جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

Senior Honors Thesis

Exploring Visual Odometry for Mobile Robots

Hatem Alismail
Computer Science Department
hatem@cmu.edu

Advisor:
Brett Browning, Ph.D.
Robotics Institute
brettb@cs.cmu.edu

May 3, 2009

Abstract

Visual odometry makes use of an image sequence to estimate the motion of a robot and optionally the structure of the world. The low-cost and small-size of cameras, combined with the high-information content of the images they capture make them ideal for robot platforms. In this work, we develop a visual odometry system based on stereo input. The output of the algorithm is a 3D map of the environment as well as camera/robot trajectory suitable for mobile robots localization and navigation tasks. Our algorithm makes use of Structure-From-Motion techniques by exploiting the projective geometry between 3D point landmarks in the world and their projection into 2D imagery. To establish 2D-3D correspondences over multiple frames, we tracks CenSurE features from a stereo camera. By combining the 3D depth information from the stereo process with robust pose estimation using RANSAC and relative orientation, we show that it is possible to robustly estimate camera pose over time. Furthermore, we show that careful use of Sparse-Bundle-Adjustment (SBA) produces refined solutions that estimate both world structure and camera motion. We compare different approaches within this framework and show that relative orientation is superior to using absolute orientation to estimate pose. Secondly, we introduce a novel track selection process that improves the fault tolerance of SBA to short baseline feature tracks. We test our algorithm on outdoor and indoor environments and present results showing its effectiveness.

Acknowledgments*

A Gb of thanks goes to my advisor for his extreme generosity in his time and resources that he was willing to give me to finish this thesis. Without his help and guidance I would not be able to finish this work. I would also like to thank my family for their continuous and endless support. Many thanks also goes to the CS faculty in Carnegie Mellon Qatar for their dedication and amount of time they are willing to give to teach their students. I would also like to thank Dr. Bernardine Dias for not only teaching me a lot, but also for feeding me something other than CBM's† while I was working on this thesis in Pittsburgh.

I would also like to thank Dr. Dudley Reynolds for his very helpful comments and suggestions for writing this thesis. Thanks also goes to Noura El-Moughny (CS '08) and Noha Al Afifi for their creative suggestions and help in the design of the Meeting of the Minds poster for this thesis. Many thanks goes to Qatar Foundation as well for funding some of the equipment used in this work.

Last but not the least, a big thank goes to the research staff at the Robotics lab in Qatar campus, Wael Ghazzawi, Ameer Abdulsalam and Imran Fanaswala, for their hard work in recovering a hard disk failure of 'scorpion', the lab's beloved server.

*The equipment and computing resources for this paper was funded by the Qatar Foundation for Education, Science and Community Development. The statements made herein are solely the responsibility of the authors and do not reflect any official position by the Qatar Foundation or Carnegie Mellon University.

†Cheese and Bread in Microwave

Table of Contents

1

Introduction

Navigating an unknown environment is a key task for a variety of mobile robots applications. The robot needs to be able to keep track of where it is in the world, a process called *localization*, and simultaneously build and maintain a map of the environment suitable for navigation, a process called *mapping*. As localization and mapping must be performed simultaneously, this problem is commonly known as Simultaneous Localization And Mapping (SLAM).

The SLAM problem has been studied extensively [1, 2], with particular emphasis on LIDAR as the primary sensor. Recently, there has been renewed interest in vision-based SLAM (vSLAM, or visual odometry)*, as cameras offer low cost, high information content sensors that are eminently suitable for human environments. Recent advances in Computer Vision also places within reach a variety of synergistic capabilities, such as object detection, recognition, scene and terrain classification. The main goal of visual odometry is to recover the camera motion and the 3D structure of the world concurrently by exploiting the projective geometry relating multiple views.

In this thesis, we study the vSLAM problem without relying on other sensors. The aim is to be able to use a camera to estimate the robot's *trajectory* as well as to build a 3D model of the world structure. Having the ability to generate maps from camera input only has many advantages. In addition to being integrated seamlessly in our lives, cameras are low-cost, small-sized, low-power consumption and high-information content sensors that makes them ideal for a variety of applications, especially the development of small-sized intelligent units.

Deployment of cameras as a navigation sensor seems to possess many advantages over other sensors. However, there are several issues to be addressed in dealing with visual input. The most important issue is error accumulation and propagation. This issue is not restricted to visual input only. A motion estimation algorithm would necessarily suffer from an accumulated unbounded error, as typical motion estimation algorithms

*We will use the terms interchangeably.

are iterative. However, error propagation is more exaggerated with the noisy nature of images. Further, 3D motion estimation has 6 degrees of freedom (DoF), which makes it more complicated than a 4 DoF problem in a 2D plane. It is important to stress the fact that a vision-based localization and mapping algorithm could benefit from other sensors inputs in an integrated system. In fact, the best vision-based navigation performance, in terms of distance covered, is a combination of motion estimate from a stereo camera as well as an Inertial Measurement Unit (IMU), see [3].

In this thesis, we focus on motion estimation and map building from only visual input in order to advance the core technology. Once a robust camera-based motion and mapping algorithm has been developed, it could be integrated with other sensors to provide a higher level of robustness. In this thesis, we do not claim to achieve the most robust visual odometry system, but analyse the problem carefully and propose possible enhancements that can further increase the robustness of camera-based motion estimation. In particular, the research question we try to address is:

> **Given a stream of images from a calibrated camera, can we estimate the camera position and 3D world structure robustly and efficiently over time?**

The problem is graphically depicted in Figure 1.1, where a typical stream of images is shown as well as an illustrative trajectory.

1.1 Approaches to Visual Odometry

There has been significant interest in visual odometry recently (e.g. [4, 5, 6, 3, 7]). Solutions to the problem can be categorized into two groups: (1) Structure-From-Motion (SFM) techniques, which draw from multi-view projective geometry (e.g. [8, 4, 3]) and (2) Probabilistic Filtering approaches (e.g. [7]), which draw from the state estimation literature and are popular in Robotics. Although filtering approaches, which include the Kalman and Particle Filters, provide accurate estimates, they assume a small number of 3D features/landmarks and do not scale well to larger problems with hundreds to thousands of features. Here, we focus our attention on the SFM-based approaches to visual odometry, with relatively large numbers of features in each image.

Current SFM-based approaches to visual odometry use the following steps:

1. Initialize the model with a set of 3D points as well as a camera pose. Typically, the camera pose is chosen to be the identity rotation and zero translation, although any initial frame could be chosen
2. Match features across frames to build tracks of features suitable for refining the motion and structure. The longer the track corresponding to a 3D point and the larger the baseline between features, the more accurate the refinement is

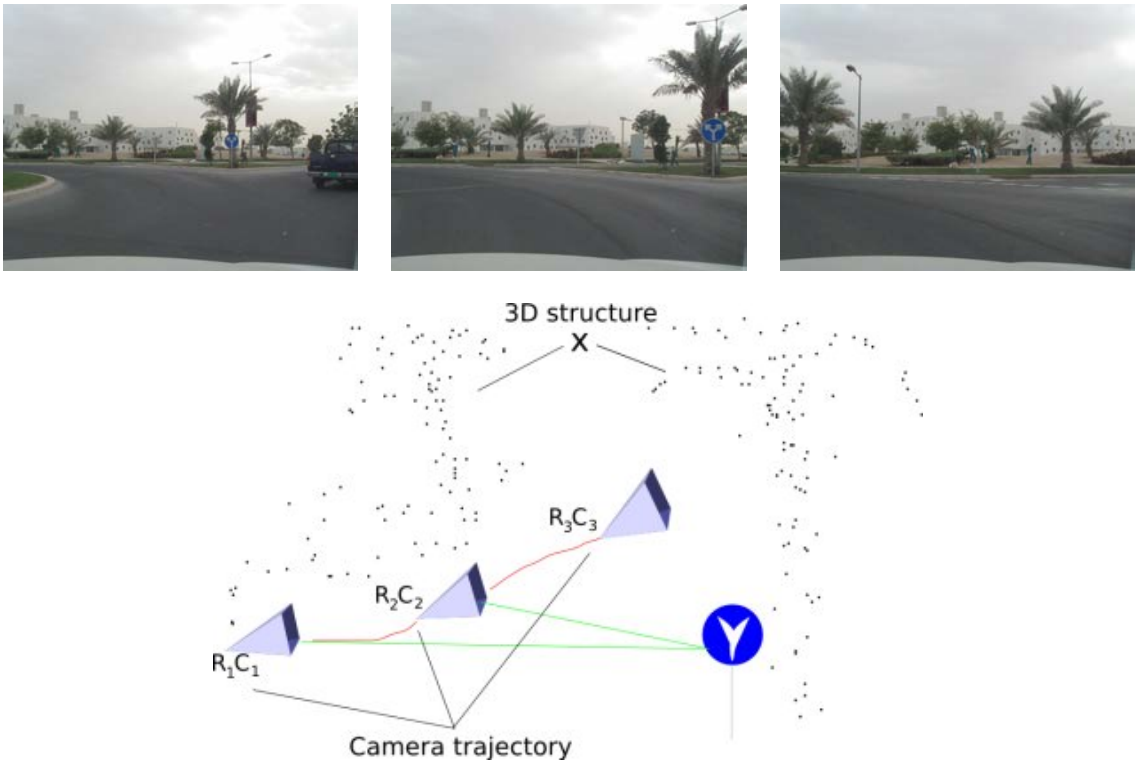


Figure 1.1: Overview of Visual Odometry

3. Triangulate a set of 3D points over frames. The initial set of 3D points will be soon invisible due to camera motion and it is necessary to triangulate additional 3D points to make sure that the algorithm does not run out of 3D points
4. Obtain an initial estimate of motion between pairs of consecutive images. Several approaches could be used to recover the motion between two frames. This step is generally combined with RANSAC [9] for robustness
5. Optionally, refine the estimated motion and structure iteratively using nonlinear minimization methods. This step is very essential for reducing error propagation caused by image noise and the iterative nature of the algorithm
6. Repeat from 2

1.2 Overview of Our Approach

Our algorithm initializes a set of 3D points from stereo using a STereo-On-Chip camera (STOC)[†]. The camera triangulates 3D points using hardware in real-time. After that, a set of CenSurE [10] features is extracted from the left frame form each stereo pair.

[†]From http://www.videredesign.com

Those features are used to initialize a feature tracker. The feature tracker uses Zero-Mean Cross Correlation (ZNCC) and a *marriage assignment* scheme to match features and track them over time. The matched features between consecutive frames are used to estimate the camera motion using *relative orientation* [11]. The process is combined with RANSAC to filter outliers and obtain a consistent set of matches. The final step in the algorithm is the use of *Sparse Bundle Adjustment* (SBA) to refine camera motion and, if possible, the 3D world structure. Structure refinement might not be always possible due to short tracks, or insufficient baseline. Hence, we employ a *track selection* scheme to select the points suitable for refinement.

1.3 Contributions

The main contributions of this thesis include:

- Analysis of reasons that prevent visual odometry from scaling to large datasets
- Empirical experiments proving that the use of relative orientation to estimate the motion of the camera, outperforms the use of absolute orientation [12]
- Introducing a track selection scheme to choose a specific subset of the 3D points to be included in the refinement step, which increases the robustness and accuracy of the algorithm
- Implementation of the visual odometry system[‡]
- Evaluation of the algorithm on indoor and outdoor datasets collected by stereo head mounted on a mobile robot

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 provides an overview of Visual Odometry. This includes a literature review of some current algorithms, related work, different camera types, initialization methods and different results reported from different systems. In Chapter 3, we present the details of the visual odometry algorithm we chose to implement and the motivations behind design decisions.Experiments conducted and evaluations of the proposed approach on indoor and outdoor datasets are presented in Chapter 4. A discussion and analysis of the work done is presented in Chapter 5. Finally, we conclude the paper and discuss possibilities of future work and enhancements on the approach in Chapter 6.

[‡]The system is implemented using unoptimized MATLAB code that runs in near real-time

2

Overview of Visual Odometry

Visual odometry is the process of determining a visual sensor orientation and position in 3D space from a sequence of images, or simply put, motion estimation using visual information only. To us, humans, as well as many other living beings, our perception system provides the main sensory input for navigation purposes. For example, it has been shown that honey bees [13] use *optical flow* [14] as an essential navigation aid. It is not only that visual inputs are naturally suitable for navigation, but also visual inputs allows a variety of useful tasks. For example, besides navigating the environment the robot can generate a 3D reconstruction, detect objects of interest, classify the terrain, etc. All of those tasks can performed with low-cost and low-power consumption, which is ideal for robotics platforms.

In Robotics research, the use of visual input for navigation purposes started late in the 1970's (see [15]). Among the first uses of cameras for mobile robot navigation can be traced back to Moravec's [16], who used several cameras to navigate a robotic cart in a room. However, the use of vision in mobile robotics has been hindered by the limited computational power. Typical image processing and understanding tasks require much computational power due to the amount of data in images, which was not available until recent advances in hardware Computer Vision algorithms.

Nowadays, visual odometry is attracting much attention in the Robotics and Computer Vision communities. Several real-time visual odometry implementations have been reported and results are very promising. However, much work remains to be done in this area. Several improvements and enhancements could be added to current systems to allow them to scale for very large terrains. The main issue is the ability to deal with unbounded accumulated error induced by the iterative nature of motion estimation and exaggerated with the large amount of data in images and the associated noise.

This chapter is organized as follows: First, we introduce some of the camera types that have been used successfully in visual odometry, leading to a review of related work. After that, the assumed camera model and geometry is explained. We also discuss background materials related to feature extraction and 3D structure from images.

Finally, this leads to a discussion of estimating camera motion between frames and the use of RANSAC for robustness.

2.1 Camera Types Used in Visual Odometry

A variety of camera types exist and a survey of all of the different types of imaging devices and lenses is beyond the scope of this thesis. Nevertheless, we can identify three main camera types that have been successfully used in various implementations of visual odometry. Those are shown in Figure 2.1.

The first and the most recommended camera for a robust visual odometry is a stereo camera. The reasons behind the choice of a stereo camera will become apparent when we discuss the main steps in visual odometry, which is mainly the ease of triangulating 3D points. The fixed and known baseline of a stereo camera allows efficient and more accurate triangulation process. However, one drawback of using stereo is the relatively high cost compared to conventional cameras.

Another type of cameras, is the conventional monocular camera. The main motivation for using a monocular camera is the relatively cheap cost and easy deployment. Monocular cameras are being integrated seamlessly in our lives. For example, many cell phones and laptops are now camera-enabled upon purchase.

