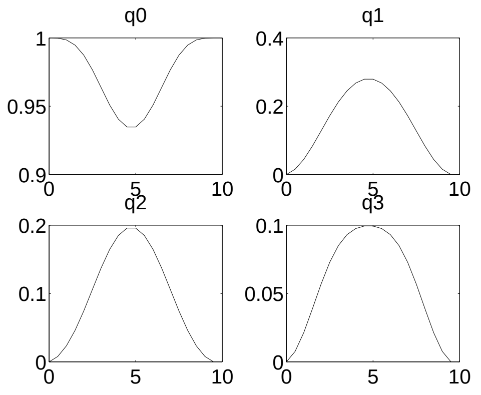
We also show that the reconstruction improves along the path in figure A.6. All the reconstructions are up to a scale factor. The reconstructed points are inaccurate when the translational distance is small, as shown in figure A.6(b) and A.6(c).

We can now compare the performance of the omnidirectional SFM with the conventional SFM using the experimental setting in Figure A.2. Note that the objects in front are 100 meters away and we only move the robot 0.25 meter forward. When the translational distance is small, it is very difficult for SFM with the conventional camera, and it is less difficult for the omnidirectional camera which can look at its sides. Figure A.7 shows that in this case the rotation (quaternion) estimate is comparable for both cameras while the translation estimated by the conventional camera is far off (50 degrees with the N(0,1) noise).

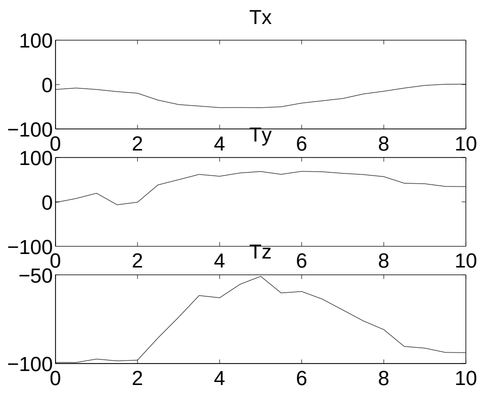
Experiment 3: We also evaluate our algorithm on real images taken with a cadaoptric omnidirectional camera in the outdoor environment. Figure A.8 shows three frames of omnidirectional images taken at different positions along a path. The robot was moving forward between two buildings. The figure A.8(a) is taken at the start point. Figure A.8(b) and A.8(c) are taken at 0.5 meters and 2.5 meters respectively. The circles on the images are feature points manually selected and tracked. The dark points are feature points that are very far away. Figure A.8(d) and A.8(e) show the flow fields. We verified that the translation and rotation are properly recovered. In figure A.9 we show the reconstructed scene points at 0.5 meters and 2.5 meters respectively.



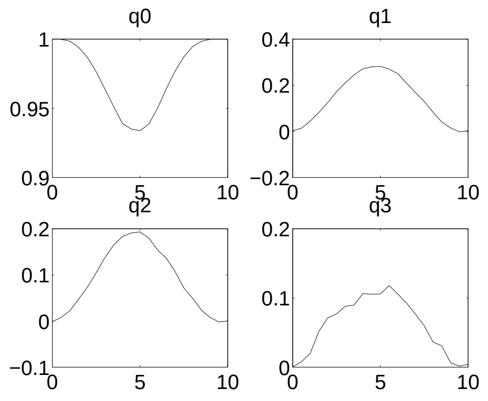
(a)



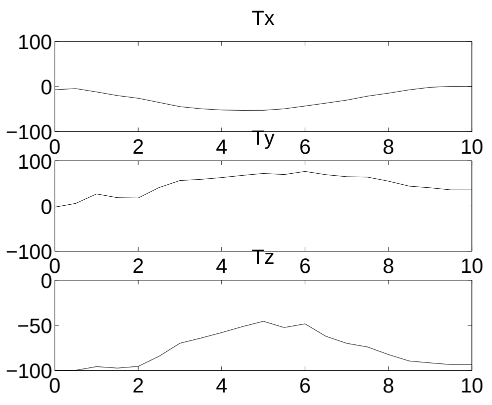
(b)



