

AMERICAN UNIVERSITY OF BEIRUT

NON-ITERATIVE VISUAL ODOMETRY USING A
MONOCULAR CAMERA

by
FIRAS AKRAM ABI FARRAJ

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Mechanical Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
March 2014

AMERICAN UNIVERSITY OF BEIRUT

NON-ITERATIVE VISUAL ODOMETRY USING A
MONOCULAR CAMERA

by
FIRAS AKRAM ABI FARRAJ

Approved by:



Dr. Daniel Asmar, Assistant Professor
Mechanical Engineering

Advisor



Dr. Elie Shammass, Assistant Professor
Mechanical Engineering

Member of Committee



Dr. Imad Elhajj, Associate Professor
Electrical and Computer Engineering

Member of Committee

Date of thesis defense: March 28 2014

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Abi Farraj, Firas Akram


Master's Thesis

Master's Project

Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.



Signature Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

ACKNOWLEDGEMENTS

I am here to say a few words of gratitude for some people who really deserve such words.

I would like to first thank my advisor Dr. Daniel Asmar for the great support and coaching throughout the time I spent here at AUB. Looking back, I am thankful that I tumbled upon his office and got the chance to work with such a committed, humble, and trustworthy person.

I would like to thank my friends at the lab and outside the lab for all the support they gave me. I would like to also thank some special people with whom I lived my best moments throughout the past three years. Life separated me from some of them while others are still near. I would like to say a word of gratitude even though most of them won't read this.

I would like to also thank my family for always being there. My special brothers and my great parents. They are the main reason for me being the person I am now and I am thankful for all the quarter century I spent with them :)

I am here to say that I am grateful for every person with whom I had any form of human interaction whether that was a simple smile or a lifetime of lengthy complex discussions on life and existence. I am here to say that I am grateful for the everlasting system governing this universe and for everything it has given me. And it has given me much!

At last, and if I am to have a simple word for everyone who may take the time to read this one day that would be: Never postpone your life for tomorrow. It is not that I have to do my masters or phd now and I will start living later. If you are not living today you will not live tomorrow. You will also regret all the moments you lost.

Thank you

AN ABSTRACT OF THE THESIS OF

Firas A. Abi Farraj for Master of Engineering
Major: Mechanical Engineering

Title: Non-Iterative Visual Odometry using a Monocular Camera

This thesis presents a visual odometry system for ground vehicles using a single downward-facing camera and two tilt sensors. Conventional visual odometry algorithms use the probabilistic and iterative RANSAC to remove outliers and calculate the motion. They suffer from different problems including dynamic obstacles, changes in lighting conditions, inaccuracy in depth estimation and the high computational cost. The proposed method calculates the motion from a monocular camera without using any probabilistic (non-deterministic) or iterative routines. It makes use of the constant distance between the camera and the ground to impose the depth and improve the accuracy. Moreover, it makes use of the concept of a downward looking camera and the known depth to implement the inliers-detection method as a substitute for RANSAC to remove outliers. This improves the speed of the algorithm and decreases the computational cost. The algorithm is validated for real data sets and shows competitive accuracy and robustness with a loop closure error reaching as low as 1.25% for a run of 461 meters.

CONTENTS

ACKNOWLEDGEMENTS	v
ABSTRACT	vi
LIST OF FIGURES	viii
LIST OF TABLES	x
1 INTRODUCTION	1
1.1 Thesis Outline	3
2 BACKGROUND	5
2.1 Projective Geometry	5
2.1.1 Camera Intrinsic Parameters	7
2.2 Multiple View Geometry	7
2.2.1 Features Extraction and Matching	8
2.2.2 Two View Geometry Computation	9
3 VISUAL ODOMETRY STATE OF THE ART	13
3.1 Thesis Contribution	16
4 VISUAL ODOMETRY SYSTEM	18
4.1 Inliers Detection	22
4.2 3D-to-3D Motion Estimation	24
4.3 Imposing the 3D Structure	25
5 CALIBRATION	27
5.1 Experimental Setup	27
5.2 Calibrating the Rotation Matrix Between the Tilted Camera and the Vertical	30
5.3 Scale Estimation	32
5.4 Coupling the tilt sensors	33
6 RESULTS	35
6.1 Results for the runs on a 2D plane	35
6.2 Results with the tilt sensors	38
7 CONCLUSIONS	44
8 APPENDIX	45
REFERENCES	57

LIST OF FIGURES

2.1	Projecting a 3D point on the projection plane	6
2.2	Projecting a 3D point on the projection plane (top and side view) . .	6
2.3	the pixels in an image from a digital camera	8
4.1	Left: the downward-looking slightly tilted camera. Right: After rotating to the virtual downward-looking camera	19
4.2	Virtual downward-looking camera with the features	21
5.1	A picture of the experimental setup showing the camera and the robot	27
5.2	Effect of the pitch angle on our proposed algorithm.	29
5.3	Top: 10 degree polynomial fitted to the tilt sensor noisy data. Bottom: 10 degree polynomial fitted to the tilt sensor noisy data after dividing it into several intervals.	34
6.1	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 461meters.	37
6.2	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 90.4 meters.	38
6.3	Above: Estimated trajectory on a non-planar path using my algorithm. Below: Estimated path using fused data from a GPS and an Inertial Measurement Unit (MTi-G IMU)	40
6.4	This figure shows the time consumed by each of the inliers detection method and RANSAC (with 160 iterations) versus the number of processed frames. An improvement by 58% is observed.	43
8.1	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 13.66 meters and an error of 0.43%.	45
8.2	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 12.53 meters and an error of 0.35%.	46
8.3	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 12.5 meters and an error of 1.67%.	47
8.4	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 20.4 meters. The error is 0.78%.	48
8.5	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 15.8 meters. The error is 0.78%	49

8.6	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 14.7 meters. The error is 1.6%	50
8.7	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 26.4 meters. The error is 0.91%	51
8.8	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 26.6 meters. The error is 0.46%	52
8.9	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 100.2 meters. The error is 1.48%	53
8.10	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 90.4 meters. the error is 0.56%	54
8.11	Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 418.1 meters. The error is 1.25%	55
8.12	Above: Estimated trajectory on a non-planar path using my algorithm. Below: Estimated path using fused data from a GPS and an Inertial Measurement Unit (MTi-G IMU)	56

LIST OF TABLES

6.1	The results from the different runs showing the error and the distance covered in each	36
-----	--	----

CHAPTER 1

INTRODUCTION

Navigation is the process of planning, recording, and controlling the course and position of a body. Automating this process is a key aspect of the evolution of transportation systems and is still one of the major challenges facing full autonomy. The ego motion estimation of a body is the initial step in navigation and it refers to estimating the path a body has moved and the final position and orientation it has reached.

The most popular technology for navigation is the global positioning system (GPS) although one of its drawbacks is outages that occur in urban canyons and under foliage. Dead-reckoning systems, on the other hand, are based on incrementally estimating position by integrating the differential motion estimates in time. Each position is dependent on previous measurements and the track is estimated by calculating the displacement between each two positions x_{i-1} and x_i and summing them all together. An error in calculating one position estimate propagates throughout the sequence incrementing the error with each step.