Finally, there have been some work on visual odometry using an omnidirectional camera, i.e. a camera that has more than 180° field of view (FOV). The very wide field of view from an omnidirectional camera provides many advantages, especially for mobile robots. The most important advantage is that 3D landmark features remain in the field of view of the camera for a longer period of time, which contributes to generating a more dense and better refined 3D model of the world structure.

(a) (b) (c)

Figure 2.1: Different camera types used in visual odometry, (a) monocular camera (http://ptgrey.com), (b) multi-camera omnidirectional system (http://ptgrey.com), (c) stereo camera (http://videredesign.com)

It is not only that different types of cameras have been used to obtain visual odometry, but also the approach has been used in a variety of terrains and different robotics

platforms. One of the most interesting applications of vision-based robot navigation is currently running on the Mars Exploration Rovers, Spirit and Opportunity developed by NASA. According to [17, 18], visual odometry was a "life saver" in NASA's Mars exploration program. The authors report that the left back wheel of the rover "Opportunity" is drawing more current that it should be and it had to be dragged most of the time to maintain its expected lifetime. Dragging the wheel is a major cause of inaccurate wheel odometry and hence visual odometry was there to the rescue.

Back on earth, visual odometry has been used in several terrains and environments, including rough terrains, urban areas, indoor environments, underwater and in air [3, 19, 20, 21]. An interesting example of using vision is the underwater vSLAM algorithm developed in [22]. Another interesting application of vision based navigation system is the one developed for helicopter navigation [23].

2.2 Related Work

The idea of using vision as the main sensor for navigation purposes in not new. The idea has been around since late seventies, early eighties. Some work on SFM started in 1979 by Bonde et al.[15]. In 1980, Moravec [16] demonstrated obstacle avoidance and navigation on a robotic rover using computer vision. However, research on vision-based navigation stalled for a while, due to several factors including the high computational complexity and the existence of more accurate, but more expensive sensors, such as Laser range finders LIDAR, that became the dominant navigation sensor. Recently, however, advances in computer hardware and algorithms are allowing image processing tasks to run in real-time. Such advances are renewing interest in the use of vision for localization and mapping tasks.

Visual odometry has been used in a variety of terrains and environments. The most relevant environments are outdoor and indoor environments accessible by mobile robots. Visual odometry algorithms seem to be more successful in outdoor than indoor terrains as it is easier to extract more unique features compared to features extracted in typical indoor environments. For example, a robot navigating a meadow is more likely to find distinctive features compared to navigating a building. Blank walls and uniform textured floors in indoor environments makes difficult the extraction of distinctive features that could be used for triangulation or tracking. However, a visual odometry system designed to address navigation in indoors environments could benefit greatly from the known Euclidean geometry in indoor environments, such as the straightness of walls.

One of the leading approaches to visual odometry is the work of Nister et al. [4]. The authors demonstrate the use of visual odometry on rough terrains using a monocular as well as a stereo camera. Nister's monocular scheme as well as the stereo scheme operate in real time, partially by tailoring modern CPU instructions to extract features very

efficiently as well as the use of preemptive RANSAC [24]. Preemptive RANSAC plays a key role in the system, adds robustness to the approach by rejecting outliers to the motion model, and it achieves this very efficiently.

The monocular scheme of Nister's approach depends on extracting Harris corners and tracking them across frames. Once a feature track has a large enough baseline, the first and last features are used to triangulate the 3D point, while the rest of features on the track are used in an iterative minimization step later on. Motion between consecutive keyframes of the camera sequence is estimated using the 5-point algorithm [25], followed by an iterative refinement step. The last step is put the current motion estimate in the coordinate system of the previous image to obtain a consistent trajectory of the camera.

The stereo scheme in Nister's work is very similar. However, the fixed baseline of the stereo head simplifies the problem of 3D point triangulation and resolves the scale ambiguity. Dense correlation based stereo is performed to obtain 3D points at every frame and motion is estimated using the 3-point algorithm [11] followed by an iterative refinement step. In both cases, Nister et al. limit error propagation by introducing a *firewall* into the system. The idea is that the motion estimation is stopped after a certain number of frames, and visual odometry is reinitialized. This reduction in error propagation in the system comes at the cost of reducing the number of 3D points used for tracking and refinement and has to be balanced correctly to prevent the system from running out of 3D points.

The use of Bundle Adjustment, or more specifically, Sparse Bundle Adjustment (SBA) has been also used to determine camera motion and world structure without an initial motion estimate. The work of Süderhauf et. al. [26] investigated the use of SBA to estimate camera motion and structure directly from stereo data. This is achieved by using feeding SBA with features from both the left and right stereo pair without a proper initialization of 3D structure or camera motion. The approach is also combined with a simple outlier rejection method to reject points with an error larger than a certain threshold. However, providing good initial estimates of camera motion and structure is essential for seriously motivated applications. The relatively low error rate reported in [26] does demonstrate the power of SBA and its effectiveness in visual odometry applications.

Using SBA with proper initialization of camera motion from point matches using absolute orientation has been implemented by Konolige et al. [6]. The authors show that the use of Bundle Adjustment in visual odometry can reduce errors by 50%. They also combine visual odometry output with an IMU and are able of estimating a trajectory up to 10Km with only 10m of error.

Next, we describe the background materials as well as the basic components needed in a visual odometry system.

2.3 Camera Models

In this work, we assume an ideal *pinhole* camera model. The pinhole camera model is arguably the simplest camera model. Camera projection is essentially a mapping from a 3D world to 2D plane (the imaging surface). The camera matrix, $\mathbf{P}$ is a 3×4 matrix that achieves this mapping. According to the pinhole camera model, $\mathbf{x} \equiv \mathbf{PX}$, where $\mathbf{X}$ is a 3D point in the world represented in homogeneous coordinates, and $\equiv$ denotes equality up to scale. The homogeneous representation of a 3D world point and a 2D image points are $\mathbf{X} = [X\ Y\ Z\ W]^T$, and $\mathbf{x} = [x\ y\ w]^T$ respectively. To obtain the final 2D pixel coordinates one divide the coordinates by w to get $\mathbf{x}_c = [x/w\ y/w]^T$. The use of homogeneous coordinates in Computer Vision is essential for the projection task and very important to be able to seamlessly represent points at infinity.

The camera matrix is usually represented as $\mathbf{P} = \mathbf{K}[\mathbf{R}\ \mathbf{T}]$, where $\mathbf{K}$ is a 3×3 intrinsic parameters matrix, $\mathbf{R}, \mathbf{T}$ are the rotation and translation that transform the world 3D points to the camera frame, or camera extrinsic parameters.

The camera intrinsic parameters matrix provides the transformation from retinal coordinates to image coordinates as depicted in Figure 2.3(b). A typical intrinsic parameters matrix is an upper triangulate matrix in the form

$$\mathbf{K} = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Where:

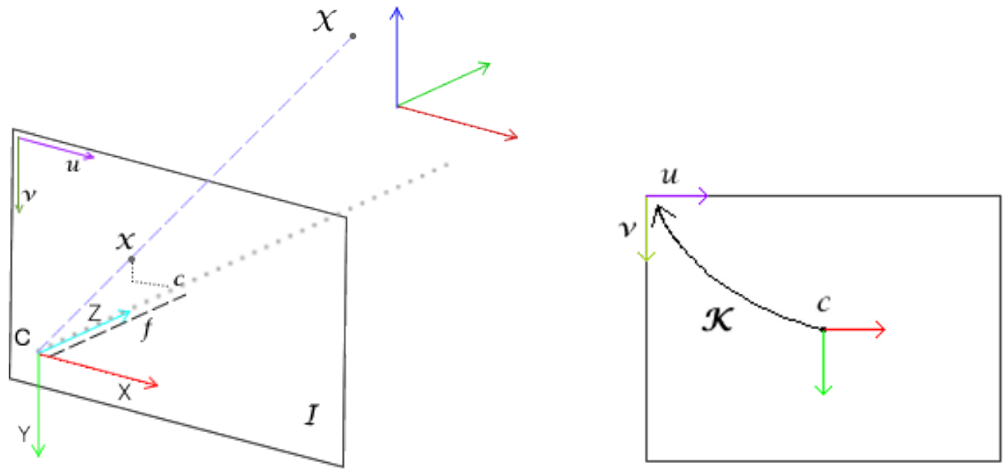
- f_x, f_y are the horizontal and vertical focal lengths respectively
- c_x, c_y are the x, y coordinates of the camera projection center
- α is the camera skew parameters to account for non-rectangular pixels. For the majority of cameras, especially the CMOS camera used in this work, the imaging plane has rectangular pixel, or a very insignificant skew that it is safe to assume $\alpha = 0$.

Note that the camera matrix could also be written as:

$$\mathbf{P} = \mathbf{KR}[\mathbf{I}| - \mathbf{C}] \tag{2.2}$$

where the pixel coordinates of the point $\mathbf{X}$ becomes $\mathbf{x} \equiv \mathbf{KR}[\mathbf{I}| - \mathbf{C}]$. Here, $\mathbf{C}$ is the coordinates of the camera center in the world frame. Comparing this notation with the more convenient notation 2.2, we see that

$$\mathbf{C} = -\mathbf{R}^T\mathbf{T} \tag{2.3}$$



(a) (b)

Figure 2.2: (a) the geometry of the pinhole projection model, (b) transformation between retinal coordinate to image coordinates by the intrinsic matrix **K**

Hence, knowledge of camera rotation,**R**, and translation, **T** allows the computation of the camera center. A more comprehensive treatment of camera models and geometry of camera can be found in [8]. Note, for this exposition, we are glossing over distortions to the moel which we assume can be removed through a good calibration process.

2.4 Image Features

We now describe how image features are extracted and tracked across frames.

2.4.1 Feature detection

Features are points/regions in the image that have desirable properties to make them distinctive and therefor easily detected. The ability to extract distinctive features from images is essential to reducing the amount of information encoded in images. Given the projection of a feature in at least two views, it is possible to reconstruct the 3D location of the point. Further, given a large enough set of feature correspondences between views it is possible to recover the underlying epipolar geometry and perform several tasks, such as camera motion recovery and 3D triangulation.

A single pixel in a typical image is useless without its surrounding texture. Hence, an image feature can be thought of as a vector describing the local image patch surrounding a pixel (see Figure 2.3). This vector is typically called a 'descriptor'. The aim from computing a feature descriptor is to robustly describe the image patch so that comparing

two image patches becomes a distance similarity operation on the two descriptors.

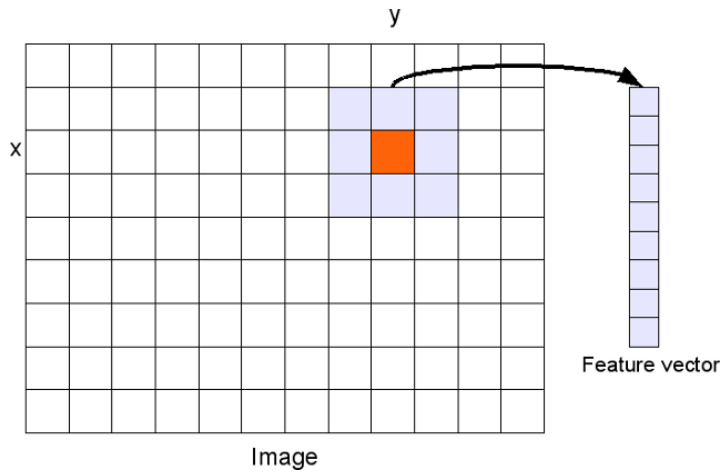


Figure 2.3: Feature vector extract from a 3×3 region centered at pixel (x, y)

Desirable properties of a descriptor include: invariance to illumination, scale changes, viewpoint and affine transformations. However, the more operations applied to increase the robustness of a descriptor, the more computation power needed and the slower the process becomes. Thus, the choice of features and feature descriptors depend highly on the applications and whether real-time performance is needed.

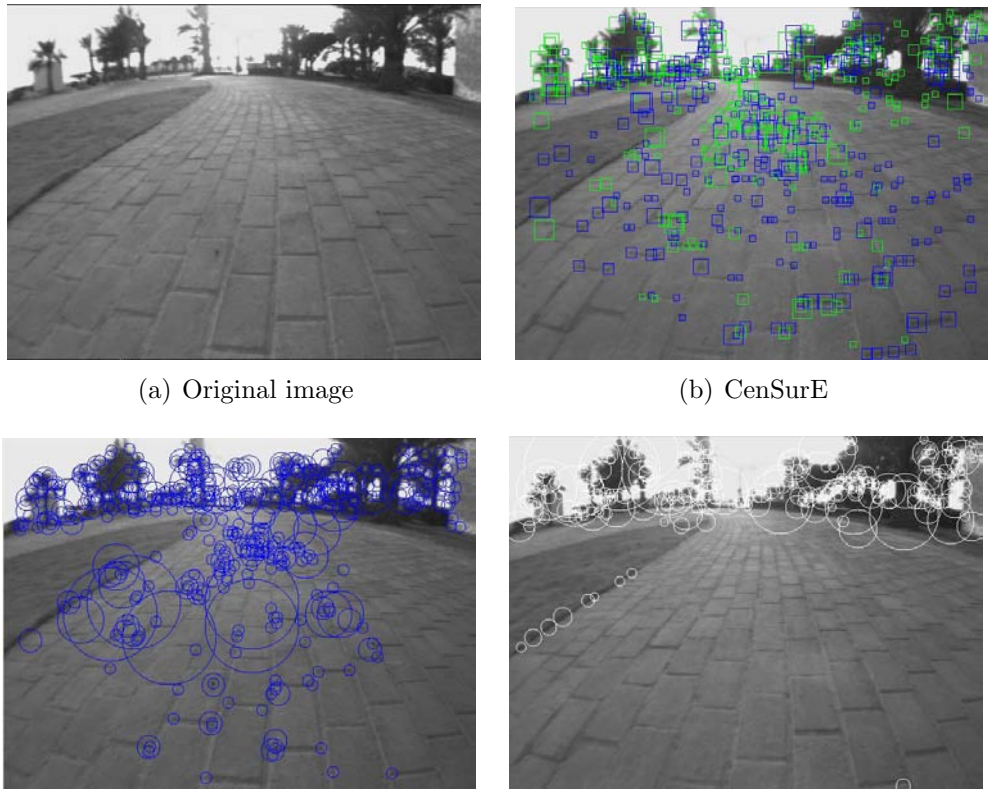
Available feature point extraction and description algorithms include, Harris corners [27], CenSurE features [10], FAST [28], Hessian Affine [29], SURF [30], SIFT [31]. The last two feature detectors, SIFT and SURF belong to the category of affine invariant feature descriptors that generate great feature matching accuracies. However, they require lots of computational power that might not available for high frame rate real-time systems. Hence, simple corner features, that are very efficiently computed, are perfectly valid as the main feature types.