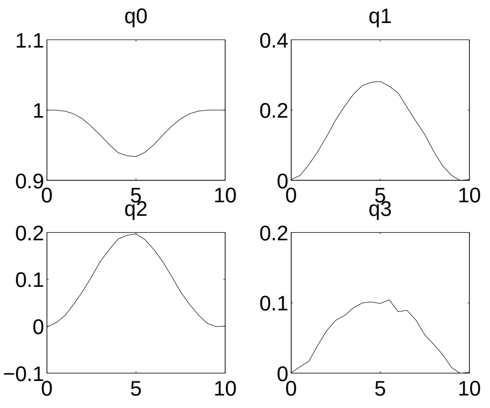
(c)



(d)

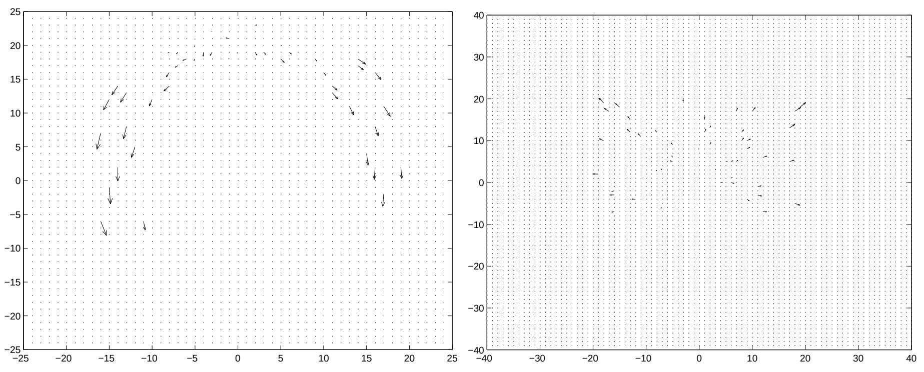


(e)



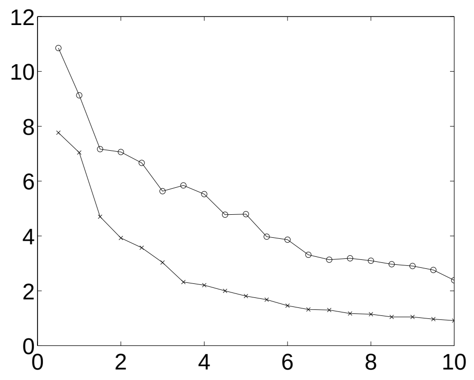
(f)

Figure A.3: Results of Omnidirectional SFM (a)-(b) is the true motion parameters. (c)-(d) show the parameters estimated by linear step. (e)-(f) show the parameters estimated by nonlinear step.

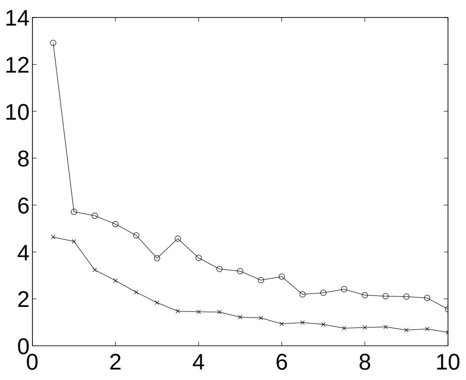


(a) (b)

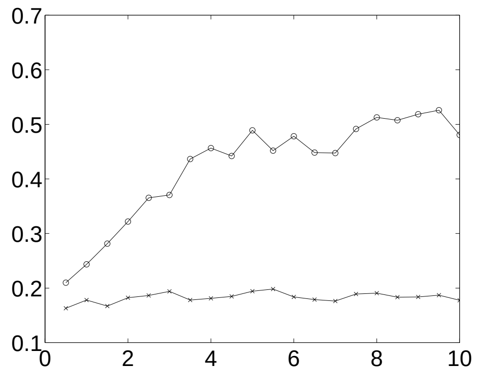
Figure A.4: Two flow fields generated by omnidirectional camera (a) and conventional camera (b). Both move five meters forward. The omnicam looks at points described in Figure A.2 while the conventional camera looks at points fifty to sixty meters away.