Local positioning systems differ widely and can be divided into different categories. The most popular sensors used in local positioning systems are encoders and Inertial navigation systems (INS). Encoders are cheap and easy to use but are relatively inaccurate because of unequal wheel diameters, misalignment of wheels, uneven travel floors, wheel slippage and others [6]. On the other hand, Inertial Navigation Systems can be accurate but this accuracy comes at a price. Conventional IMUs drift quickly which highly impacts their accuracy [38].

Range finding sensors, like lasers and LIDARs can also be used for local

motion estimation. LIDARs are used to form a 3D map of the environment in which the robot is moving. This 3D map is then used to navigate using the Iterative Closest Point method (ICP) or Simultaneous Localization and Mapping (SLAM).

One of the latest techniques developed for localization and tracking are those based on computer vision which have become more and more influential with the increasing computational power of modern computers. Real-time visual odometry algorithms include one of the variants of optical flow, Structure From Motion (SfM) [39] or Visual SLAM [8, 27, 3, 40, 5] and others [20, 10, 14, 31, 25]. SLAM/SfM can produce good results but the computational overhead is generally overbearing for real-time applications, unless additional complexities are introduced to the standard algorithms. For instance, in the filtering techniques of SLAM, computation is reduced by marginalizing out all poses except for the current one [9]. In SfM the practice has been to limit Bundle Adjustment to certain keyframes of the image sequence [22].

When one is only interested in recovering the motion of a vehicle without particular focus on the structure of the environment, a special case of SfM known as Visual Odometry (VO)[36, 12] is applied. While most flavors of VO also involve constructing a local map, one only cares about local consistency of the trajectory; whereas in SLAM global motion and map consistency is sought usually after loop closure. Further constraints can be applied when the vehicle is moving in a structured environment among which one of the most common is the assumption that the ground is locally planar [21, 31, 25].

Visual odometry algorithms are highly dependent on the working environment in which they are used [12]. They need specific conditions to give accurate results and can be totally unreliable if these conditions are not satisfied. One of the important conditions for SfM/SLAM to give accurate results is the

presence of a rich 3D structure in the camera's field of view [17]. Having a dominant plane in the field of view of the camera is a mathematical singularity which leads to unacceptable results. Moreover, an accurate calculation of the translation needs a high number of close features whereas rotation is more dependent on far ones. The absence of close features leads to a bad estimate of the translation which has a bad impact on the calculated path.

In vast areas like a desert, a playground or a highway the above conditions are not satisfied. Most of the features in the camera's field of view are far and not reliable for an accurate motion estimation. Furthermore, the remaining close features belong to a dominant plane (the ground plane) and as previously mentioned, the presence of a dominant plane is a mathematical singularity for both algorithms. Adding to that the high computational cost resulting from the iterative nature of all algorithms used on a monocular camera, the need for a better solution for monocular visual odometry arises.

I present in my work a visual odometry system for ground vehicles using a single downward-facing camera and two tilt sensors. The method calculates the motion from a monocular camera without using any probabilistic (non-deterministic) or iterative routines. Accordingly, the proposed method improves the accuracy, the certainty, and the speed of the algorithm. It is validated for real data sets and shows competitive accuracy and robustness with a loop closure error reaching as low as 1.25% for a run of 461 meters.

1.1 Thesis Outline

This thesis proceeds as follows:

Chapter 2 gives a background for this thesis. An overview of basic models and theories in computer vision ranging from the feature extraction and matching

to the motion estimation itself.

Chapter 3 presents a literature review for visual odometry. Previous studies tackling visual odometry are put forward and the main contribution of this thesis is described.

Chapter 4 explains the theory of the presented algorithm and the details of each step.

Chapter 5 presents the experiments and results we obtained by applying the algorithm on real data.

Chapter 6 concludes the thesis with a discussion and suggestions for future development.

CHAPTER 2

BACKGROUND

Visual Odometry is the process by which we calculate the motion of an agent (vehicle, robot, etc) using the input from one or multiple cameras. Images taken are studied and analyzed to get the needed data for calculating the agent's position. In this section the basic theoretical background needed for understanding visual odometry is presented.

2.1 Projective Geometry

An image is the projection of a 3D scene on a plane. As can be seen from Fig. (2.1), the light ray emitted from a 3D point P and passing through the center of projection (COP), also called the focal point, intersects the projection plane situated at a distance $d = focallength$ from the COP at a certain point p. Now moving from a single 3D point to a general 3D scene, all light rays reflected by the scene and passing through the COP form the image at the projection plane (image plane).

The frame of reference is taken to have the COP (focal point) as its origin and the axis perpendicular to the projection plane as its z-axis. This is called the camera frame. It is related to the world frame of reference by a 3D transformation consisting of a translation t and a rotation R —called the extrinsic camera parameters. A 3D point with coordinates $(X, Y, Z)^t$ with respect to the camera frame is projected in the image plane to the 2D point $(d * Y/Z, d * X/Z)$ as can be seen in Fig. (2.2). Rearranging and replacing d by its value f (the focal length of the camera) we get:

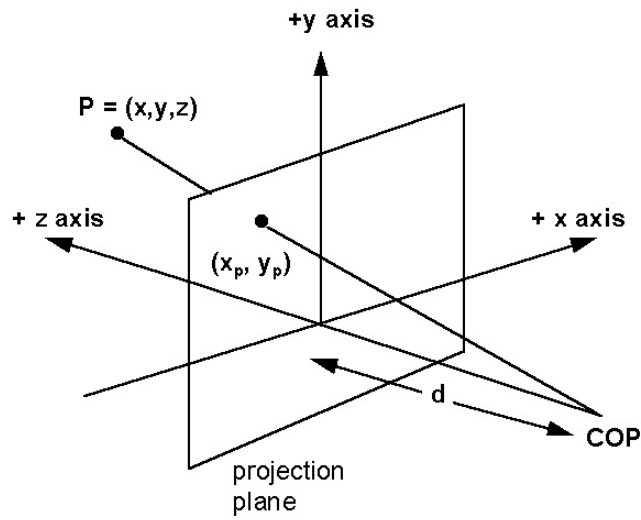


Figure 2.1: Projecting a 3D point on the projection plane

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 1/Z \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad (2.1)$$

where (X, Y, Z) are the coordinates of the 3D points and (x, y) are the coordinates of the 2D projections.