2.4.2 Feature Matching & Tracking

In the context of visual odometry, image features are useful only when they are matched with features from other views. In Structure From Motion the aim is to recover the structure of the world as well as the camera motion from point correspondences in different views. More formally, let $\mathbf{x} \equiv \mathbf{PX}$ and $\mathbf{x}' \equiv \mathbf{P}'\mathbf{X}$ be the projections of the same 3D world point $\mathbf{X}$, then the problem becomes recovering $\mathbf{P}'$ and $\mathbf{X}$. For a calibrated camera, where $\mathbf{K}$ is know, this resolves to finding, $\mathbf{R}, \mathbf{T}, \mathbf{X}$ such that $\mathbf{P}' = \mathbf{K}[\mathbf{R}\ \mathbf{T}]$

Several methods exist to match features between frames. The variation of methods depends mainly on the feature types used as well as application demands. For high dimensional features, such as the 128-d SIFT descriptor, nearest neighbor search using the L2-norm distance is the method of choice using, or a KD-tree for performance. Cross

Figure 2.4: Examples of image features



(a) Original image

(b) CenSurE

(c) Multiscale Harris

(d) SIFT

correlation can also be used to match features. Normalized Cross Correlation (NCC) is a method of matching feature descriptors, which incorporates more robustness into the descriptor in the normalization step.

Once feature matching is done, visual odometry can benefit from maintaining tracks of features, especially in the refinement step. A sufficiently large baseline, or the distance between two projections of a feature will generally guarantee a stable refinement of the reconstruction. However, at high frame rates, two features matched in two consecutive views may not guarantee a large enough baseline. Hence, tracking feature across many frames increases the accuracy of visual odometry considerably. Tracking a feature point across frames could be done with mathematical models, such as the well-known Lucas-Kanade tracker [32], or repeated usage of the previously feature matching methods.

2.5 Obtaining 3D Points/Triangulation

An essential part of the algorithm is obtaining 3D feature points (landmarks) from the camera. Those points are not only for map generation and visualization, but also they serve as a way of obtaining and estimating the camera position. This step is commonly referred to as the initialization step of the algorithm. The way points are triangulated varies with the type of camera. We now present ways of initializing the visual odometry algorithm using a monocular camera as well as a stereo camera.

2.5.1 Triangulation using monocular camera

In the monocular case, 3D points are triangulated using the Epipolar geometry between multiple views. The basic idea is computing the Fundamental Matrix (or the Essential Matrix if calibration is present), to reconstruct the camera matrix relative to another camera assumed to be at the origin with a canonical camera matrix $\mathbf{P} = \mathbf{K}[\mathbf{I}\ \mathbf{0}]$.

Many uncertainties in the use of the Epipolar constraint arise especially when images are taken from a mobile robot. In general, a monocular camera for navigation is mounted on the robot facing the direction of movement such that it can have front view. In this case, the Epipole between consecutive view is at the center of the image, along the direction of motion, which increases the triangulation uncertainty considerably as illustrated in Figure 2.5. Moreover, the reconstruction can only be recovered up to scale.

2.5.2 Triangulation using stereo

The fixed baseline in the stereo case is a great advantage in the process of triangulation. Knowledge of the base line permits an accurate computation of the disparity image and allows for fast and dense 3D triangulation. Moreover, it resolves the scale ambiguity found in the monocular scenario.

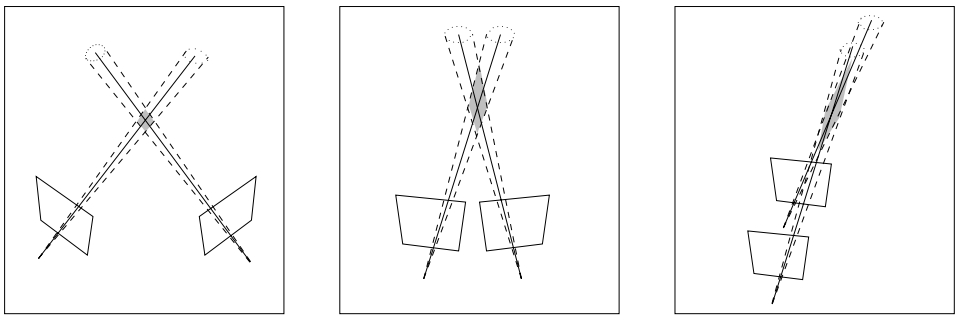
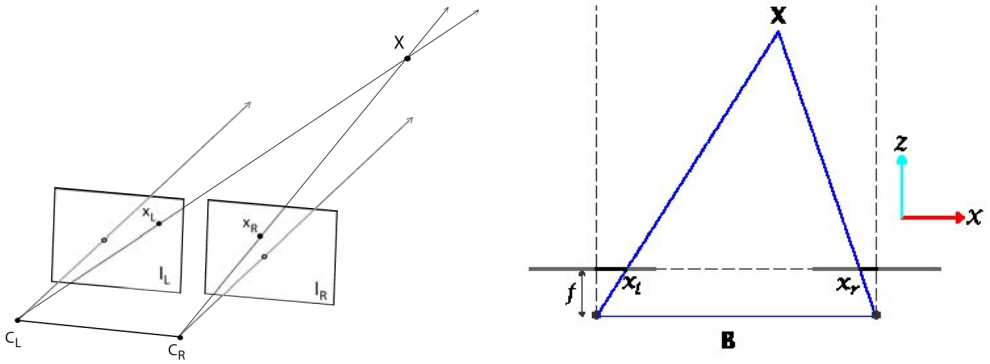


Figure 2.5: The shaded area represents the area of uncertainty in triangulating 3D points from different views. The smaller the baseline (distance between the image planes optical centers) the more uncertain the triangulation. In particular, notice the case in the last image when the two camera are in front of each other. The Epipole in this case is at the center of image and causes the largest uncertainty in estimating the 3D points in the FOV. Figure from [8]



(a) Stereo setup (b) Stereo geometry

Figure 2.6: Stereo vision geometry

Figure 2.7(a) shows a canonical depiction of a fronto-parallel stereo rig. Given the projection ray of point $\mathbf{X}$ on each camera image plane and knowledge of the baseline between the two image planes, we can intersect the two rays in space and estimate the 3D location of the point $\mathbf{X}$ in the camera coordinate system. The process of computing 3D information from the stereo camera starts by computing a disparity image. Since the two image planes in the majority of stereo systems are fronto-parallel, the vertical, y, coordinate of image corresponding image pixels is the same. Hence, it is enough to compute the horizontal disparity, which the difference between the x coordinate of corresponding pixels in each image plane*. Once the correspondence problem between the two view is solved, computing the $3D$ coordinates of a point $\mathbf{X} = (X\ Y\ Z)^T$ can be obtained from trigonometry. From similarity of triangles in Figure 2.7(b), observe that:

$$\frac{Z}{B} = \frac{Z - f}{B - x_l + x_r} \tag{2.4}$$

By rearranging the equation above to isolate Z in on side, we get:

$$Z = \frac{Bf}{\delta} \tag{2.5}$$

where $\delta = x_l - x_r$, the disparity value.

Now, we can obtain the X and Y coordinates of the point using the basic pinhole projection model:

$$X = \frac{Z}{f}x_l \qquad Y = \frac{Z}{f}y_l \tag{2.6}$$

2.6 Recovery of Camera Motion between Frames

To recover camera motion between frames, there are two common approaches: Absolute orientation and relative orientation.

2.6.1 Absolute Orientation

The problem of absolute orientation is the following: Given, at least 3, 3D points in coordinate system A and a corresponding set of points in a different 3D coordinate system B. Find the rotation $\mathbf{R}$, translation $\mathbf{t}$ and a scale factor $\mathbf{s}$ such that $X_A = \mathbf{s}\,(\mathbf{R}X_B + \mathbf{t})$†. The algorithm has been studied extensively and a variety of solution were discovered. In 1983, Faugeras and Hebert [33] published the first solution using unit quaternions to represent rotation. In 1987, Horn published a closed form solution to the problem also using unit quaternions to represent rotation [12]. A solution using rotation matrices was

*Note, we again ignore distortions and skew in the approach as they can be recovered by careful calibration

†Note the use of scaling factor $\mathbf{s}$. For stereo, $\mathbf{s}$ is forced to be 1

presented by Arun et al. [34] using singular value decomposition (SVD). Following to that, in 1991, Umeyama discovered and corrected some degenerate cases [35].

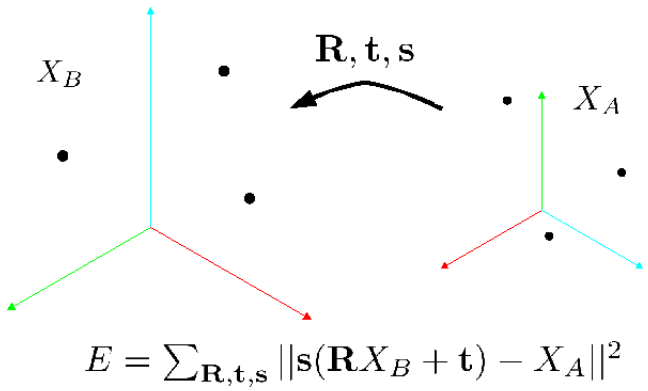


Figure 2.7: Absolute orientation

The basic intuition in most of the absolute orientation algorithms is that solving for $\mathbf{R}$, $\mathbf{t}$ and $\mathbf{s}$ can be done separately. Solving for rotation is the most difficult part in the problem. Once rotation is found, estimation of translation and scale become an easy task. Translation can be estimated between the centroid of X_A and the rotated and scaled centroid of X_B, while scale is handled similarly. Finally, because of the noisy nature of measurements, solutions to the problem are casted as finding $\mathbf{R}$, $\mathbf{t}$, and $\mathbf{s}$, such that an error criteria is minimization.

Typically the quantity to be minimized is:

$$E = \sum_{\mathbf{R},\mathbf{t},\mathbf{s}} ||\mathbf{s}(\mathbf{R}X_B + \mathbf{t}) - X_A||^2 \tag{2.7}$$

This least squares fitting method is very sensitive to outlier measurements. Therefor, the approach should be used as part of hypothesis-and-test RANSAC [9] framework to reject such outliers.

2.6.2 Relative Orientation

Relative orientation is camera motion estimation obtained from a set of 2D-3D point correspondences. The set of 3D points are in the world frame and the 2D points are their projections as seen by a camera. The task is to recover the relative camera position based on the given information. Two key steps are performed to obtain a motion estimate form 2D-3D correspondences. First, the depth, or the distances of each of the 3D points from the camera center is estimated. Second, the Euclidean transformation from the estimated points in camera frame to the world frame is recovered. This transformation describes the camera motion we are interested in. The two steps are explained below.

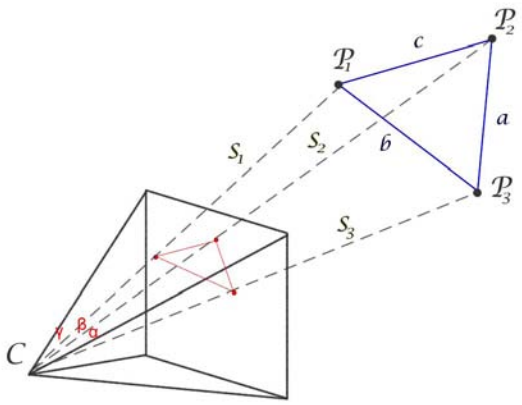


Figure 2.8: Relative orientation

First, estimating depth of a set of 3D points in the camera frame, based on their projection and their coordinates in a reference frame is called the Perspective-N-Point problem (PnP). The PnP problem is well studied problem in photogrammetry and Computer Vision. Solutions to this problem can be traced back to Grunert's work in 1841 [36]. A formal definition of the problem has been given Fischler and Bolles in 1981 [9] as: given the relative spatial locations of n control points and the angle to every pair of control points from the center of perspective (CP), find the lengths of the line segments joining the CP to each of the control points. This is different from the camera resectioning problem [8] where the camera intrinsic parameters are allowed to change. Depending on the number of points used, there exists the P3P, P4P and P5P variants of the problem. Cases with $n < 3$ are not applicable as 1 or 2 points are not enough to constrain the problem and result in an infinity of solutions.

The fundamental problem, and the one we are interested in, is the P3P problem. By looking at Figure 2.8 we observe that there is a tetrahedron consisting of the rays projected from each of the 3D points onto the camera center C. Each of the 3D points are on the form

$$P_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} \tag{2.8}$$

The known lengths of the sides of the triangle connecting the three 3D points is:

$$a = ||P_2 - P_3|| \tag{2.9}$$
$$b = ||P_1 - P_3|| \tag{2.10}$$
$$c = ||P_1 - P_2|| \tag{2.11}$$

The projection of each of the 3D points, from the camera frame, is a 2D point $q_i =$

$\begin{pmatrix} u_i \\ v_i \end{pmatrix}$, where each coordinate is obtained using the perspective equations:

$$u_i = f\frac{X_i}{Z_i} \qquad v_i = f\frac{Y_i}{Z_i} \tag{2.12}$$

The unit vectors from C to each of the P_i's is given by

$$j_i = \frac{1}{\sqrt{u_i^2 + v_i^2 + f_i^2}} \begin{pmatrix} u_i \\ v_i \\ f \end{pmatrix}, \qquad i = 1, 2, 3 \tag{2.13}$$

Since the camera calibration is known, we can determine the angles between each of the rays (tetrahedron legs) and the camera center C. Let α, β, γ be the angles corresponding to a, b, c, line segments connecting the three 3D points P_i's, respectively, then the angles can be given by the following equations:

$$\alpha = \cos^{-1}(j_2 \cdot j_3) \tag{2.14}$$
$$\beta = \cos^{-1}(j_1 \cdot j_3) \tag{2.15}$$
$$\gamma = \cos^{-1}(j_1 \cdot j_2) \tag{2.16}$$

Where, j_i's are the unit vectors given by 2.13. Let S_i be the unknown distance from each P_i to the camera center C, then $S_i = ||P_i||, \quad i = 1, 2, 3$. Since $P_i = S_i j_i$, it is sufficient to find each of the S_i's to determine the position of each of the points in the camera frame.