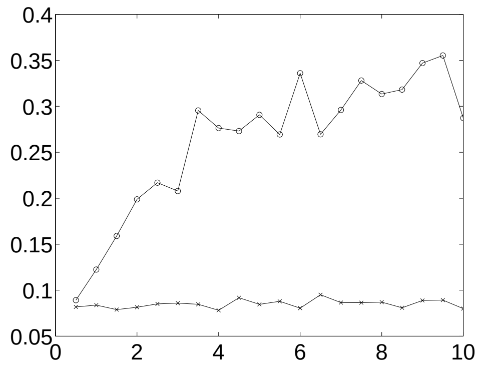
(a)



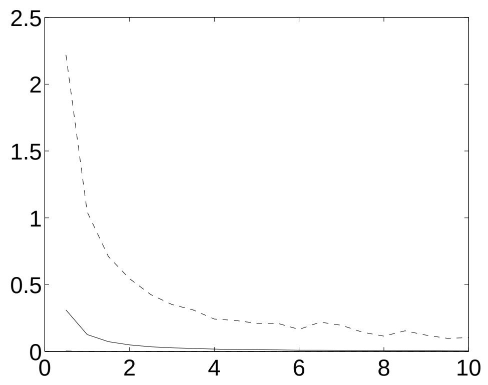
(b)



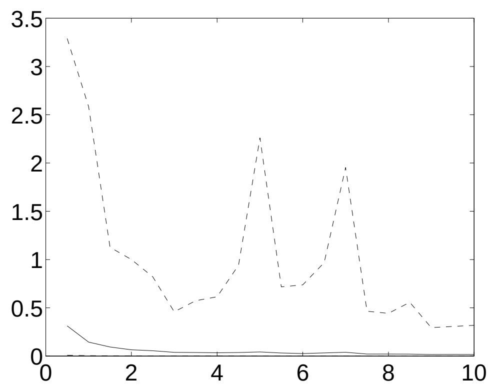
(c)



(d)

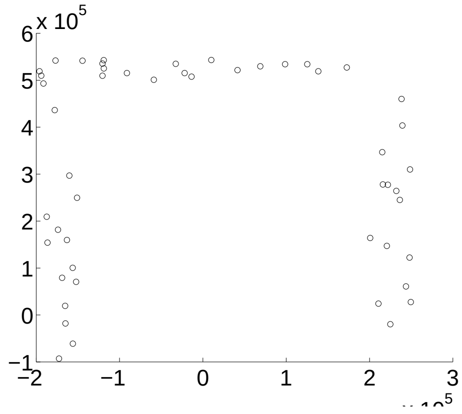


(e)

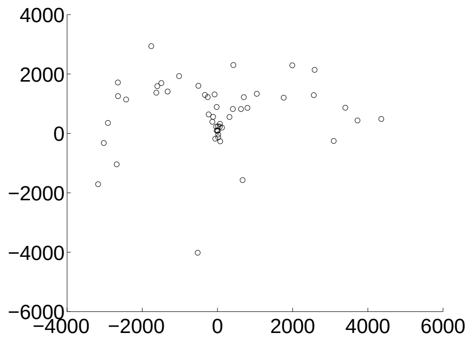


(f)

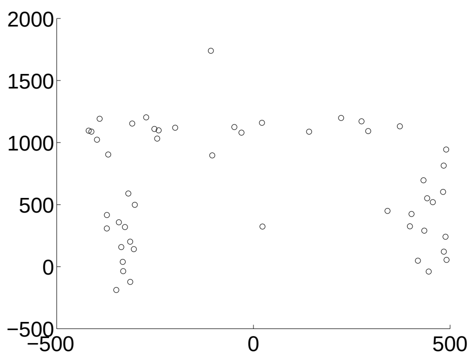
Figure A.5: "o" represents linear step and "x" represents LM optimization. X axis is the distance along the path (a) absolute mean error of translation direction in degree. (b) standard deviation of translation direction in degree. (c) absolute mean error of quaternion direction in degree. (d) standard deviation of quaternion direction in degree. (e) mean reconstruction error (meter), the dash line is the maximum value and the solid line is the mean value (f) standard deviation of reconstruction error (meter)



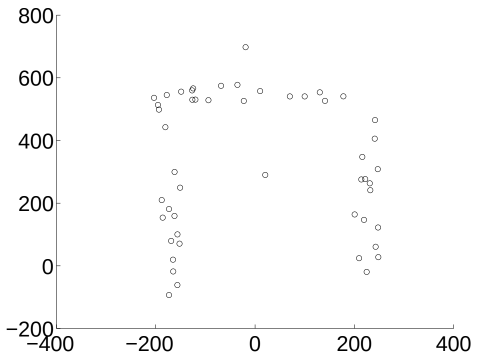
(a)