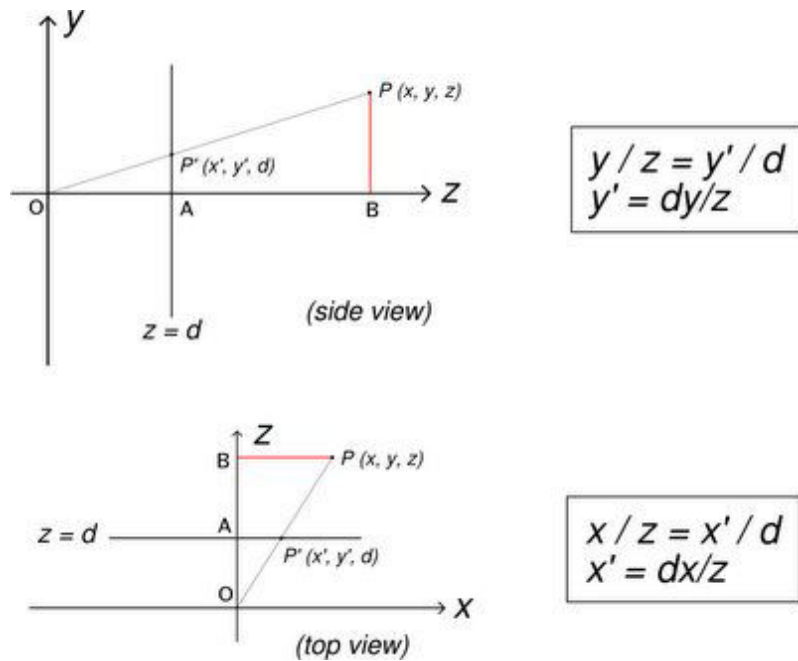


Figure 2.2: Projecting a 3D point on the projection plane (top and side view)

2.1.1 Camera Intrinsic Parameters

Fig. 2.3 shows a simplified example of how an image is perceived by a computer. Here, the coordinates of a point are given in pixels with respect to the upper left corner of the image and are called pixel coordinates. When these coordinates are transformed to absolute coordinates expressed in mm with respect to the center of the image, they are called normalized coordinates. Rearranging equations from Fig. (2.3) we get:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & O_x \\ 0 & s_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (2.2)$$

where (x', y') are the pixel coordinates and (x, y) are the normalized coordinates.

If the pixels are not rectangular but distorted by an angle θ , the above relation becomes:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & O_x \\ 0 & s_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \quad (2.3)$$

2.2 Multiple View Geometry

Two images of the same scene have multiple common features between them. These two images can be matched together extracting the common features between them after which the extracted matches can be used to calculate the displacement between them.

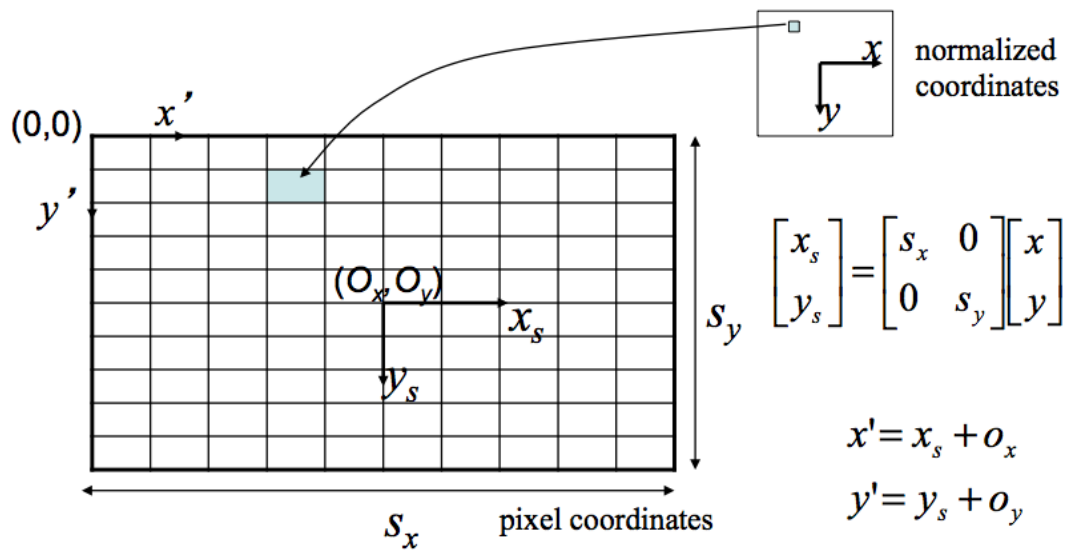


Figure 2.3: the pixels in an image from a digital camera

2.2.1 Features Extraction and Matching

Extracting the unique features from each image and matching them together defines the first most essential step to proceed with visual odometry. These features are usually located where there is a high contrast in the image like objects' edges and corners. Researchers presented many types of features including SIFT [26], SURF[4], FAST[34], ASIFT[42], HARRIS[16] corners detector, and others. Extracted features are then matched between each two frames using a matching technique that is dependent on the type of feature-extraction algorithm involved. Much work has been done on these algorithms and they have reached a notable accuracy in the past few years although mismatches are still inevitable. Some typical cases in which matching fails are the following:

- Different lighting conditions:

Rapid changes in lighting conditions, like when a car flashes its lights at you for example, lead to a big change in the colors of the pixels and the intensity of light in each which highly disrupts the implemented matching algorithm.

- Spectral surfaces (like the surface of a mirror)

The positions of the features reflected on a spectral surface change as you move. This change in position makes them useless in estimating the motion of the camera. However, the matching algorithm can't differentiate them from real fixed features and they must be treated off-line.

- Periodic structures where the same pattern is repeated

Periodic structures obviously fool the matching algorithm. Most of the remarkable features in a periodic structure are repeated several times and the matching algorithm can't differentiate one from another.

- Strong 3D structure where a change in the view angle changes the object's aspect completely.

This case is mostly specific to some sculptures and some forms of modern art architecture where the 3D shape can be very complex in contrast to the simple conventional cases.

If matching between two frames fails, the calculated transformation (R and t) between these frames will be drastically affected. Wrong matches are often called outliers and are treated in many ways among which the most famous is RANSAC [28, 11]. However, even with these algorithms, the accuracy of the system is still much effected when the amount of outliers and noise is notable.

2.2.2 Two View Geometry Computation

Our aim here is to find the mathematical relation between features of the two images. This relation is a function of the displacement between the two frames (*i.e.* , the rotation and translation between the two camera poses). It depends on the structure of the scene that is captured by the camera. In the case of a general 3D scene, the fundamental matrix governs the relationship between the two frames while for a planar scene, a simpler homography may do. Other techniques for

calculating the displacement between two frames include optical flow and correlation or template matching.

SFM (structure from motion) algorithms are among the most popular visual odometry algorithms. They start by calculating the Fundamental matrix (or Essential matrix in the case of a calibrated camera) from the matches between the two images using several algorithms among which the most known are the 8 points and the 5 points algorithms [17, 29]. The rotation and the translation between the two frames are then extracted from those matrices after which the 3D environment is reconstructed using triangulation and a resulting map is created. The map can be a dense map or simply a sparse one in cases like our case where we are just interested in tracking the robot. SFM is used either with a single camera where a scaled map is obtained or with a stereo rig where the map is to the correct scale.

Another variant in which a map is formed and the robot is localized in that map is SLAM. The dependency between localization and mapping is what makes SLAM and SFM problems difficult to resolve. SLAM uses a Gaussian or a particle filter on previously gained information to predict the location of a robot with a certain probability. It then finds the best estimate using measurements and observations like features or depth values from a camera or a laser. While in SFM we may concentrate on the local consistency of the trajectory, SLAM seeks the global motion and map consistency - usually after loop closure.

SLAM and SFM can produce good results but the computational overhead is generally over-bearing for real-time applications, unless additional simplifications are introduced to the standards algorithms and thus reducing the accuracy. Optical flow is an alternate algorithm where features are tracked through frames over time and their motion is transformed into velocity vectors which are then used to calculate the rotation and translation between the frames.