The main idea to find a solution to the problem is the use of the known information to form a high degree polynomial. The real roots of the polynomial correspond to possible solutions, which are then verified for correctness.

The general approach starts by using the Law of Cosines to get the following:

$$a^2 = S_2^2 + S_3^2 - 2S_2S_3\cos\alpha \tag{2.17}$$
$$b^2 = S_1^2 + S_3^2 - 2S_1S_3\cos\beta \tag{2.18}$$
$$c^2 = S_1^2 + S_2^2 - 2S_1S_2\cos\gamma \tag{2.19}$$

Let

$$S_2 = uS_1 \qquad S_3 = uS_1 \tag{2.20}$$

then:

$$a^2 = S_1^2(u^2 + v^2 - 2uv\cos\alpha) \tag{2.21}$$
$$b^2 = S_1^2(1 + v^2 - 2v\cos\beta) \tag{2.22}$$
$$c^2 = S_1^2(1 + u^2 - 2u\cos\gamma) \tag{2.23}$$

Therefore

$$S_1^2 = \frac{a^2}{u^2 + v^2 - 2uv\cos\alpha} \tag{2.24}$$

$$= \frac{b^2}{1 + v^2 - 2v\cos\beta} \tag{2.25}$$

$$= \frac{c^2}{1 + u^2 - 2u\cos\gamma} \tag{2.26}$$

After this point, several solutions exist [11]. The solution we use in this work is the one given by Fischler and Bolles [9], in which they manipulate the equations above to obtain a fourth order polynomial

$$D_4u^4 + D_3u^3 + D_2u^2 + D_1u + D_0 = 0 \tag{2.27}$$

The polynomial in 2.27 could have up to 4 solutions. For each solution a value for u_i and v_i are recovered. By plugging the u value in 2.26, we can obtain S_1, while S_2 and S_3 can be obtained by Equation 2.20. The correct solution, of of the possible four solutions, must satisfy the Law of Cosines, Equation 2.19. Once the position of each of the P_i's is determined in the camera frame, P_i, then the relative orientation of the camera with respect to the world reference frame can be obtained using absolute orientation as explained in Section 2.6.1.

A complete survey of the major relative orientation/P3P algorithms can be found in the work of Haralick et al. [11]. An efficient $O(n)$ solution to the PnP was recently developed in 2007 by Moreno-Noguer et al. [37].

2.7 RANSAC

Obtaining an estimate of camera motion using either relative or absolute orientation is not the end of the story. The aforementioned algorithms rely on the minimum number of samples to obtain an estimate of motion. However, in the presence of noise, some measurements are gross outliers and hurt the estimation significantly.

RANSAC (RANdom SAmple Consensus) is an algorithm developed by Fischler and Bolles in 1981 [9]. The algorithm's core idea is very simple, but it is probably the most cited algorithm in the Computer Vision literature. RANSAC repeats a set of of steps L times, where at each step a minimal set of points (3 in this case) is sampled at random and the pose is estimated. The number of points that agree with this solution as determined by a threshold are counted, and are called inliers. The pose with the most

outliers is kept. The algorithm runs at most L steps, where

$$L = \frac{\log(p_{fail})}{\log(1 - (p_{good})^k)} \tag{2.28}$$

Where p_{good} is the probability of a randomly selected data point being in a good model and $p_{fail} = (1 - (p_{good})^k)^L$ is the probability of the algorithm failing. While it is possible to use overdetermined solutions, we can see that the running time of the algorithm is proportional to sample size used to generate the hypothesis. Hence, it is important to use the smallest possible sample size for a more efficient solution. Pseudo code for the RANSAC routine is show in Algorithm 1.

Algorithm 1 The RANSAC algorithm

RANSAC(DATA, $N, k, p_{good}, \epsilon, \tau$)

```
1  Select a random minimum number of samples k to compute a model
2  Fit the model to the data set and record the number of inliers that are within a threshold ϵ
3  if k/N > τ
4      then
5          Accept the model
6      else
7          Repeat from 1 L times
8  Cannot find a model, data is too noisy or threshold is too low
```

2.8 Nonlinear Motion & Structure Refinement

Unbounded accumulated error, image noise and the many nonlinearities involved in estimating camera motion require the use of a nonlinear minimization step to reduce error. *Bundle Adjustment* [38] algorithm to refine motion and structure estimates as a large nonlinear minimization problem. The name stems from the analogy of adjusting the 'bundle' of rays projected of 3D points and converging onto each of the camera centers. The goal of the process to adjust the world structure and camera parameters in one bundle. Examples of using the bundle adjustment technique include [5, 3, 19]. Assuming Gaussian noise and the absence of outliers, the bundle adjustment algorithm provides an optimal solution to the problem of refining motion and structure based on image measurements. Further, the sparse nature of the problem makes it possible to use Sparse Bundle Adjustment (SBA) in real-time applications in many times. Bundle adjustment could be run in real-time using local bundle adjustment, mask/feature selection and rejecting outliers to simplify the minimization equations.

The SBA algorithm tries to minimize the reprojection error. The reprojection error

is the Euclidean distance between the predicted projection of all the 3D points using the model and measured projections in the image given by:

$$\min_{a_j, b_i} \sum_{i=i}^{n} \sum_{j=1}^{m} d(\mathbf{Q}(a_j, b_i) - \mathbf{x}_{ij})^2 \tag{2.29}$$

where, n is the number of points, m number of cameras, $\mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})$ is the predicted projection of point i on image j, $\mathbf{x}_{ij}$ is the measured image coordinates and $d(\cdot)$ denotes the Euclidean distance. The minimization problem can solved using iterative nonlinear minimization least squares, such as the Levenberg-Marquardt (LM) algorithm [39, 40]. LM provides a way to iteratively minimize the cost by robustly solving a modified form of *normal equations*:

$$\mathbf{J}^T \mathbf{J} \delta = \mathbf{J}^T \epsilon \tag{2.30}$$

where $\mathbf{J}$ is the Jacobian for the projection function $\mathbf{Q}(\cdot)$. Input to the projection function is a vector of camera parameters $\left(a_1^T, a_2^T, \ldots, a_m^T\right)$, and a vector of points parameters $\left(b_1^T, b_2^T, \ldots, b_n^T\right)$, where $a_i = \left(Q_i^T \; T_i^T\right)^{\ddagger}$ and $b_i = (X_i \; Y_i \; Z_i)$. The output of the projection function is $\left(\hat{\mathbf{x}}_{11}^T, \hat{\mathbf{x}}_{12}^T, \ldots, \hat{\mathbf{x}}_{mn}^T\right)$, which is a set of projection for each of the 3D points in each camera where the point is visible.

The Jacobian of the projection function $\mathbf{J}$ is constructed from $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial a_k}$ and $\frac{\partial \hat{\mathbf{x}_{ij}}}{\partial b_k}$. The spareness of the Jacobian comes from the fact that $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial a_k} = 0$ and $\frac{\partial \hat{\mathbf{x}_{ij}}}{\partial b_k} = 0$, unless $j = k$ because the projection of a point in a camera i depends only on the parameters of camera i and nothing else.

In this chapter, we present the details of our approach to visual odometry.

[‡] Q_i is the imaginary part of unit quaternion corresponding to rotation

3

Our Approach

Our approach is inspired by several algorithms in the field [5, 3, 6, 4]. The approach does not place any constraints on the camera (such as forcing the estimated motion to be above the ground plane) and does not incorporate any predictive models on the camera motion. The motion of each frame is computed using an efficient relative orientation computation in a hypothesize-and-test RANSAC [9] framework. The iterative nature of this motion estimation scheme requires careful attention to error propagation. We make use of the *B*undle Adjustment technique to refine motion and possibly the world structure (see Hartley and Zissermen [8]). Bundle Adjustment is considered the method of choice for optimal refinement of motion and structure if the error is modeled as a zero-mean Gaussian. In this chapter, we explain the details of our visual odometry algorithm. Figure 3.1 outlines the main steps in our approach and Figure 2 is our algorithm in pseudo code. The core algorithm is currently implemented in MATLAB while feature detection and tracking is implemented in C/C++.

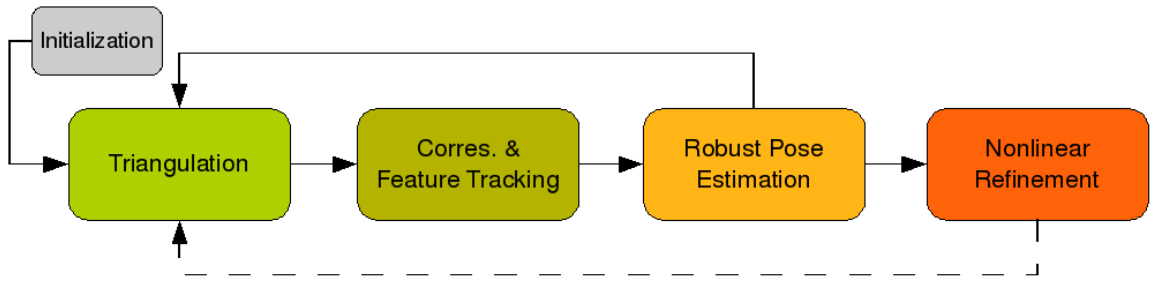


Figure 3.1: Overview of the main steps in Visual Odometry. The dotted line denotes a less frequent use of nonlinear refinement depending on the need and available image projections

The algorithm starts with initializing the world model with an initial set of 3D points from stereo and aligns the world coordinate system with the first camera, i.e. $Q_0 =$

Algorithm 2 Overview of the visual odometry algorithm, details in the text

VISUAL-ODOMETRY($\mathbf{K}$)

```
1  [X, x] = INITIALIZE-FROM-STEREO
2  while there are images to process
3        do
4           ▷ Obtain data from the stereo head
5           [X_m, Y_m, y_m] = FINDCORRESPONDENCE(x,y)
6           [Q_i, T_i, inliers] = RANSAC-RELATIVE-ORIENTATION(X_m, Y_m, y_m, K)
7           [Q̂_i, T̂_i, X_m(inliers)] = SBA-MOTION-REFINEMENT
8           if there are tracks longer than 3 frames staring from frame k
9             then
10                ▷ Refine motion & structure using SBA
11                [Q̂_k:i, T̂_k:i, X_s] = SBA-MOTION-AND-STRUCTURE
```

Figure 3.2: The STOC camera used in the system

$(1\ 0\ 0\ 0)^T$ *, $T_0 = (0\ 0\ 0)^T$, where Q_i is camera rotation represented as a quaternion. After that, a three-step processes is then repeated: (1) finding a set correspondence between 3D points and their 2D features in the left image of the stereo, (2) obtaining a robust estimate of motion using RANSAC and relative orientation, (3) refining motion parameters only using SBA. To increase the accuracy of the approach, we sub-select features that are tracked for more than three frames to refine motion and structure concurrently using SBA.

3.1 Initialization / 3D Points Triangulation

In this work, we delegate the task of triangulation and obtaining a set of 3D points at every frame to hardware using a Stereo On Chip Camera (STOC)†, which performs dense correlation-based stereo. The camera has 9cm baseline and is connected to the computer using Firewire, see Figure 3.2.

The small baseline as well as the relatively wide FOV of the camera do not allow the

*Equiveantly $\mathbf{R}_0 = \mathbf{I}_3$

†From Videre design http://www.videredesign.com

camera to accurately triangulate 3D points that are far away from the camera. During empirical experiments with the camera, we observed that points that are farther than $5m$ from the camera have unreliable depth and are not used in the algorithm. This has been in a problem in testing the approach for outdoors datasets.

3.2 Finding Correspondence and Feature Tracking

The STOC camera tries to find a dense set of 3D points by matching as many points as possible between the left and right image. Knowledge of epipolar lines allow a fast and dense set of features to be matched resulting in a dense 3D point cloud. However, the high density of features used for triangulation introduces more challenges to feature matching across frames. For the approach to be successful, it needs more distinctive features that could be matched between frames with higher confidence. Thus, we extract different types of features from the left image of each stereo frame to use them for motion and structure refinement.

Once features are extracted, a 3D-2D correspondence needs to be established between the extracted features and the 2D coordinates of the 3D points generated by the stereo head. The correspondence approach used is very simple by matching the coordinates of the extracted features to the 2D coordinates of stereo head. In this work, we extract CenSurE features [10] from the left image of each stereo pair, see Figure 2.4 for an example of CenSurE features. CenSurE features were extracted in scale-space to add invariance to feature scale [41]. The use of CenSurE features has two advantages: one, computing CenSurE features in an image is a relatively efficient task, thus providing means for a real-time implementation. Two, CenSurE features seem to perform well in outdoor as well as in indoor environments.

Each extracted feature is assigned a unique ID to be able to build tracks of features over time. For each image I_i a number of CenSurE feature vectors v_i's are extracted. Similarly, for image I_{i+1} we extract a number of feature vectors v_{i+1}'s. Two feature vectors are matched if the Zero Normalized Cross Correlation (ZNCC Equation 3.3) exceeds a certain threshold. Further, the spatial information is also incorporated by limiting the search window around the feature vector to a certain number of pixels. Moreover, features are matching in a marriage assignment scheme, to obtain a more reliable matching. Each of the feature vectors in v_i is assigned a matching feature vector in v_{i+1} based on the ZNCC score and spatial information. At the same time, each of the feature vectors in v_{i+1} is assigned a matching vector in v_i, again using the ZNCC score and spatial information. Finally, two feature vectors v_i, v_{i+1} are declared as matches if both features have been assigned to each other by the marriage assignment scheme.

As a note, the choice of the many different thresholds involved in the process (e.g. search window size, ZNCC threshold, scale factor between every octave, etc.) is rather

$$\frac{1}{N-1}\sum_{x,y}\frac{(I_{x,y}-\mu_I)\,(w-\mu_w)}{\sigma_I\sigma_w} \tag{3.1}$$

Figure 3.3: Zero-mean Normalized Cross Correlation (ZNCC), where $I_{x,y}$ denotes the subimage, w is the feature vector, N is the number of entries in the feature vector, μ the mean and σ is the covariance

cumbersome. Choices are highly dependent on the image sequence on hand and are a matter of art and experimentation to get the best combination of magical numbers. In this work, we decide the best combination of thresholds for each datasets separately by experimentation and visual verification.