(b)



(c)



(d)

Figure A.6: ground projection of the scene points (a) true scene (b) (c) (d) recovered scene points at 0.5, 5, 10 meter along the path

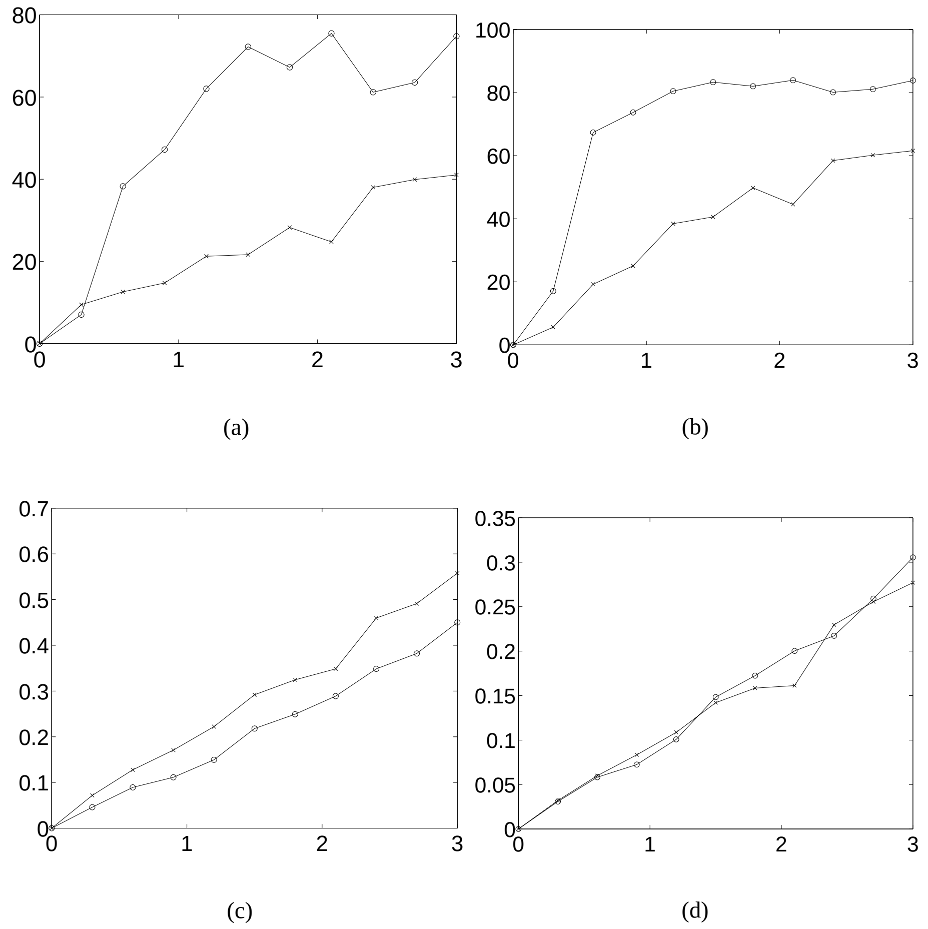
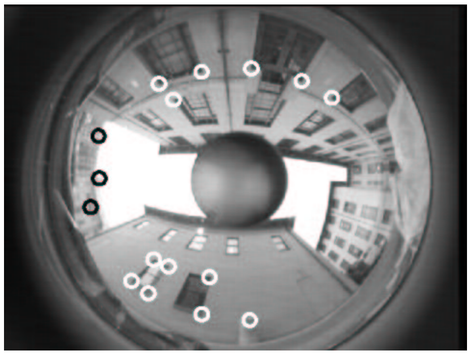
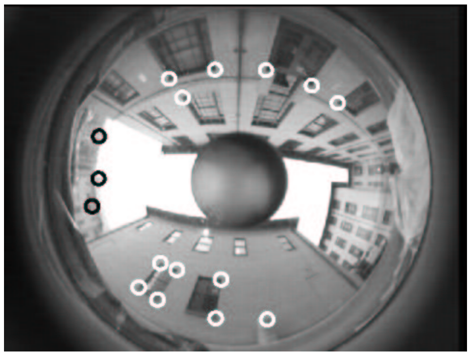