SFM/SLAM are the most important algorithms used in the case of a general 3D scene. They are sensitive and applicable only in certain environments where we have a rich 3D structure and abundance of features. Moreover, as the settings differ from an environment to another, the algorithms must be tweaked and tuned to fit the new settings by relieving some constraints and applying others. For example, in certain circumstances we may need to isolate close features from far ones and calculate the translation from the former and the rotation from the latter. Moreover, we may totally depend on 3D to 2D motion estimation after we have an initial rich 3D map or we may still need to incorporate 2D to 2D in case we have a low number of features. Such tweaks are highly dependent on the nature of the environment we are working in and are mostly the result of experimentation.

However, in some particular environments like when using a camera pointing to the ground, the scene is planar and some constraints can be added to develop new algorithms more accurate and robust than the former two. In fact any two views of the same plane are related by what is known as a homography which is a projective transformation mapping two planes in space. Using matches between two different views of the same plane, we can calculate the homography governing the relationship between the two views and use it to extract the displacement between the two camera poses. Many applications use homographies even if we have more than one plane in the camera's field of view. In such cases, a different homography is calculated for each plane and included in the motion estimation. Sometimes homographies are also coupled with a laser range finder to improve the results as will be seen in the next section.

The last method to mention is using correlation or template matching [31] which differs from all of the above. This algorithm works only on a pure planar scene. Instead of extracting and matching features between two frames, a template (a section of the image) is taken from the first image and we search for a window

in the second image to match the saved template and subsequently, the translation and rotation between the two frames are calculated. Fourier-Mellin [14] or Fast Fourier transforms [31] are coupled with optimization techniques to estimate motion in such algorithms. Some good results have been retrieved using those methods but they are also dependent on some particular conditions. Detailed literature review on homography-based and correlation-based methods will follow in the next chapter.

CHAPTER 3

VISUAL ODOMETRY STATE OF THE ART

Recovering the motion of an agent using a monocular camera is a subject that dates back to the 1980s with [24] and [15]. In 2004, Hartley and Zisserman summed up the state of the art in their famous book [17]. Later that year Nister presented his landmark paper on the five point algorithm [29]. The algorithm is an algebraic solution for the five-point relative pose problem, which imposes the constraints resulting from the camera parameters and should supposedly give more accurate results. Although five-point-based algorithms achieved relatively good accuracies on long distances [30, 37], they are still sensitive to noise and outliers and dependent on the environment in which they are implemented (urban, rural, indoor, outdoor, ...).

When the vehicle being considered is restricted to travel on the ground, and the environment is locally planar, further constraints can be applied to make VO more robust. Ke and Kanade [21] present a system based on the local ground planarity assumption, where the ground plane is first segmented and the motion is iteratively calculated after transforming the camera geometry to that of a ‘virtual’ downward-looking camera. The advantage of using this transformation is the removal of the ambiguity between translational and rotational motion parameters. In their work, the authors concentrated more on extracting the ground plane and few experimental data on the egomotion estimation were presented. Fig. ?? shows the geometry of the forward tilted camera transformed by Ke and Kanade to a ‘virtual’ downward-looking camera and used in the calculations.

Fernandez and Price were among the first to propose a camera facing the ground [10]. They used a pseudo optical flow (a simplified optical flow algorithm)

to calculate the robot's egomotion. The method proves to work on different terrains including gravel, tarmac and grass but problems were faced with static and dynamic shadows and variations in lighting conditions. The importance of this work is that it was the first to present a visual odometry algorithm with a downward-looking camera. This approach was later adopted by many researchers who presented more efficient and accurate algorithms.

In 2005, Wang et al. [41] present a visual odometry algorithm based on homographies. They restrict the motion of the robot to a flat plane and use a feature tracker, which rejects outliers in an iterative manner using RANSAC [28]. Moreover, they use a Kalman filter to predict the motion of a camera and thus the motion of the pixels on the image to help guide the feature extraction and matching algorithm and increase its speed. The error is roughly three percent on a run of twenty three meters and at a processing speed of twenty-five frames per second. The drawback of the technique is that it is iterative and the features must be tracked through frames; a condition that limits the speed of the robot in order to enable features to be tracked between three or more frames.

Template matching methods using Fast Fourier Transforms [31], Fourier-Mellin Transforms [14, 20] and other nonlinear iterative techniques [25, 41] were also used on a camera facing the ground. One of the best results reported by such methods are those presented by Kazik and Goktogan [20] using a Fourier-Mellin transform with a downward-looking camera. They achieved an accuracy of less than one percent on a straight path. However, the tests were carried out on relatively small distances of one to three meters, which raises questions as to their performance on longer runs.

One of the recent template-matching-based works is that of Nourani-Vatani and Borges in 2011 [31] who present a visual odometry algorithm for ground vehicles using Fast Fourier transforms. Conventional template matching

is the process by which we extract a template (window) from the first image and search for it in the second one. However, in the mentioned work, the authors introduce a quality measurement technique to assess the quality of the extracted template. Several templates are extracted from each image and the highest-quality template is used for the motion estimation. Experiments were performed on a car over long distances totaling 6km. The average accuracy was found to be 8%. Although the algorithm proved to be robust over long distances and on surfaces with scarce features, an accuracy of 8% is not sufficient for many real-world implementations, which need more precision.

Lovegrove et al. [25] used the rear parking camera of a car for estimating its motion. The rear parking camera captures a planar surface (the ground behind the car) at a certain angle. This planar scene is used to calculate the homography between two successive frames using nonlinear optimization techniques. No numerical results are presented in the publication but the path calculated was overlapped on a Google Earth map showing an error of no less than 7-10% if we are to compare the final offset to the covered distance.

We will now digress a little to set the tone for the remainder of the thesis, where we are promoting non-iterative motion estimation techniques. As an example of iterative motion estimation techniques let us recall Nister's 5-point algorithm [29] where a minimal set of five points are used to calculate the Essential matrix, from which Rotation and Translation are extracted. Outliers are rejected with the help of iterative RANSAC using approximately 162 iterations for each pair of images. With this number of iterations, one can guarantee that with a probability of ninety-nine percent, the calculated Essential matrix and set of inliers are correct. While this percentage is relatively high, one out of each one hundred image pairs could be wrong and if undetected, will propagate throughout the sequence, thereby corrupting the remaining motion estimate.

Howard [19] proposes to solve this problem by using the inliers-detection method, initially proposed by Hirschmuller et al. [18]. Here motion is estimated by minimizing the image re-projection error using the standard Levenberg-Marquardt least-squares algorithm. Howard reports an error of 0.25% while testing his system on a wheeled robot for a run of approximately four hundred meters. In his implementation, Howard uses a stereo camera tilted slightly downwards, where a large number of relatively close features is present; a condition which favors excellent depth estimates by the stereo rig. Another advantage of the inliers-detection method is the inherent reduction in implementation time due to the non-iterative nature of the technique. The disadvantage of using a stereo camera becomes significant when there are not enough features close to the camera and the system has to rely on far away features, where the problem of uncertainty in the depth estimation is high. Moreover, stereo cameras are not readily available everywhere, unless you create your own rig, which can easily become uncalibrated as a result of motion and vibrations.