3.3 Camera Motion Estimation using Relative Orientation

To obtain an initial estimate of the camera motion we use a 3-pt relative orientation algorithm [11]. We combine the relative orientation algorithm with RANSAC [9] to reject outliers and obtain a robust pose estimates. The input to the 3-pt relative orientation algorithm is a set of 3D points in the world coordinate frame $\mathbf{X}_i$, a corresponding set of 3D points in the camera coordinate frame $\mathbf{Y}_i$, a set of 2D image projections $\mathbf{y}_i$ corresponding to each of the $\mathbf{Y}_i$'s and the camera intrinsic parameters matrix $\mathbf{K}$. Since we have the calibration matrix, we can work directly in normalized image coordinates to reestimate the depth of each of the points $\mathbf{Y}_i$ and obtain a more accurate set of 3D points $\hat{\mathbf{Y}}_i$ in the camera coordinate system. After that, we apply an absolute orientation step to recover $\mathbf{R}$ and $\mathbf{T}$, such that $X_i = \mathbf{R}(\mathbf{Y}_i - \mathbf{T})$ (see Section 2.6.2).

The use of relative orientation instead of directly recovering the camera motion using absolute orientation step is motivated by the uncertainties in computing the depth of a point from stereo. Although both relative and absolute orientation are sensitive to noise, the latter is empirically more stable. Thus, reestimating the points as a preprocessing step before the absolute orientation step yields better results (see Chapter 4).

3.4 Motion & Structure Refinement

The step of refining motion and structure is crucially important. The unbounded nature of the accumulated error is the main reason preventing visual odometry algorithms to scale. As can be seen, there is much room for errors involved in every step of the algorithm influenced by the noisy nature of images. Sources for error in the system include:

- Triangulating 3D points from stereo: the process of triangulation is not error-free

and error could be introduced during feature matching steps. Further, more gross errors could easily be introduced to the system by the nonlinear proportionality of disparity uncertainty with respect to depth.

- Feature tracking: the feature tracker is also not error-free. In fact, one of the most devastating errors are wrong image matches. However, a good percentage of those errors are handled by RANSAC

- Errors in initial motion estimates: again, initial motion estimates are prune to errors. The major cause of error in this case are wrong feature tracks as well as inaccurate 3D triangulations. We try to handle as much as possible of those errors by using RANSAC as well as reestimating the 3D points using relative orientation prior to apply an absolute orientation step.

The problem of motion and structure estimation is essentially non-linear. Although the use of an outlier rejection scheme as RANSAC is extremely important, a nonlinear minimization/refinement step should be applied. In this work, the Bundle Adjustment [38] algorithm to refine the motion as well as the structure whenever possible. In this work, we use the Sparse Bundle Adjustment implementation provided by Lourakis et al. [42]. We have observed that it is almost always possible to refine estimate motion using bundle adjustment, however care should be taken when using the bundle adjustment method to refine structure. The choice of the 3D feature points to be included in the bundle adjustment computation should be made carefully. An important factor in determining whether a 3D point is included in the refinement is the number of corresponding image projections. More precisely, the baseline between those projections is the quantity we would like to have as large as possible to obtain an accurate reconstruction. However, consecutive images passed to the algorithm are of actual image motions. Thus, we assume that the number of frames we track a feature for is a good indication for sufficient camera movement to allow accurate reconstruction. In our experiments, we include a 3D point into the refinement step iff we can see the point for more than 3 frames. In other words, the point is projected into at least 3 cameras.

In the next chapter, we describe experiments done and report on results obtained from this algorithm in outdoor and indoor datasets.

4

Experiments & Results

In order to test and evaluate the performance of our approach, we have collected stereo imagery taken by a camera mounted on a mobile robot in indoor and outdoor environments. The camera used is STOC camera shown in Figure 2.2(c) with 9cm baseline and a wide Field of View (FOV). The short baseline of the camera as well as the wide FOV favored the use of the camera in indoor environments with limited scene depth. We have also tested the approach in outdoor environments, but with the camera titled towards the ground to get more useful 3D points.

We experimentally tested major components of the visual odometry system. In particular we experiment with:

- Relative orientation vs. Absolute orientation
- Global vs. Local SBA
- Visual odometry without nonlinear refinement
- Feature selection mechanism
- Complete system results on indoor and outdoor datasets

Next, we explain experiments conducted and report results of testing the algorithm on outdoor and indoor datasets.

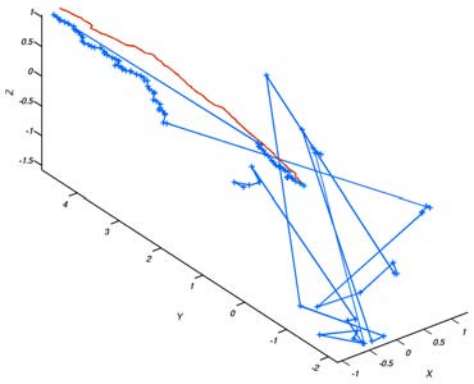
4.1 Relative orientation vs. Absolute orientation

Since 3D points can be triangulated from stereo in camera frame and can be matched to a set of points in the world frame, why not use absolute orientation directly? In absolute orientation, a transformation between corresponding 3D points in different coordinate

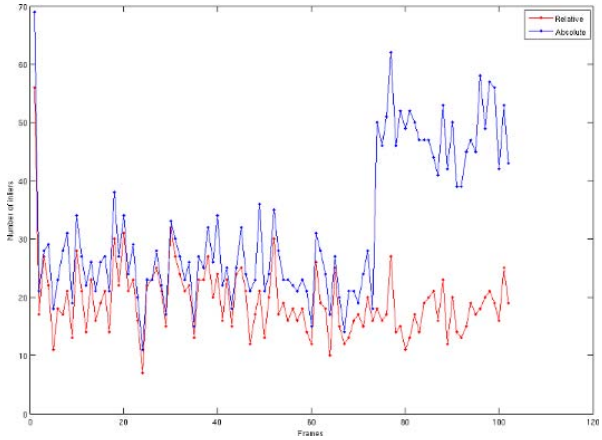
systems can be computed efficiently and in closed form [12]. Nonetheless, relative orientation proves superior in this problem. In our experiments, we have tested pose estimation using relative orientation and absolute orientation and we have experimentally found that relative orientation provides more stable results.

Both relative and absolute orientation are very sensitive to erroneous measurements. Although the use of RANSAC adds robustness to the algorithm, it does not solve the problem of inaccurate measurements. The situation is more exaggerated in the case of triangulating 3D points from stereo imagery, especially when 3D points triangulation uncertainties increase significantly in the direction of depth.

In Figure 4.2(a) the trajectory of the camera was estimated using relative orientation (red) versus absolute orientation. Both methods seem to perform well until the 73rd frame of the sequence in which trajectory estimation using absolute orientation becomes useless. Figure 4.2(b) is a plot of the cardinality of the set of inlier 3D points at every frame of the sequence. From the plot, we can see that absolute orientation is able to maintain a slightly higher number of inliers at every frame. More interestingly, at frame 73, when motion estimation using absolute orientation breaks, we can see a significant increase in the number of inliers. This high number is due to a low error when the absolute orientation model is fitted to data that is should not be consistent with motion.



(a) Relative (red) vs. Absolute orientation trajectory. See text for details



(b) Number of RANSAC inliers over time

Figure 4.1: Relative vs. Absolute orientation results

Since the trajectory estimation is iterative and depends on previous estimates, it is not possible to recover the correct trajectory once it fails. It is possible to establish another segment of odometry, however, there is no easy way to connect those segments without using other sensors or more complex algorithms. Accepting inaccurate points as inliers hurts the algorithm significantly and our algorithm does not provide a solution to this problem. Instead, we use relative orientation to compute a robust estimate of motion between frames that correctly rejects 3D points that are outliers to the motion

model based on the geometric reprojection error.

4.2 Global vs. Local SBA

Bundle Adjustment is an established algorithm in the field of photogrammetery and provides an optimal refinement of motion and structure assuming that error is zero-mean normally distrusted. However, naive implementation of the algorithm is very computationally expensive as it requires inversion of very large matrices. Even with the introduction of fast matrix inversion methods, the size of the matrices are typically large preventing the use of Bundle Adjustment in real time. The structure of the problem, however, is naturally sparse. Exploiting this sparsity leads to significant gains in performance and allows careful use of the algorithm to run in real-time.

Nonetheless, the matrix inversion step is still a burden when the problem grows and we are dealing with large matrices. In solution to this problem, local Sparse Bundle Adjustment (SBA) could be used instead of running the SBA globally on all frames. In fact, the iterative nature of the problem does not allow SBA to run globally as images are not known beforehand.

In our experiments, we have used SBA to refine all frames whenever a new frame is added, which we call global SBA, and we have used SBA to refine only a fixed number of frames locally. Results obtained from local SBA are of comparable quality to using global SBA. In fact, this minor loss of quality is compensated by the ability refine motion and structure much faster than refinement using global SBA.

4.3 Visual Odometry without nonlinear refinement

We have analyzed the performance of visual odometry using relative orientation combined with RANSAC without any use of nonlinear refinement and we have found that results depend on the dataset. In particular, relative orientation provides acceptable results when camera rotations are small, other than rotations along the direction of depth. If camera motion consists of translation only, then the use of relative orientation generates a trajectory and map that are adequate for mobile robot navigation purposes. However, as soon as rotations are introduced, the nonlinear refinement step is essential. Results for this experiment are in Figure 4.3 and 4.2.

Figure 4.2 shows the trajectory of the robot driving in a line up a gentle slope. Red is the trajectory estimated using SBA, while the other is the trajectory estimate by relative orientation only. In both algorithms, the X and Y coordinates of the trajectory report that the robot did not move neither left or right. However, as the robot was going up, relative orientation overestimated the robot's rotation.

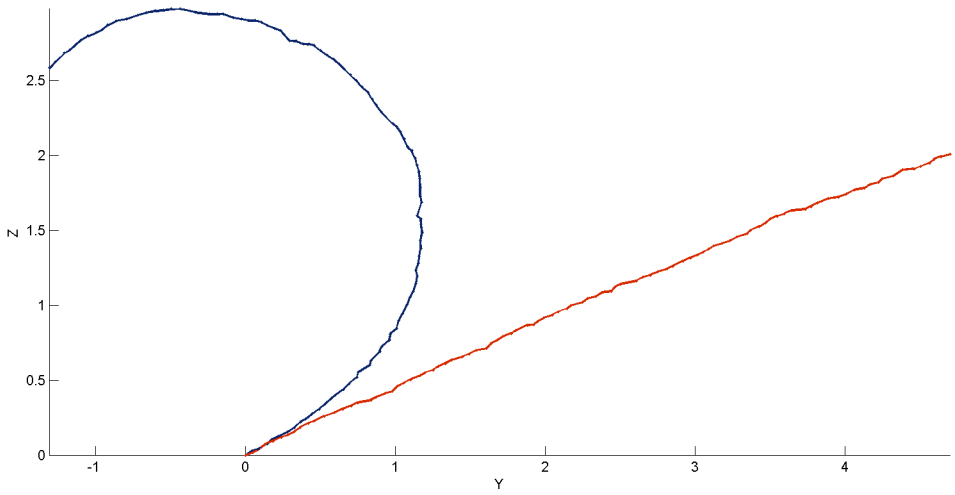


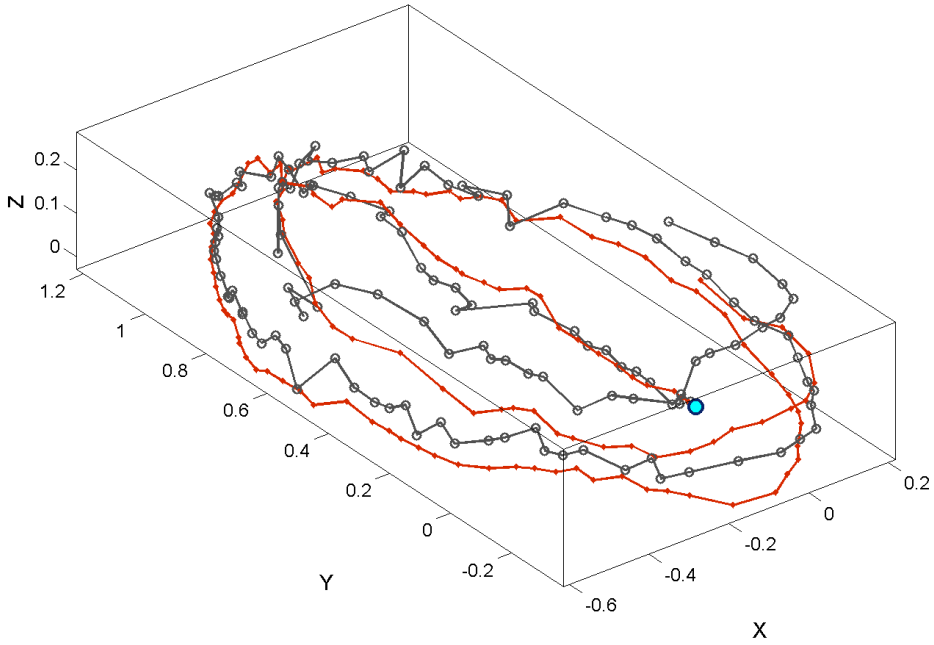
Figure 4.2: Relative orientation is prone to errors. Nonlinear refinement (red) is essential for obtaining accurate results. See text for details.

Nonlinear refinement is also required in the case of rotations. Figure 4.3 shows SBA trajectory (in red) versus relative orientation only on a dataset collected by moving the camera in circular trajectory by hand.

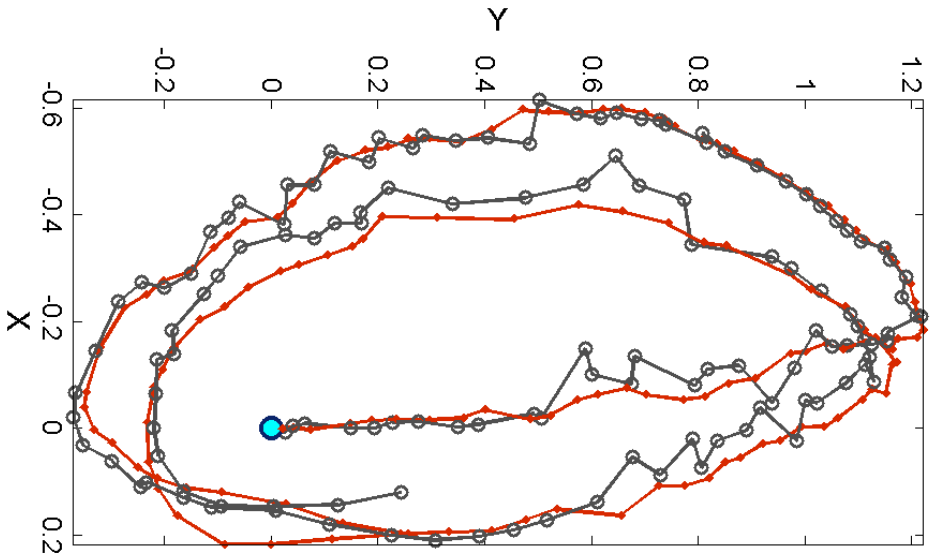
More analysis is shown in Figure 4.4 where the mean re projection error is plotted before applying SBA and after applying SBA. In this plot, SBA is ran on all frames in the sequence in a global fashion. The spike in the error plot was due to large rotation that caused blurry image affecting the quality of features. However, after motion refinement using SBA the error goes back to an acceptable values.