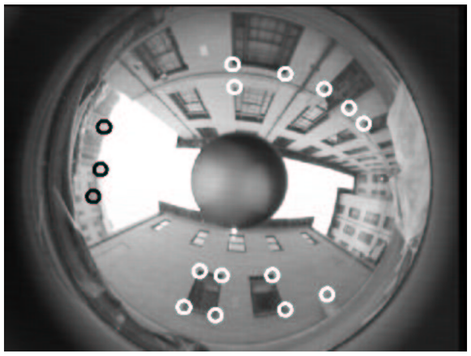
Figure A.7: Comparison between SFMs with an omnicam and a conventional camera when objects are far away. “x” indicates the omnicam and “o” indicates the conventional camera. X axis is the variance of the noise and Y is the actual error in degree. (a) absolute mean error of translation (b) standard deviation of translation (c) absolute mean error of quaternion (d) standard deviation of quaternion estimation



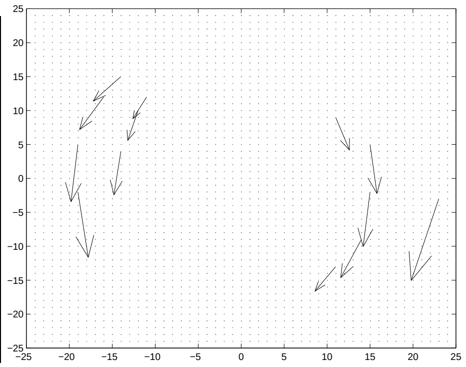
(a)



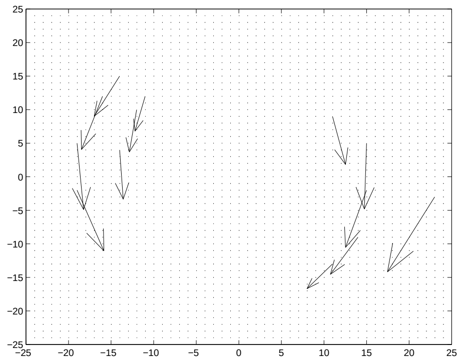
(b)



(c)

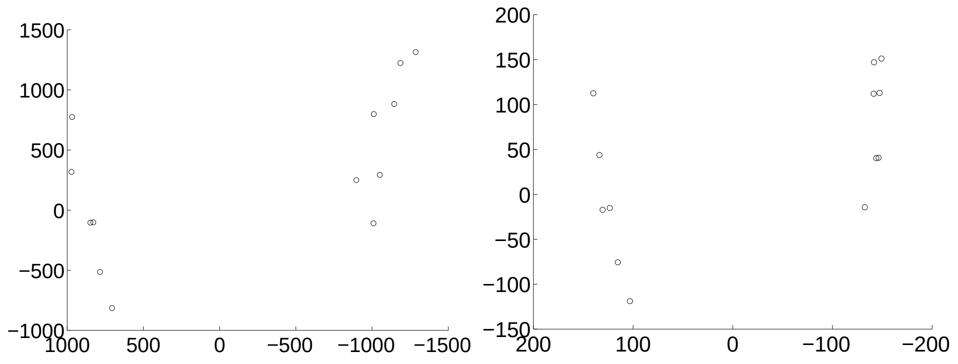


(d)



(e)

Figure A.8: (a) first frame (b) frame grabbed after driving 0.5 meters (c) frame at 2.5 meters (d) omnidirectional flow at 0.5 meters (e) omnidirectional flow at 2.5 meters



(a) (b)

Figure A.9: Reconstructed scene points (up to a scale) (a) reconstruction after driving 0.5 meters (b) reconstruction after driving 2.5 meters

Appendix B

Proof of Convergence

We follow the notations defined in Section 4.3. We need to prove that if sample set X_{k-1} converges to $p(\mathbf{x}_{k-1} \mid \mathbf{I}_{k-1})$ then distribution of Z_k converges to $p(\mathbf{z}_k \mid \mathbf{I}_k)$ and X_k converges to $p(\mathbf{x}_k \mid \mathbf{I}_k)$. We basically follow the ideas from Isard and Blake [47].