3.1 Thesis Contribution

This thesis combines the most important notions of the above techniques, especially that of the inliers-detection method and the virtual downward-looking camera, addressing their shortcomings and presenting an ego motion estimation technique using a single camera. The method we present also imposes the local planarity of the ground as a constraint on the motion. Moreover, it makes use of the fixed position of the camera with respect to the ground to transform camera geometry to a virtual downward-looking camera, which allows us to calculate the 3D coordinates of the features (the depth being known after transforming the camera into a downward-looking one) and remove the ambiguity between the different motion parameters. After calculating the 3D coordinates, outliers are

removed using the inlier detection method and the planar motion is calculated.

Moreover, tilt sensors are coupled with the camera to calculate the full 5 dimensional motion of the wheeled robot. The main contributions of this thesis are:

- It makes use of the rigid position of the camera with respect to the ground to eliminate the uncertainty in the depth and thus improve the accuracy of the system.
- It implements the inliers-detection method on a monocular camera and avoids the probabilistic iterative RANSAC and thus presenting the first purely non-iterative method on a monocular camera, thereby decreasing the computational cost and achieving higher accuracy.

CHAPTER 4

VISUAL ODOMETRY SYSTEM

In VO the camera can be mounted to either face forwards, downwards, sideways, or a combination of the above with each having its advantages and disadvantages according to the specific VO algorithm that is used.

In the case of a forward looking camera (Z -axis pointing forward) the XZ plane is parallel to the ground, and motion includes rotation w_y about the y -axis and translation (T_x, T_z) . In this setup, there is an ambiguity between a translation T_x of the camera or a rotation w_y as they both induce the same displacement in the observed features. The same ambiguity exists for a sideways-looking camera between T_z and w_y . However, for a downward-looking camera, the planar motion resulting from w_z is different from that of (T_x, T_y) and the displacement of the features resulting from each of these parameters is independent from that of the other. For this reason, we mount the cameras facing the ground in our proposed motion estimation system.

Furthermore, the downward-looking camera is at a known constant distance from the ground, which means that the depth of the features that are captured by the camera with respect to the camera frame is known. This eliminates the uncertainty in the depth, which is a major source of error in algorithms on visual odometry. It also allows to use an inliers-detection method instead of RANSAC to decrease the computational cost, and decrease problems related to dynamic scenes. This method is described in detail later in Section 4.1.

To cope with the limited field of view of a downward-looking camera, it is tilted slightly upwards (see Fig. 4.1). Then, we back-rotate the camera view into

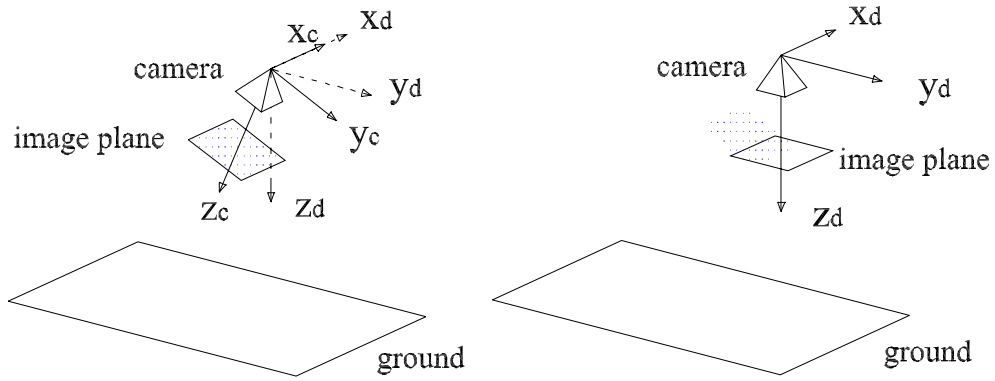


Figure 4.1: Left: the downward-looking slightly tilted camera. Right: After rotating to the virtual downward-looking camera

that of a virtual downward-looking one. This way, we will be able to impose the constraints of the downward-facing camera and estimate the motion accordingly. The angle at which the camera is tilted should be less than 30 degrees because as it increases the capability of the camera to capture accurate and detailed features from the ground plane decreases—thereby affecting the matching process.

Our approach is based on three initial assumptions, which are reasonable for wheeled robots:

1. The camera is calibrated
2. The camera is rigidly fixed to the mobile robot and the rotation matrix between it and a virtual downward-looking camera is calculated
3. The robot is moving on a locally planar ground (not a rough off-road terrain)

The input to the algorithm is a set of images extracted from a monocular camera facing the ground in addition to the data gathered from two tilt sensors. The vision algorithm is used to calculate the planar motion while the tilt sensors are used to impose the displacements in the pitch and the roll angles. The algorithm is as follows:

Algorithm 1 Motion estimation using inliers-detection

$i = 1$; **While** !end

- extract features from images i and $i + 1$
- match the images
- transform camera geometry to a downward-looking camera, impose the depth, and calculate the 3D coordinates of the features
- find the maximum set of inliers
- Compute the planar motion between the two frames
- impose the variation of the pitch and the roll angles using the data from the tilt sensors
- increment i

end while

The details of the different steps of our proposed system are presented next. Affine-SIFT (ASIFT) [42] features, X_A and X_B , are first extracted from two images, im_A and im_B , respectively. X_A and X_B are matched using their corresponding descriptors. Next, using the camera's calibration matrix K , the normalized coordinates Xn_a and Xn_b of the features are calculated. Xn_a and Xn_b define the 2D coordinates of the features in the image plane, which is located at a distance of 1 mm from the camera's focal point. The 3D coordinates X_i of those features with respect to the camera frame having the camera's focal point as its origin will then be $X_i = (x_i, y_i, 1)$.

Rotating back to the virtual downward-looking camera as can be seen in Fig. 4.1, we obtain the 3D coordinates of the features with respect to the frame of reference that is attached to the camera. This transformation is performed as follows

$$Xd_i = R_{c/d} * X_i, \quad (4.1)$$

with Xd_i being the coordinates of the feature with respect to the reference attached to the virtual downward-looking camera, $R_{c/d}$ is the rotation matrix between the real camera's reference and the virtual downward-looking camera's reference, and X_i is the 3D coordinates of the features with respect to the camera's reference.

Knowing that the features belong to the ground plane which is located at a distance d from the camera's focal point (Fig. 4.2), we can calculate the real world coordinates of the features where $\frac{x_w}{d} = \frac{x_{v_i}}{z_{v_i}}$, and $\frac{y_w}{d} = \frac{y_{v_i}}{z_{v_i}}$.