4.4 Feature Selection Mechanism

Information about 3D points and their projection is stored in a matrix, where rows of the matrix correspond to 3D points identified by a unique feature identifier and columns correspond to frame number in the image sequence, as shown in Figure 4.5. The matrix is binary, such that $M_{i,j} = 1$ if the 3D point with the feature i is visible at camera frame j. Following the convention of [42] we refer to this matrix as the visibility mask, or mask for short. The mask contains a lot of information about the features and the performance of feature tracking. The example mask shown in Figure 4.5 is a diagonal matrix as one would expect. As the camera moves in space, 3D features are expected to disappear from the field of view. Hence, features observed on the first frame might not be visible at the 20^{th} frame depending on how fast the robot moves, camera frame



(a)



(b)

Figure 4.3: Large camera rotation requires the use of nonlinear refinement

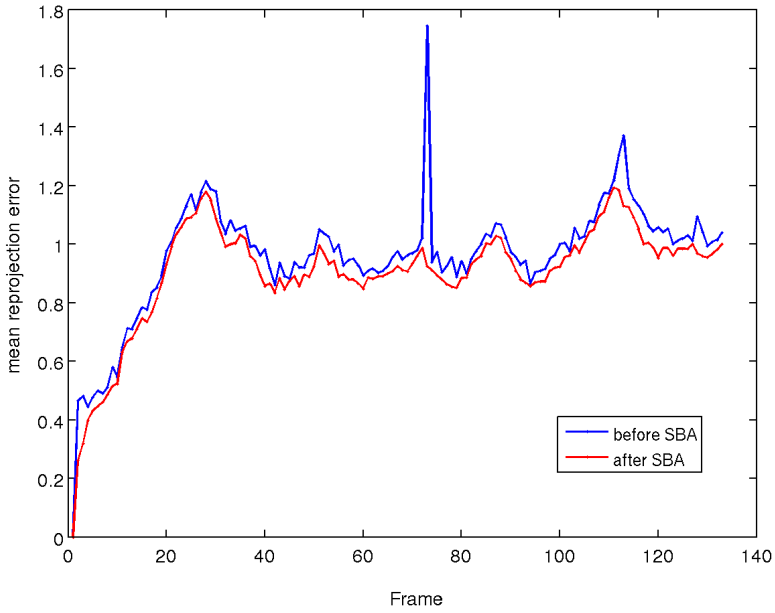


Figure 4.4: Mean reprojection error at every frame before and after applying a global SBA motion refinement. Motion refinement only was used and no structure refinement used to generate the figure

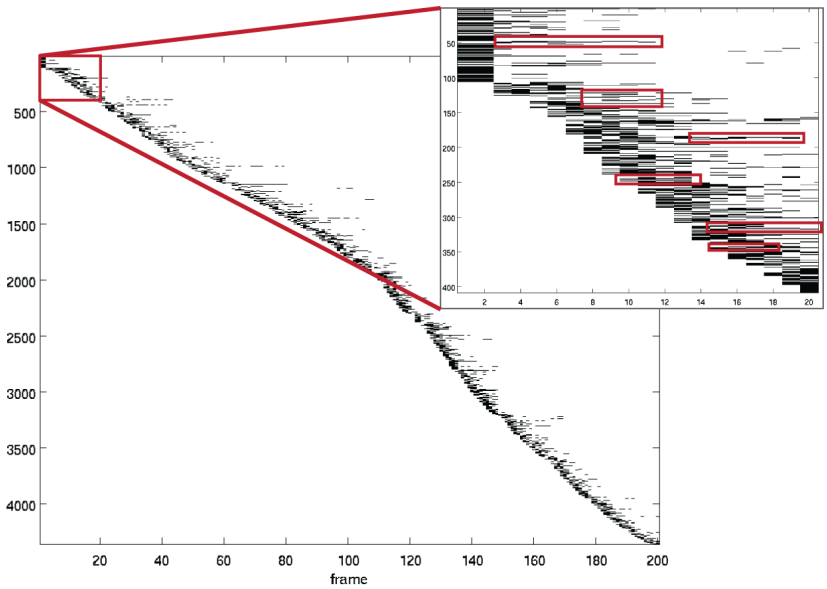


Figure 4.5: Example of a visibility mask, the red rectangles in inside the close up are some of the included points for nonlinear refinement

rate, and feature tracking performance.

One merit of the visual odometry approach is that it does not require a feature to be visible at every frame. However, features that appear in very few frames, e.g. features visible in 2 frames only, should be treated with care in the nonlinear refinement step as they often do not constrain the refinement significantly.

To accurately refine a 3D point location in space it is necessary to have a sufficient baseline. Most of the features that appear in two frames only are very close to each other and not suitable for refinement. Refining those points will most likely result in hurting the performance of the algorithm significantly. In our experiments, we noticed two devastating behaviors if points with only two projections are used. The refinement could either scale the 3D points coordinates down very close to zero, or it could scale the coordinates up to values very close to infinity, in order to minimize error.

In this work, we select 3D points that are seen in more than 3 views for inclusion in a joint motion and structure refinement step. Otherwise, we refine motion only, leaving the structure intact, as shown in Figure 4.5.

4.5 Complete system results on indoor and outdoor datasets

We have tested the approach in indoors and outdoors environments. In this section we present results on an outdoor dataset taken by a mobile robot. The robot, shown in Figure 4.6, is a modified ER1 robot from Evolution Robotics*. A sample from one of the outdoor datasets used to test the algorithm is shown in Figure 4.9. This dataset is particularly challenging because of the abrupt robot motion caused by the bricks on the ground and the small-sized robot wheels. The visual odometry results from that dataset are shown in Figure 4.10.

Figure 4.6: The ER1 robot, one of the robots used to collect data

*www.evolution.com

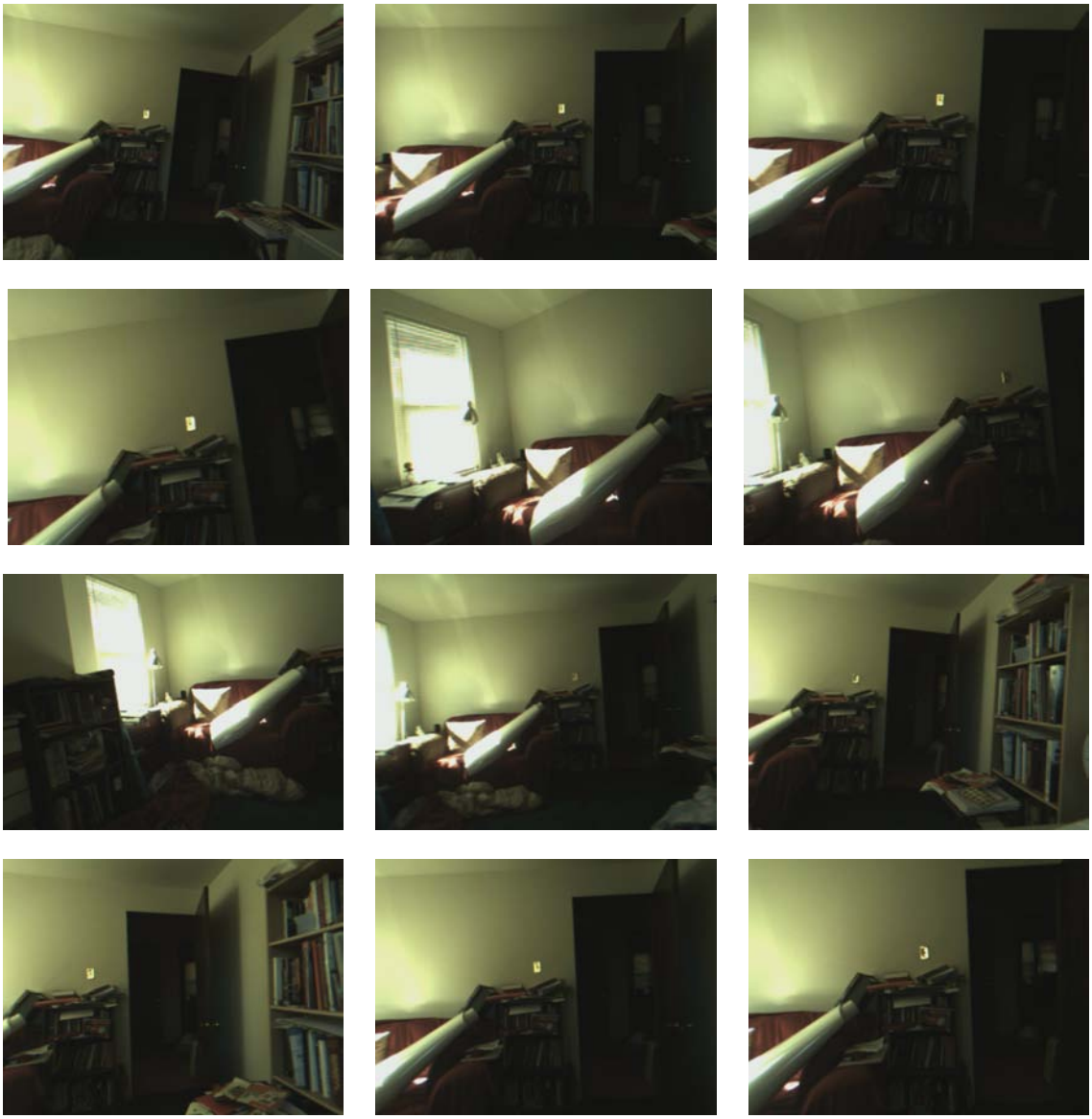


Figure 4.7: Sample frames from an indoor dataset

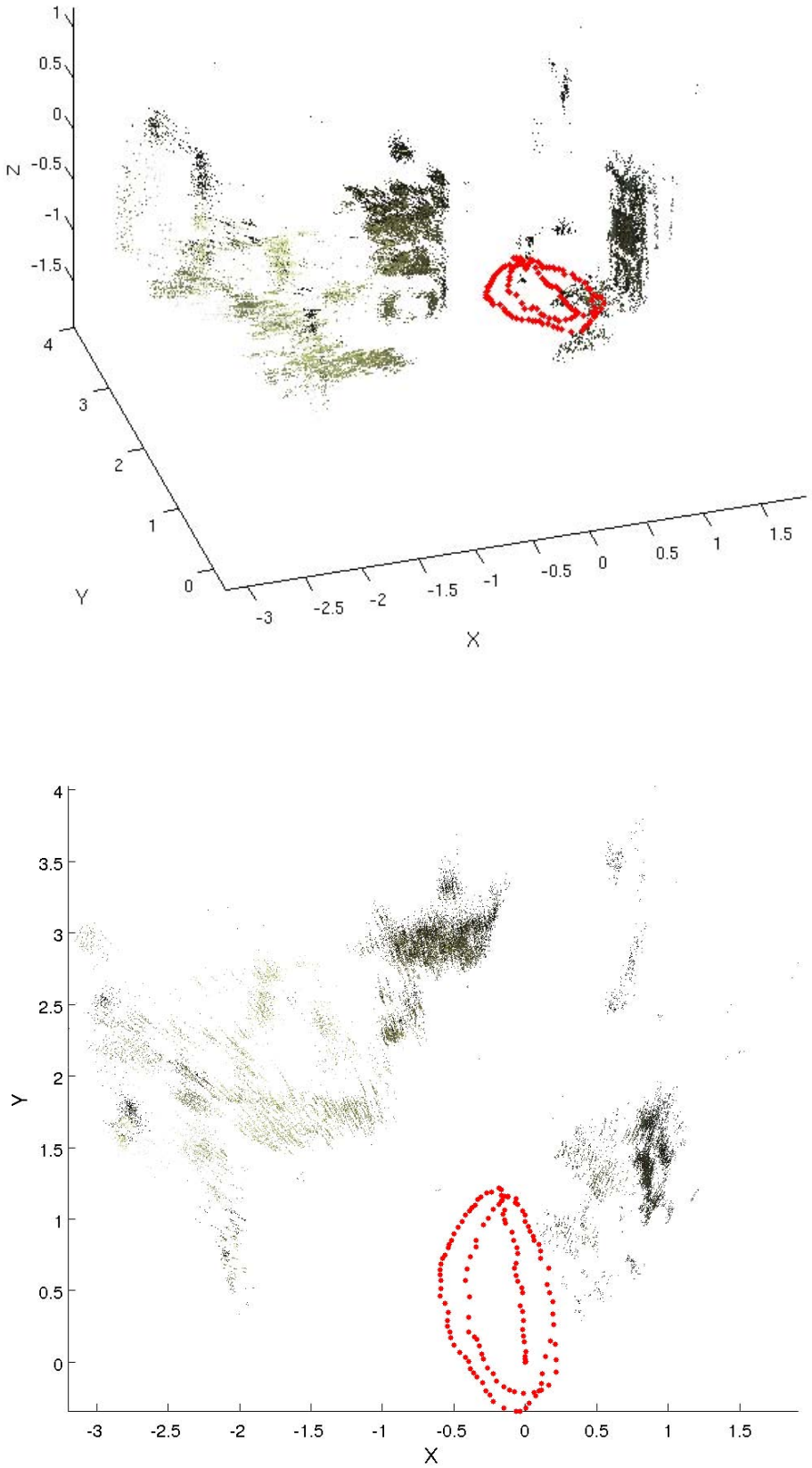


Figure 4.8: Results on one of the indoor datasets, showing the circular trajectory and 3D world texture