The probability of Z_k is $p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1})$,

$$\begin{aligned}
p(\mathbf{z}_k|I_k, \mathbf{x}_{k-1}) &= \int p(\mathbf{z}_k \mid I_k, \mathbf{x}_{k-1} = X_{k-1}) p(X_{k-1}) dX_{k-1} && \text{(B.1)}\\
&\rightarrow \inf p(\mathbf{z}_k \mid I_k, \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{I}_{k-1}) d\mathbf{x}_{k-1} && \text{(B.2)}\\
&= \int p(\mathbf{z}_k \mid I_k, \mathbf{x}_{k-1}, \mathbf{I}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{I}_{k-1}, I_k) d\mathbf{x}_{k-1} && \text{(B.3)}\\
&= \int p(\mathbf{z}_k, \mathbf{x}_{k-1} \mid \mathbf{I}_k) d\mathbf{x}_{k-1} && \text{(B.4)}\\
&= p(\mathbf{z}_k \mid \mathbf{I}_k) && \text{(B.5)}\\
& && \text{(B.6)}
\end{aligned}$$

The probability of X_k is $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{x}_{k-1})$,

$$
\begin{aligned}
p(\mathbf{x}_k|\mathbf{z}_k,\mathbf{x}_{k-1}) &= \int\int p(\mathbf{x}_k \mid \mathbf{z}_k = Z_k, \mathbf{x}_{k-1} = X_{k-1}) p(X_{k-1}) p(Z_k) dZ_k dX_{k-1} && \text{(B.7)}\\
&\to \int\int p(\mathbf{x}_k \mid \mathbf{z}_k = Z_k, \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{I}_{k-1}) p(Z_k) dZ_k d\mathbf{x}_{k-1} && \text{(B.8)}\\
&= \int p(\mathbf{x}_k \mid \mathbf{z}_k = Z_k, \mathbf{I}_{k-1}) p(Z_k) dZ_k && \text{(B.9)}\\
&\to \int p(\mathbf{x}_k \mid \mathbf{z}_k, \mathbf{I}_{k-1}) p(\mathbf{z}_k \mid \mathbf{I}_k) d\mathbf{z}_k && \text{(B.10)}\\
&= p(\mathbf{x}_k \mid \mathbf{I}_k) && \text{(B.11)}\\
& && \text{(B.12)}
\end{aligned}
$$

Bibliography

[1] A. Azarbayejani and A Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transaction on Pattern Recognition and Machine Intellegence*, 17(6), 1995.

[2] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[3] A. Bartoli, P. Sturm, and R. Horaud. Projective structure and motion from two views of a piecewise planar scene. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[4] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierachical model-based motion estimation. In *European Conference on Computer Vision*, 1992.

[5] D.N. Bhat and S.K. Nayar. Stereo in the presence of specular reflection. In *International Conference on Computer Vision*, 1995.

[6] M. Black and A. Jepson. Eigen-tracking: Robust matching and tracking of articulated objects using a view based representation. *International Journal on Computer Vision*, 1998.

[7] M. J. Black. *Robust Incremental Optical Flow*. PhD thesis, Yale University, New Haven, CT, 1992. Research Report YALEU/DCS/RR-923.

[8] M.J. Black and P. Anandan. Robust dynamic motion estimation over time. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1991.

[9] Y. Boers. On the number of samples to be drawn in particle filtering. In *IEE Colloquium on Target Tracking: Algorithms and Applications*, 1999.

[10] J. Bouguet and P. Perona. Visual navigation using a single camera. In *International Conference on Computer Vision*, 1995.

[11] C. Bregler and J. Malik. Tracking people with twist and exponential maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[12] G. Brelstaff and A. Blake. Detecting specular reflections using lambertian constraints. In *International Conference on Computer Vision*, 1988.

[13] T.J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive estimation of 3-d kinematics and structure from a long image sequence. *IEEE Tran. AES*, 26(4), 1990.

[14] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for non-linear problems. *IEEE Proc. Radar Sonar and Navigation*, 146(1), 1999.

[15] P. Chang and M. Hebert. Omnidirectional visual servoing for human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[16] P. Chang and M. Hebert. First results on visual servoing with an omnidirectional camera. *Advanced Robotics*, 14(9), 2000.

[17] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The Confluence of Vision and Control*. Springer-Verlag, 1998.

[18] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Motion and structure causally integrated over time. *IEEE Transaction on Pattern Recognition and Machine Intellegence*, 24(4), 2002.

[19] K. Cho, P. Meer, and J. Cabrera. Performance assessment through bootstrap. *IEEE Transaction on Pattern Recognition and Machine Intellegence*, 19.

[20] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. In *European Conference on Computer Vision*, 1998.