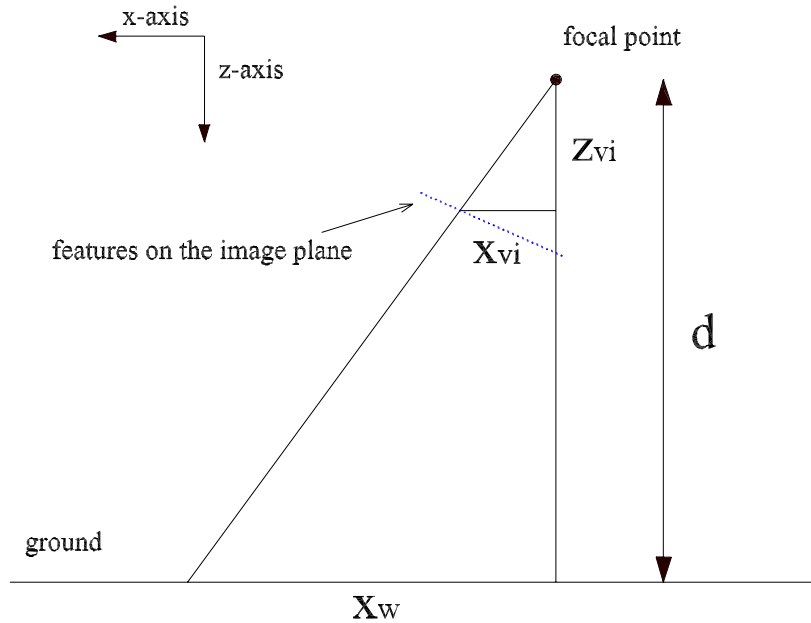


Figure 4.2: Virtual downward-looking camera with the features

where x_w and y_w are the coordinates of the features in the world coordinates, x_{v_i} , y_{v_i} and z_{v_i} are the coordinates of the features in the downward-looking camera frame and d is the distance between the camera's focal point and the ground.

This results in the determination of the 3D coordinates of the features with respect to the frame attached to the virtual downward-looking camera. The process is repeated on both im_A and im_B .

4.1 Inliers Detection

False matches are inevitable in the features matching stage regardless of the matching algorithm that is used. The most common approach to reject outliers is to use RANSAC.

RANSAC is the process by which we select at random a sample consisting of the minimal number of points needed to calculate a model—three points in our case. This model is calculated and tested on the total set of matches. The number of matches that fit to the model is then calculated and the process is repeated several times. Once we are done, the model with the highest vote (the one with the highest number of fitting matches) is considered to be the best solution and the matches that do not fit to this model are considered outliers. As the number of RANSAC iterations increases, the probability that the correct solution is reached increases. In practical scenarios, thousands of RANSAC iterations are used but this number may decrease to hundreds in real time implementations trading efficiency for speed.

Though RANSAC has many advantages it still has its disadvantages. It is a non-deterministic algorithm meaning that the correct model is selected up to a certain probability. In the case of real time implementation this probability is taken to be ninety-nine percent which seems high but actually means that for every hundred image pairs, there is one wrong pair. Reflecting on real scenarios, which have thousands of frames processed for distances of hundreds of meters, tens of wrong frames would probably exist which, if undetected, will propagate throughout the sequence, thereby corrupting the whole motion estimate. In addition, the high computational cost due to the large number of iterations promotes the use of a different paradigm.

Based on the above, and in order to achieve better robustness and

substantially reduce the computational cost, we use an algorithm inspired from Hirschmuller et al. [18] to compute the maximum set of inliers. The algorithm relies on the fact that the distance between two 3D points belonging to the same rigid body will always remain constant. This means that the distance between the 3D features X_{i_A} and X_{j_A} from frame img_A should be equal to the distance between their corresponding matches X_{i_B} and X_{j_B} from frame img_B .

Having calculated the 3D coordinates of the features in previous steps, we now need to find the set of maximum consistent inliers. We take a feature X_{i_A} from frame img_A and its correspondence X_{i_B} from frame img_B and compare the distances between X_{i_A} and all the other features X_{k_A} belonging to frame img_A to the distances between X_{i_B} in img_B and all the other features X_{k_B} belonging to frame img_B . If features i and k are inliers, then we should have

$$X_{i_A}X_{k_A} - X_{i_B}X_{k_B} = 0. \quad (4.2)$$

However, this equation is never zero because of noise. To overcome that, the average distance between each two features is compared to a threshold δ . Thus we have

$$\frac{X_{i_A}X_{k_A} - X_{i_B}X_{k_B}}{X_{i_A}X_{k_A} + X_{i_B}X_{k_B}} < \delta. \quad (4.3)$$

The inliers-detection method is summarized in Algorithm 2. After checking each feature i with all other features k for consistency, those with the highest agreement form a set of robust consistent inliers Q . Other features are then checked for consistency with this set and are added to it if they are consistent. The remaining matches are outliers and are thus removed.

Algorithm 2 Inliers-detection algorithm

for $i = 1 \rightarrow TotalNumberOfMatches$ **do**

for $k = 1 \rightarrow TotalNumberOfMatches$ **do**

if $\frac{X_{iA}X_{kA} - X_{iB}X_{kB}}{X_{iA}X_{kA} + X_{iB}X_{kB}} \leq \delta$ **then**

$mark(i) \leftarrow mark(i) + 1$

end if

end for

end for

- Features with the highest marks are inliers as they are the most consistent
 - Other features are checked again for consistency with high-mark inliers and those that prove to be consistent are considered inliers too
 - Inconsistent matches are removed
-

4.2 3D-to-3D Motion Estimation

After extracting the set of inliers Q , consisting of two matched sets of 3D features X_a and X_b , motion between the two frames is calculated using the method proposed by Arun et al. [2], which consists of calculating the rotation using SVD, followed by the calculation of the translation from the centroids of the two 3D sets

$$USV^T = svd[(X_a - \bar{X}_a)(X_b - \bar{X}_b)^T], \quad (4.4)$$

$$R_b = VU^T, \quad (4.5)$$

$$t_b = \bar{X}_b - R\bar{X}_a, \quad (4.6)$$

where X_a and X_b are sets of corresponding 3-D points, \bar{X}_a and \bar{X}_b the centroids of the 3-D sets of points X_a and X_b respectively, and R_b and t_b the rotation and translation mapping the first set of 3D-points to the second set.

4.3 Imposing the 3D Structure

The algorithm described above, allows us to calculate the motion in a 2D plane (three degrees of freedom). However, we have 5 degrees of freedom for a wheeled robot:

- translation along the x-axis
- translation along the y-axis
- yaw
- pitch
- roll

There is no translation along the z -axis as the robot is in continuous contact with the ground. The translations along the x-axis, the translation along the y-axis and the variation in the yaw are calculated using the vision algorithm while the variations in the pitch and the roll remain unknown. This ambiguity can be resolved by coupling the camera with a tilt sensor (such as an IMU or an inclinometer).

After calculating the displacement in x, y and yaw between frame $n - 1$ and frame n using the vision algorithm, the variations in the pitch and the roll are imposed using the data from the inclinometers:

$$P_{5D} = P_{vision} * P_{pitch} * P_{roll}, \quad (4.7)$$

Where P_{5D} is the projection matrix representing the full five dimensional motion between frame $n - 1$ and frame n , P_{pitch} is the projection matrix containing the variation in the pitch, P_{roll} is the projection matrix containing the variation in the roll and P_{vision} is the projection matrix representing the local planar motion calculated using the vision algorithm.

After calculating the relative motion (P_{5D}) between each frame $n - 1$ and n in the sequence, the projection matrix T_n describing the absolute position and orientation of each frame n is then calculated as:

$$T_n = T_1 * P_1 * P_2 * \dots * P_{n-1} \quad (4.8)$$

CHAPTER 5

CALIBRATION

5.1 Experimental Setup

For experimental evaluation, we equipped a Pioneer 3 robot [1] with a Sony DFW-VL500 camera [32]. The camera is rigidly fixed to the robot and directed downwards towards the ground. It is slightly tilted forward to increase the field of view as can be seen in Fig. 5.1.