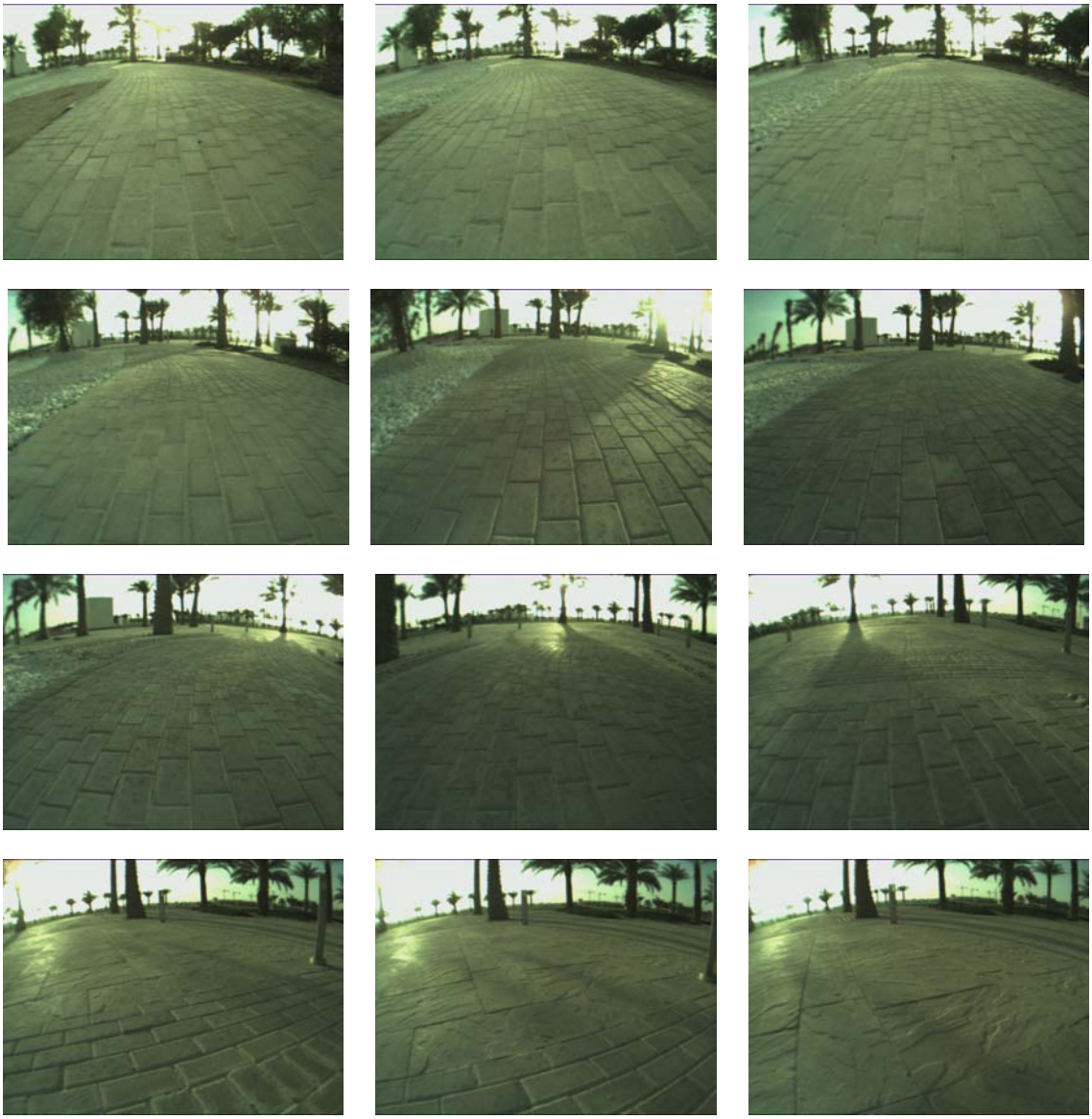
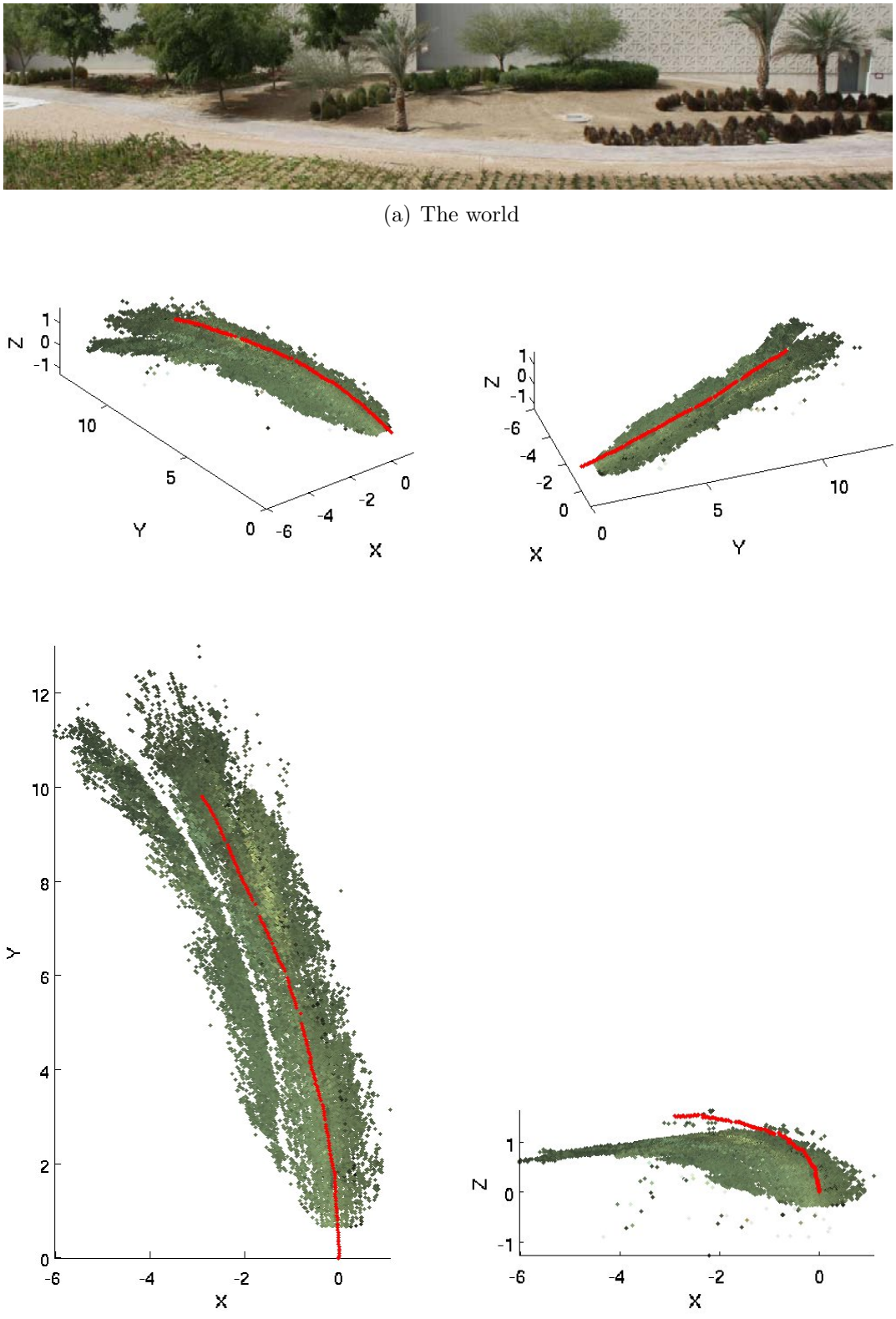


Figure 4.9: An outdoor dataset used to evaluate the algorithm



(a) The world

Figure 4.10: Visual odometry results shown on one of the outdoors datasets

5

Discussion

Visual odometry for localization and mapping is a certainly promising approach. The ability to use the algorithm indoors as well as outdoors is a great advantage. Other advantages include the ability to generate 3D maps of the environment using a camera, rather than a 2D map using other sensors. Cameras in particular have several advantages including the low-cost and high-information content. However, some issues need to be addressed in order to obtain a scalable and robust visual odometry system. In this chapter we discuss some of the major issues needed to obtain scalable and robust performance. These issues include:

- Feature detector & tracker performance
- Motion estimation from point correspondences
- Nonlinear minimization
- RANSAC model-acceptance threshold

5.1 Feature Detector & Tracker Performance

Performance of feature detector and especially the tracker is crucial to robust visual odometry. Three main issues to be considered:

1. Number of features extracted
2. Baseline length between tracked features
3. Accuracy of tracking across frames

5.1.1 Number of features

An integral part of the approach is the use of RANSAC to obtain an initial robust motion estimates. The more the number of samples used to estimate motion, the better RANSAC performance is. For example, if 30% of the extracted features are correct on average, then extracting 1000 features will yield 300 correct matches on average. However, if only 100 are extracted, then we would only expect 30 matching pairs to be correct. Although it is possible to determine camera pose uniquely from three 3D points, the least-square fitting approach will be hurt in the presence of non-Gaussian noise. Thus, it is better to always solve the overdetermined system and choose the best fit with the help of RANSAC.

Image noise is an important factor that affects features extraction and matching significantly, especially in visual odometry. The real-time requirements of visual odometry prevents the use of sophisticated feature extraction methods. Feature extracted in real-time have a much higher percentage of noise. Further, feature extraction and matching across frames is not an error-free processes. Noisy points propagate to feature matching across frames reducing the accuracy of matching, let alone the challenges of feature matching in different views. Correlation-based matching between feature vectors in different views does not explicitly make use of the recovered camera motion parameters as the scale of features change. In our approach, we use CenSurE features extracted in scale space to avoid this problem. However, scale-space feature extraction does not eliminate the problem entirely. Accepting inaccurate matches hurts the algorithm significantly. Feature extraction and tracking are the main challenges in our approach. We extract approximately 500 features per frame, 200 of those are distinctive enough to be tracked. However, the 3D-2D correspondence step with the stereo data reduces the number to less than 100 features per image. The number is further reduced to an average of 50 features after the RANSAC outlier rejection step. The problem is exaggerated with the small baseline of the stereo, which requires tilting the camera towards the ground to keep the 3D triangulation step as accurate as possible.

With that mentioned, visual odometry could benefit greatly if the camera is able to triangulate far away points, especially in outdoor environments. Farther points remain the field of view of the camera for a longer period of time, which allows tracking those feature for a longer temporal window. Another advantage is that the motion of far away points seem minimal with respect to the camera, which make their use ideal for rotation estimation. This idea of separating rotation from translation has been implemented by Tardif et al. [43]. Their approach of decoupling rotation from translation is key that allows their system to achieve a trajectory of 2.5Km in urban areas.

Problems caused by an insufficient number of inliers, or matches, is shown in Figure 5.1. In this situation, the robot turned quickly and introduced a significant blur to the image. Feature extraction and definitely matching was hurt significantly by the

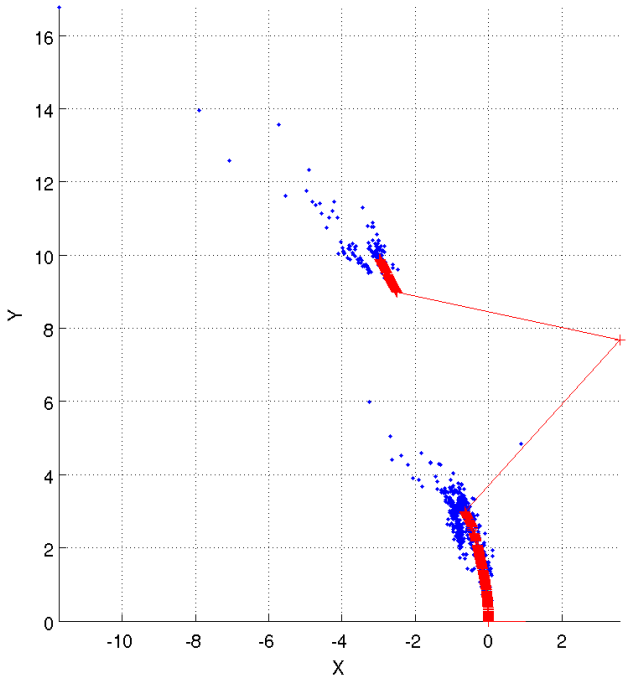


Figure 5.1: Lost pose caused by a low number of matched points, exactly three points, used to find an initial estimate of motion. The very low number of matched features is due to image blur caused by a fast turn by the robot

blurry image and caused the algorithm to break.

5.1.2 Accuracy of tracking across frames

Several solutions to the problem of feature tracking and matching have been proposed. An obvious solution is the use of more robust feature extraction methods, such as the use of SIFT feature descriptor [31], or MSER regions [44]. However, real-time requirement of visual odometry especially for mobile robots prevents the use of computationally expensive feature extraction/description methods. A simpler solution is to extract as many points as possible and hope that a large enough subset of those points could be correctly matched. The problem with this approach is that the density of features will hurt matching accuracy and might cause a large number of correct features to be incorrectly rejected.

Other solutions include the use of guided feature matching based on the epipolar constraint between multiple views [8]. However, rotations around the center of projection without translation require special treatment. Other guided matching methods include the use of the estimated motion to predict the location of matches and reject inconsistent matches early on. Guided matching based on motion models requires the assumption

that camera motion can be expectedly modeled. In other words, the camera cannot suddenly move significantly. For example, a camera cannot rotate 180° between two frames in normal circumstances. In the case of mobile robots, we can predict the robot motion based on previous measurements, especially at high frame rates and incorporating the estimated motion to guide feature matching could potentially help the performance of the algorithm.

Matching across several frames before motion estimation is a more robust solution. As mentioned earlier, features tracked for two frames only are most probably wrong or inconsistent matches. Feature extraction and tracking could be modified such that it only tracks points that are visible for more than two frames. Triplets of features could be used to estimate the initial motion, which is a more robust approach to feature tracking and motion estimation for a visual odometry system for mobile robots.

5.1.3 Baseline length between tracked features

Another important issue to keep into consideration is the baseline between the features being tracked. If the baseline is too small, then structure refinement becomes unstable. The situation arises when a mobile robot is driving slowly in environments with similar textures and cannot match feature accurately. This causes many tracks to contain two features only that are very close to each other. Given that a feature could only be tracked for 2 frames and the robot is driving slowly, then it is very likely that this feature is only noise and its use in a refinement step is not recommended. In the case of visual odometry for mobile robots, the length of a track is a good indication to the quality of features and accuracy of the approach. Longer tracks imply the detection of distinctive features and constraints the refinement of the world structure. However, track length as an indication of an accurate SFM is not necessarily true in other applications.