[21] F. Dellaert. *Monte Carlo EM for Data Association with Application to Computer Vision*. PHD thesis, Carnegie Mellon University, 2001.

[22] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *International Conference on Robotics and Automation*, 1999.

[23] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun. Structure from motion without correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[24] A. Doucet, N. Freitas, and N. Gordon (Ed.). *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[25] M. Jordan (Ed.). *Learning in Graphical Models*. Cambridge MA: MIT Press, 1999.

[26] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

[27] O. Faugeras. *Three-dimensional Computer Vision: A Geometric Viewpoint*. MIT press, 1993.

[28] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar enviroment. *Journal of Pattern Recognition and AI*, 2(3):485–508, 1988.

[29] G.S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer, 1995.

[30] D.A. Forsyth, J. Haddon, and S. Ioffe. The joy of sampling. *International Journal on Computer Vision*, 41(1):109.

[31] D. Fox. Kld-sampling: adaptive particle filters. In *Advances in Neural Information Processing Systems 14 (NIPS)*, 2001.

[32] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[33] B. Georgescu and P. Meer. Balanced recovery of 3d structure and camera motion from uncalibrated image sequences. In *European Conference on Computer Vision*, 2002.

[34] J. Gluckman and S. Nayar. Ego-motion anad omnidirectional cameras. In *International Conference on Computer Vision*, 1998.

[35] N. Gordon, D. Salmon, and A. Smith. A novel approach to nonlinear non gaussian bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, 1993.

[36] U. Grenander, Y. Chow, and D.M. Keenan. *HANDS. A Pattern Theoretical Study of Biological Shapes*. Springer-Verlag, 1991.

[37] G. Hager, D. Kriegman, E. Yeh, and C. Rasmussen. Image-based prediction of landmark features for mobile navigation. In *International Conference on Robotics and Automation*, pages 1040–1046, 1997.

[38] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transaction on Pattern Recognition and Machine Intellegence*, 20, 1998.

[39] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2000.

[40] M. Hebert, R. MacLanchlan, and P. Chang. Experiments with driving modes for urban robots. In *SPIE Proceedings on Mobile Robots, Boston, MA*, 1999.

[41] J. Heel. Temporal surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 607–612, Maui, Hawaii, June 1991.

[42] B.K.P Horn and E. Weldon. Direct methods for recovering motion. *International Journal on Computer Vision*, 2(1), 1988.

[43] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[44] M. Irani and P. Anandan. All about direct method. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *International Workshop on Vision Algorithms*, pages 278–295, 1999.

[45] M. Irani and P. Anandan. Factorization with uncertainty. In *European Conference on Computer Vision*, 2000.

[46] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*, pages 343–356, 1996.

[47] M. Isard and A. Blake. Condensation: conditional density propogation for visual tracking. *International Journal on Computer Vision*, 29(1), 1998.

[48] H. Jin, P. Favaro, and S. Soatto. Real-time 3-d motion and structure from point features: a front-end system for vision-based control and interaction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[49] O. King and D.A. Forsyth. How does condensation behave with a finite number of samples? In *European Conference on Computer Vision*, 2000.

[50] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[51] J. Liu and R. Chen. Monte carlo methods for dynamics systems. *Journal of American Statistist Association*, 93:1032–1044, 1998.

[52] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981.

[53] Q. Luong and O. Faugeras. Determining the fundamental matrix with planes: instability and new algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1993.

[54] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *International Conference on Computer Vision*, pages 572–578, 1999.

[55] D. Mackay. Introduction to monte carlo methods. In *Learning in Graphical Models*. The MIT Press, 1999.

[56] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, 1999.

[57] B. Matei and P. Meer. Optimal rigid motion estimation and performance evaluation with bootstrap. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[58] L. Matthies. *Dynamic Stereo Vision*. 1989.

[59] L. Matthies and S. Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, 3(3), 1987.

[60] L. Matthies, Y. Xiong, R. Hogg, D. Zhu, A. Rankin, and B. Kennedy. A portable, autonomous, urban reconnaissance robot. In *Sixth International Conference on Intelligent Autonomous Systems, Venice*, 2000.

[61] P.F. McLauchlan and D. Murray. A unifying framework for structure and motion recovery from image sequences. In *International Conference on Computer Vision*, 1995.

[62] S. Nayar. Catadioptric omnidirectional camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 482–488, 1997.