Figure 5.1: A picture of the experimental setup showing the camera and the robot

To evaluate the accuracy of our proposed algorithms, we ensure that all the experimental runs start and end at the same location. We mark the ground where we started the run and make sure that the robot gets finally back to the same initial position. Accordingly, the accuracy of our method is captured via the

following error function.

$$error = \frac{|FinalPosition - InitialPosition|}{TotalDistance}. \quad (5.1)$$

After fixing the camera to the robot, the rotation between the camera and a virtual downward-looking camera is calculated. For our experiments, we calculated this rotation using Bouguet's camera calibration toolbox for MATLAB [7] by placing a checkerboard on the ground and capturing an image from the camera which is fixed to the robot. This is the most critical step in the set up and must be handled with much care as experiments proved that the motion estimates are sensitive to it. To gauge the effect of correctly computing the pitch angle on our algorithm, for a 100 meter run (Fig. 5.2) our motion estimation is tested with three different values for the pitch angle, one is the correct pitch angle while the other two are one degree off the correct value. The considerable effect on motion estimation for one degree error in the pitch angle is clearly visible.

However, this rotation can't be calculated up to the needed accuracy in a straight forward manner. The rotation contains three parameters: pitch, roll and yaw. Those three parameters change with changing the position of the camera and so we can't take different images, compute the rotation for each and use the average between them as they will have different values for each position. As a work-around, a manual optimization is performed to reach to the accurate value.

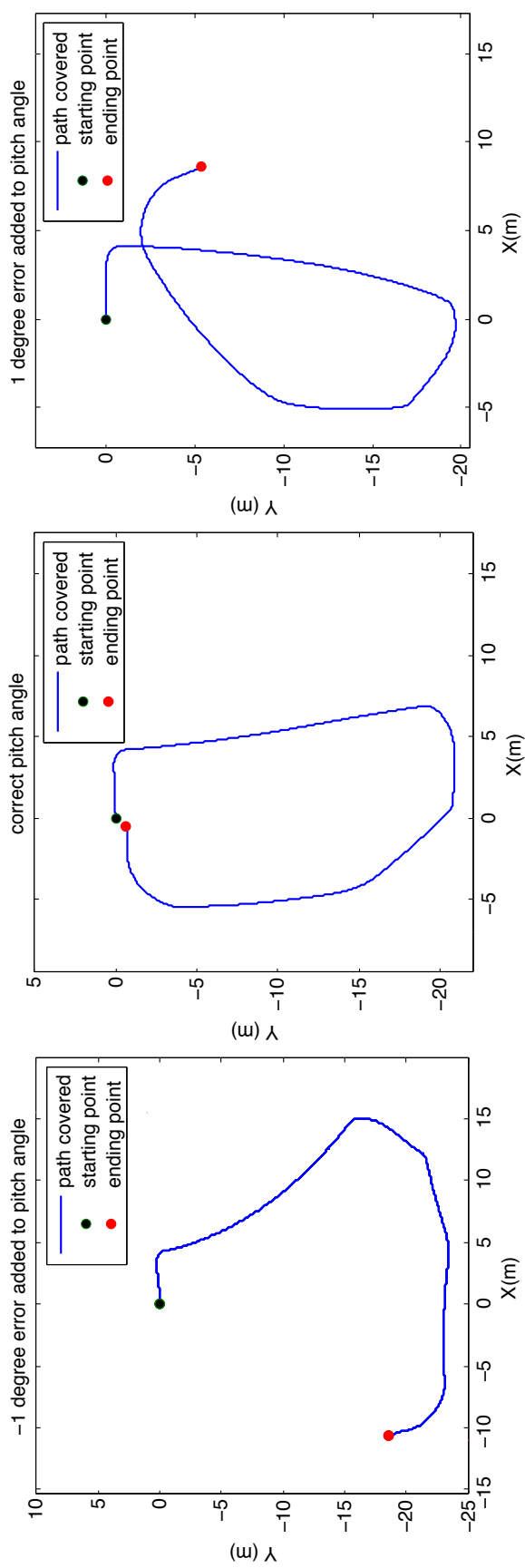


Figure 5.2: Effect of the pitch angle on our proposed algorithm.

5.2 Calibrating the Rotation Matrix Between the Tilted Camera and the Vertical

As mentioned above, the rotation matrix between the tilted camera and the vertical is calculated using vision algorithms from one image of a checkerboard placed on the ground plane. However, the calculated rotation is still within an error range of 1 degrees and not up to the needed accuracy. Moreover, it is not possible to use more than one image for the calculations as the rotation matrix will differ between one image and the other (specifically for the yaw angle).

To solve this problem, the value of the rotation calculated from an image of the checkerboard is taken as an initial estimate of the rotation. This value is then used to calculate the path for one of the experiments performed. Since all the runs started and ended at the same position, it is expected that the path calculated should be a closed loop. However, this initial estimate is not observed to be a closed loop.

Between the three angles constituting the rotation (pitch, roll and yaw), the pitch angle is the angle having the biggest impact on the results. This angle is the one to be optimized and calibrated until an optimal value which best represents the rotation between the camera and the vertical is attained. So, after getting the first estimate of the path, the value of the pitch angle is tweaked (increased by 0.5 degrees for example) and the path is re-calculated with the new rotation. The impact of increasing the pitch angle on the path assessed and we have one of the cases below:

- The results obtained are worse than the first results (the loop was open and now it is more open for example).

In this case, we conclude that we should have decreased the value of the

pitch angle instead of increasing it. So the initial estimate of the pitch angle is decreased by 0.5 degrees and the impact is re-assessed.

- The results are obtained are better but still need some adjustments:
 - The loop was open. It is better now but it is still open: Increase the pitch angle by more than 0.5 degrees and re-calculate.
 - The loop was open. It is better now closed more than needed: Increase the pitch angle by less than 0.5 degrees and re-calculate.

- The results are good

After this tweaking and tuning, we arrive to the value of the pitch angle which gives the best results for the first experiment. This value is then used to calculate the path for the other experiments and the error is calculated for each one. It is either that the calibrated rotation fits the other experiments too or that it still needs more tuning. In case more tuning is needed, one of the other experiments should be chosen to perform the optimization on it. It is advised to choose one of the experiments which still show the lowest accuracy (farthest from loop closure). The pitch angle is then optimized for this experiment (let us say Experiment 5) and the mean between the pitch angle obtained by optimizing for Experiment one and the one obtained by optimizing for Experiment 5 is calculated. This usually should be enough for an acceptable estimation of the rotation between the camera and the vertical. However, if more accuracy is needed, the above process can be repeated again for more than two runs. The mean between the values which best fit the different runs can then be used with confidence for future experiments.

5.3 Scale Estimation

In conventional monocular visual odometry algorithms, the path is calculated up to scale. This means that, for example, we can't identify if the place we are moving in is a doll's house, a real one or a giant one. However, in the algorithm proposed above, the distance between the camera and the ground is known. This allows us to calculate the real 3D coordinates of the features (unscaled) and thus the absolute motion of the camera and not a relative scaled one. This brings us to the question of how to calculate the distance between the camera and the ground with enough accuracy.