Finally, small robot motions tend to cause drift in the estimated trajectory. This is even more visible when the robot is stationary and is mainly attributed to incorrect feature matching due to image noise. In solution to this problem, wheel odometry could be used to detect visible motion before visual odometry. In cases where wheel odometry is not present, the use of the initial motion estimation from RANSAC could serve as a good indication of whether motion estimate for the current frame should be integrated into the visual odometry estimate. Another approach to solve this problem is measuring the baseline between feature tracks. More precisely, measuring the baseline between points that are close to the robot, which can be done without an explicit computation of the ground plane.

5.2 Motion Estimation from Point Correspondences

In this work, two motion estimation algorithms have been compared. One is absolute orientation that recovers camera motion from 3D-3D point correspondences in different frames. The other is relative orientation, which estimates camera motion from 2D-3D correspondences. Empirical experiments have shown that relative orientation produces more stable estimates than absolute orientation, without significantly affecting the running time of the algorithm.

Whether relative orientation is always better than absolute orientation is an open questions. For many cases, 3D triangulation from stereo will have several inaccuracies that grow nonlinearly with the distance of the triangulated points. However, Konologie et al. [3] report a successful visual odometry system from stereo data using absolute orientation as the initial motion estimation step. On the other hand, Nister et al. [4] reports similar problems of the use of absolute orientation in camera motion estimation in their work, although no detailed results were reported. The most likely explanation of these different results is the stereo triangulation algorithm in use. If 3D points triangulation is accurate enough, then absolute orientation would show a good performance comparable to the use of relative orientation. Defining *g*ood enough triangulation is not an easy tasks as well as obtaining good enough 3D points. Graph based stereo algorithms show the best performance in computing disparity from stereo [45]. However, the computational complexity of graph based stereo algorithm makes their use inapplicable for real-time performance. In summary, the use of 2D-3D point correspondence would be a better approach in the general case.

5.3 Nonlinear Minimization

Visual odometry is a nonlinear problem at its heart. The camera projection model, 3D triangulation from stereo and the 3-pt algorithm are all nonlinear. Similarly, the 3D triangulation step is nonlinear. The situation is more complex with the 3-pt algorithm that requires finding a solution for a 4^{th} degree polynomial, or even a 10^{th} degree polynomial[11]. Thus, there is a need for a nonlinear minimization step.

Further, the approach is also iterative. Camera motion estimation depends not only on the last frame, but only on several frames in the sequence. Accumulation of error is a serious problem that has to be addressed for a robust visual odometry.

Bundle Adjustment (BA) is the recommended algorithm to refine motion and structure. Exploiting the sparseness of the problem and the use of local BA allows for real-time performance and optimal refinement. Also, the use of BA gives a good indication of the accuracy of the initial motion estimation step using either relative orientation or absolute orientation.

Estimation obtained using relative orientation or absolute orientation is used as a starting point for nonlinear minimization to minimize the reprojection error. If this estimate is good enough, then BA will need less than 15 iterations to converge, which can be done very efficiently. However, poor initial solutions will result in a much more computationally expensive BA step. Using the number of iteration as an indication of the accuracy of the initial estimates could be used to assess the validity of a certain motion estimation. If BA is run for 20 iterations and no good solution is found, dropping the frame reduces the error risks that will hurt future frames. Although somewhat expensive, the idea of using the number of BA iterations could be used to select keyframes that are reliable for visual odometry.

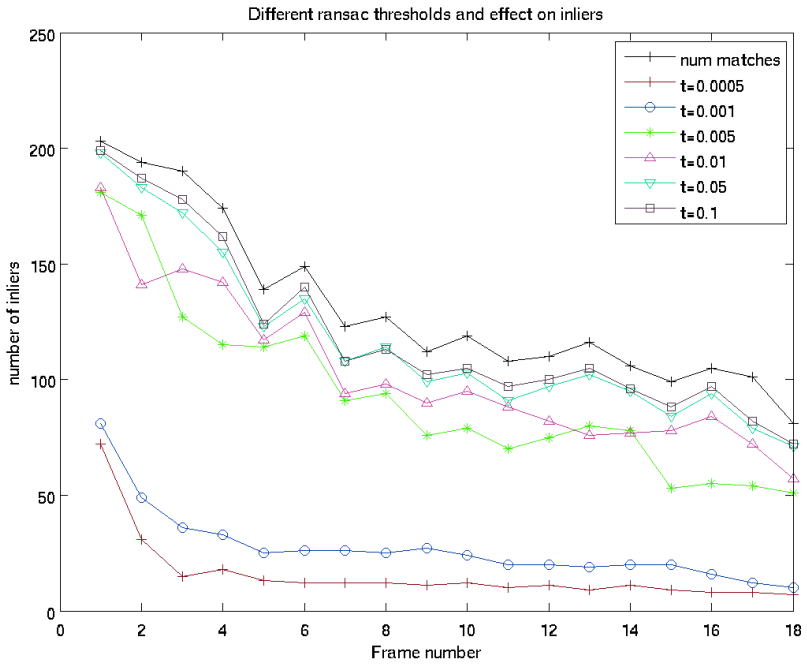


Figure 5.2: Number of RANSAC inliers with respect to threshold

5.4 RANSAC Model-acceptance Threshold

The aim of using RANSAC is to obtain an accurate subset of measurements from an initial set of noisy measurements. This is accomplished by using a randomly selected subset of data with the minimum cardinality to compute a model. After that, the model is fitted to all measurements in the set and each measurement residual error is computed. An accurate subset of measurements is obtained from the model that generates the highest number of inliers. Inliers are those measurements that have an error less than a specified threshold.

In many case, the threshold is just a magic number that is very hard to compute before hand. In the context of visual odometry, RANSAC contributions are two fold. One, it makes motion estimates robust by rejecting noisy measurements form being included in the estimation. The other, it filters inaccurate 3D points from initial set of 3D points, which will be used in the next step of the algorithm.

The next step of the algorithm is refinement using nonlinear least-squares that does not take into account error/noise models. Although it is possible to include such models, computation time of the refinement is expected to increase significantly. Hence, rejecting any outliers before the nonlinear refinement step is very important.

The number of motion estimation inliers is determined by one threshold, which is the maximum allowed reprojection error. On the one hand, a very strict threshold provides more accurate results, by accepting points that generate very low errors in the generated motion model. However, more accurate results come at the price of lowering the number of 3D points that survive to the next step of the algorithm. On the other hand, a very relaxed threshold allows many points to pass through the RANSAC robustness filter, which is good for having a more dense model of the world. However, the majority of those points are inconsistent with the motion model and hurt the algorithm significantly.

Figure 5.2 shows the relation between number of RANSAC inliers and acceptance threshold. As expected, the relationship between inliers and threshold is counter proportional. In this work, we use the value 0.005 as RANSAC threshold. This value is the maximum accepted reprojection error in normalized image coordinates.

6

Conclusions & Future Work

In this thesis, we have presented a visual odometry system based on camera input only. The system estimates a mobile robot position overtime as well as generate a 3D map of the surrounding environment. The main contributions of the thesis include

- Analysis of reasons preventing visual odometry to scale
- Empirical experiments favoring the use of relative over absolute orientation to obtain an initial estimate of motion
- Introduction of a novel track selection mechanism to select points to be included in the refinement step and the implementation of the system.
- Implementation of the system and evaluation on indoor and outdoor datasets.

One of the most important considerations in developing a visual odometry system is the performance of feature matching and tracking. Accurate matching between frames and a large enough set of features is a very important factor in a robust visual odometry system. A small number of matched feature between views will generally result in an inconsistent motion estimates that eventually break the estimation process.

Further, in this thesis, we have empirically shown that relative orientation outperforms absolute orientation. Relative orientation relies on image measurements, which are much more accurate than 3D points triangulated from imagery. Even if the triangulation process is made more accurate, relative orientation is at least as good as absolute orientation with a very little overhead in terms of computational complexity.

The many nonlinearities in the various steps of visual odometry require the use of nonlinear minimization methods. In this thesis, we use the Sparse Bundle Adjustment (SBA) for this task. SBA is probably the most accurate refinement approach, but is a computationally expensive algorithm if used on a large problem. The use of local SBA instead of global SBA simples the process and allows for faster refinement.

Selecting 3D features to be included in the refinement is also an important consideration. Short feature tracks are better excluded from the refinement as those are associated with small baseline in most of the time. Eliminating those points from the refinement step not only increases accuracy, but also reduces the running time of the algorithm as the number of 3D points in the refinement step becomes smaller.

Several possibilities for future work could be considered. To increase the robustness and scalability of visual odometry, we propose the use of robust image matching techniques to 'stitch' segments of visual odometry. The number of features extracted, or number of matches between consecutive views is a good indication for the success of the algorithm. If the number of extracted features or matches between conductive frames drops suddenly to a very low value, then it is very likely the image is blurry. Image blur is most of the time caused by fast robot rotation. Once the rotation is over, image come back to a good quality that allows extracting of a large number of features and matches. Blurry images causes by rotations are typically introduced for two or three frames. Thus, once a low number of features is detected we can stop visual odmetry and wait for a good image. Once a good image is obtained, there is a very high probability that the image will have some overlap with the last image in the previous visual odometry. Given that the overlap region might not be large enough, robust image matching techniques could be used to connect the previous segment of visual odometry with the current one using direct absolute orientation with scale estimation. The merit of the proposed approach is that it retains the benefit of fast feature extraction methods and performs the more expensive robust image matching only when needed.

Other immediate possibilities of future work include the assessment of other feature types. In this work, CenSurE features were extracted in scale-space and used for motion estimation and refinement. It will be interesting to experiment with simpler features, such as Harris corners and compare results. Further, we would like to compare the algorithm against data from (D)GPS to get a better grasp of the algorithm's performance.

Bibliography

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part II," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, 2006.

[3] K. Konolige, M. Agrawal, and J. Sol, "Large-scale visual odometry for rough terrain," SRI International Visual SLAM Workshop, 2007.

[4] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *CVPR04*, 2004, pp. I: 652–659.

[5] N. Sünderhauf, K. Konolige, S. Lacroix, P. Protzel, and T. U. Chemnitz, "Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle," in *In Tagungsband Autonome Mobile Systeme*. Springer Verlag, 2005.

[6] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," in *Proceedings of the International Symposium on Experimental Robotics*, 2006.

[7] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 865–880, 2002.

[8] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[10] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center surround extremas for realtime feature detection and matching," in *ECCV08*, 2008, pp. IV: 102–115.

[11] R. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, "Review and analysis of solutions of the three point perspective pose estimation problem," *Int. J. Comput. Vision*, vol. 13, no. 3, pp. 331–356, December 1994.

[12] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629+, April 1987.

[13] L. Chittka and J. Tautz, "The spectral input to honeybee visual odometry," *Experimental Biology*, vol. 206, pp. 2393–2397, 2003.

[14] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, 1995.

[15] T. Bonde and H. H. Nagel, "Deriving a 3-d description of a moving rigid object from monocular tv-frame sequence," in *In J.K. Aggarwal & N.I. Badler, editor, Proc. workshop on computer analysis of time-varying imagery*, Philadelphia, PA, April 1979, pp. 44–45.

[16] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Stanford, September 1980.

[17] Y. Cheng, M. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers," IEEE Robotics and Automation Magazine, 2006.

[18] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics, Special Issue on Space Robotics*, vol. 24, 2007.

[19] E. Mouragnon, F. Dekeyser, P. Sayd, M. Lhuillier, and M. Dhome, "Real time localization and 3d reconstruction," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, 2006, pp. 363–370.

[20] R. Eustice, "Large-area visually augmented navigation for autonomous underwater vehicles," Ph.D. dissertation, Massachusetts Institute of Technology / Woods Hole Oceanographic Joint-Program, June 2005.

[21] P. Corke, D. Strelow, and S. Singh, "Omnidirectional visual odometry for a planetary rover," in *In Proceedings of IROS 2004*, 2004, pp. 4007–4012.

[22] R. Eustice, "Large-Area Visually Augmented Navigation for Autonomous Underwater Vehicles," Ph.D. dissertation, MIT - Woods Hole Oceanographic Institute, June 2005.

[23] O. A. T. Kanade and J. R. Miller, "Autonomous helicopter research at carnegie mellon robotics institute," in *Proceedings of Heli Japan '98*, April 1998.

[24] D. Nister, "Preemptive ransac for live structure and motion estimation," in *ICCV03*, 2003, pp. 199–206.

[25] ——, "An efficient solution to the five-point relative pose problem," in *CVPR03*, 2003, pp. II: 195–202.

[26] N. Süderhauf, K. Konolige, S. Lacroix, and P. Protzel, "Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle." in *AMS*, ser. Informatik Aktuell, P. Levi, M. Schanz, R. Lafrenz, and V. Avrutin, Eds. Springer, 2005, pp. 157–163.

[27] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[28] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, 2005, pp. 1508–1515 Vol. 2.

[29] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, October 2004.

[30] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision - ECCV 2006*. Springer, 2006, pp. 404–417.

[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[32] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI81*, 1981, pp. 674–679.

[33] O. Faugeras and M. Hebert, "A 3-d recognition and positing algorithm using geometrical matching between primitive surfaces," in *IJCAI83*, 1983, pp. 996–1002.

[34] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, pp. 698–700, September 1987.

[35] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[36] J. A. Grunert, "Das pothenotische problem in erweiterter gestalt nebst Über seine anwendungen in der geodäsie," *Grunerts Archiv für Mathematick and Physik*, vol. Band, no. 1, pp. 238–248, 1841.

[37] F.Moreno-Noguer, V.Lepetit, and P.Fua, "Accurate non-iterative o(n) solution to the pnp problem," in *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

[38] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment – a modern synthesis," *Lecture Notes in Computer Science*, vol. 1883, pp. 298+, January 2000.

[39] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Journal of Applied Mathmatics*, vol. II, no. 2, pp. 164–168, 1944.

[40] D. W. Marquardt, "An algorithm for least-squares estimation of non-linear parameters," *Journal of the Society of Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[41] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, pp. 79–116, 1998.

[42] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Tech. Rep. 340, Aug. 2004, available from `http://www.ics.forth.gr/~lourakis/sba`.

[43] J.-P. Tardif, Y. Pavilidis, and K. Danniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *IROS*, 2008.

[44] E. Murphy-Chutorian and M. Trivedi, "Maximally-stable extremal regions," in *British Machine Vision Conference*, 2002, pp. 384–396.

[45] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *IEEE Workshop on Stereo and Multi-Baseline Vision, 2001. (SMBV 2001)*, 2001, pp. 131–140.