[63] S. Nayar and V. Peri. Folded catadioptric camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[64] J. Oliensis. A critique of structure from motion algorithms. Technical report, NEC research institute, 1998.

[65] C. Olson, L. Matthies, M. Schoppers, and M. Maimone. Robust stereo ego-motion for long distance navigation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[66] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, 1988.

[67] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipe in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[68] G. Qian and R. Chellapa. Structure from motion using sequential monte carlo methods. In *International Conference on Computer Vision*, 2001.

[69] G. Qian, A. Kale, and R. Chellappa. Robust estimation of motion and structure using a discrete h_∞ filter. 1999.

[70] B. Rao. Data association methods for tracking systems. In *Active Vision*. The MIT Press, 1992.

[71] C. Rasmussen and G. Hager. Robot navigation using image sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1996.

[72] S. Sclaroff and J. Isidoro. Active blobs. In *International Conference on Computer Vision*, 1998.

[73] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[74] H.Y. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal on Computer Vision*, 16(1), 2000.

[75] H. Sidenbladh, M.J. Black, and D.J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *European Conference on Computer Vision*, 2000.

[76] E.P. Simoncelli. *Distributed Representation and Analysis of Visual Motion*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1993.

[77] P. Smyth. Belief networks, hidden markov models, and markov random fields: a unifying view. *Pattern Recognition Letters*, 18:1261–1268, 1997.

[78] S. Soatto and R. Brockett. Optimal structure from motion: local ambiguiety and global estimates. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[79] S. Soatto, R. Frezza, and P. Perona. Motion estimation on the essential manifold. In *European Conference on Computer Vision*, 1994.

[80] T. Svoboda, T. Pajdla, and V. Hlavac. Epipolar geometry for panoramic cameras. In *European Conference on Computer Vision*, 1998.

[81] R. Swaminathan, S.B. Kang, R. Szeliski, A. Criminisi, and S. Nayar. On the motion and appearance of specularities in image sequences. In *European Conference on Computer Vision*, pages 508–523, 2002.

[82] R. Szeliski and S.B. Kang. Recovering 3d shape and motion from from image streams using nonlinear least squares. *Journal of Visual Communication and Image Processing*, 5(1):10–28, 1994.

[83] R. Szeliski and P. Torr. Geometrically constrained structure from motion: points on planes. In *3D Structure from Multiple Images of Large-Scale Environments*, 1998.

[84] C.J. Taylor, J.P. Ostrowski, and S.H. Jung. Robust visual servoing based on relative orientation. In *International Conference on Robotics and Automation*, 1999.

[85] J.I. Thomas, A. Hanson, and J. Oliensis. Refining 3d reconstructions: a theoretical and experimental study of the effect of cross-correlations. *Computer Vision, Graphics and Image Processing: Image Understanding*, 60:359–370, 1994.

[86] J.I. Thomas and J. Oliensis. Dealing with noise in multiframe structure from motion. *Computer Vision and Image Understanding*, 76(2), 1999.

[87] P. Torr and D. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal on Computer Vision*, 24(3):271–300, 1997.

[88] P. H. S. Torr and A Zisserman. Feature based methods for structure and motion estimation. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *International Workshop on Vision Algorithms*, pages 278–295, 1999.

[89] Y. Weiss and E.H. Adelson. Slow and smooth: a bayesian theory for the combination of local motion signals in human vision. Technical Report E10-120, MIT, 1998.

[90] J. Weng, N. Ahuja, and T. Huang. Optimal structure from motion. *IEEE Transaction on Pattern Recognition and Machine Intellegence*, 15(9), 1993.

[91] D. Wettergreen, H. Thomas, and M. Bualat. Initial results from vision-based control of the ames marsokhod rover. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1997.

[92] W. Wilson, C. Hulls, and G. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, 1996.

[93] G. Xu, J. Terai, and H. Shum. A linear algorithm for camera self-calibration, motion and structure recovery for multi-planar scenes from two perspective images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[94] L. Zelnik-Manor and M. Irani. Multi-frame estimation of planar motion. *IEEE Transaction on Pattern Recognition and Machine Intellegence*, 22(10), 2000.

[95] H. Zhang and J.P. Ostrowski. visual motion planning for mobile robots. 18(2):199–208, 2002.

[96] Z. Zhang. Determining the epipolar geometry and its uncertainty: a review. *International Journal on Computer Vision*, 27(2):161–195, 1998.