The depth (the distance between the camera and the ground) is directly proportional to the scale of the path estimated. This means that if the depth is estimated as d or as $10d$, the only difference will be in the length of the path and not on its form or on the loop closure. However, a scale is essential to fit the path correctly on a map without trial and error especially if the algorithm is implemented in real time. This leaves us with the need of an accurate estimation of the depth.

In order to estimate the depth, two positions separated by a known distance l are marked on the ground. Then, the robot is placed in the first position and moved to the final position after which the path traversed by the robot is calculated using the above vision algorithm. The distance between the camera and the ground is assumed to be 1. After calculating the path, the calculated distance between the initial position of the robot and its final position would be x (the distance calculated by the vision algorithm). However, the real distance is l . The scale by which the calculated path must be scaled to get the absolute motion would then be $scale = x/l$ and the distance between the camera and the ground would also be $d = x/l$. Of course, the more this calibration step is repeated the

better the accuracy of the scale we are getting.

5.4 Coupling the tilt sensors

Several runs were performed on a 2D plane to validate the accuracy of the vision algorithm after which two rotation sensors were coupled to the camera to calculate the full 5D motion of the mobile robot. The sensors used were US Digital T7 one dimensional Inclonometers. A simple platform was manufactured to ensure that the two sensors are perpendicular and correctly aligned and fixed on the robot. The track which was chosen for testing has a steep slope and four sharp turns. Moreover, the turns are on the slopes and not on the straight ground which is more challenging for the algorithm as the 5 parameters are varying altogether.

After gathering the data, the obtained results were noisy. The source of the noise was the inclinometers and the noise propagated to the calculated path which had a big impact on the results. A polynomial fit was used to remove the vibrations but even a 10 degree-polynomial didn't form a good fit (see fig. 5.3). Instead of increasing the degree of the polynomial to fit the whole set of data, the data was divided into several independent intervals and a 10-degree polynomial fit was independently used on each to remove vibrations. The number of intervals to which the data should be divided and the degree of the polynomial varies for each experiment and is dependent on the slopes present and the variations in the pitch and the roll throughout the path.

Once the noise from the tilt sensors was removed, the path calculated by the algorithm was smooth. In order to validate the path, it was imposed on a google satellite image (the start point, end point, and the path boundaries being visible on the map) but it seemed to be off the expected track. However, after further investigations, I noticed that the satellite image was really a "top view". It

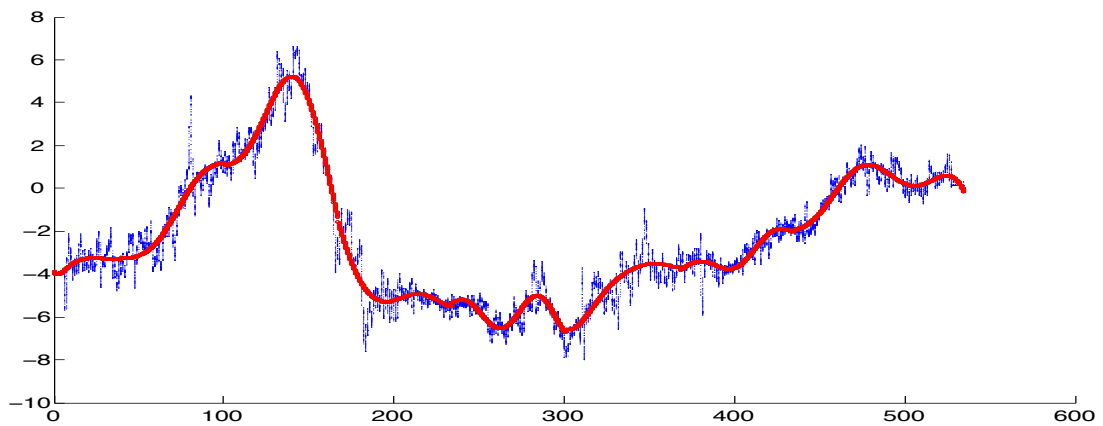
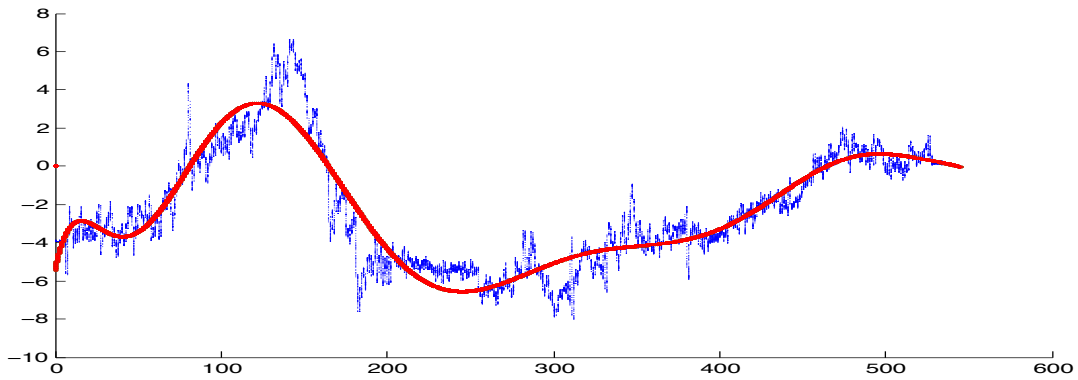


Figure 5.3: Top: 10 degree polynomial fitted to the tilt sensor noisy data. Bottom: 10 degree polynomial fitted to the tilt sensor noisy data after dividing it into several intervals.

was taken from an angle as one of the sides of the building was clearly visible in it while the other was not. Comparing the dimensions on the image to the real ones, the angle at which the satellite had taken the photo was estimated by around 12 degrees. After imposing this inclination on the path estimated by the algorithm and projecting it on the map, the results were much better. The results obtained are discussed in the coming section.

CHAPTER 6

RESULTS

6.1 Results for the runs on a 2D plane

The vision algorithm was tested on several surfaces and for different distances. After several experimental runs, the computed accuracy of our method reached 98.75% for a run of 461 m and increased to 99.44% for smaller distances of up to 90 meters. A summary of the results is presented in Table (6.1). Two of the runs (namely 1 and 3) are depicted in Fig. (6.1) and (6.2) respectively. These runs were done outdoors and a sample of the ground images is shown in the figures as well. Appendix 1 contains the detailed results of the different runs that were performed.

A frame rate of between six and ten Hertz was used for our experiments as a compromise between processing speed and minimum overlap between consecutive frames. Tests were also repeated at a higher frame rate of fifteen Hertz to investigate its affect, but no improvement resulted. In fact, almost no change was detected and the accuracy was even lower sometimes for higher frame rates. The incremental nature of the error explains this phenomena as the more frames you analyze, the more noise is susceptible to corrupt the system.

The algorithm was then tested in the case of where features were sparse. To simulate such a environment a higher shutter speed was set on the camera to decrease the quality of the images and obtain less features, where the number of matches between one image and the other was sometimes as low as 30 features. On contrary to other algorithms that fail in such settings, our algorithm sustained an acceptable accuracy of around 4% for runs reaching up to 32 meters.

Table 6.1: The results from the different runs showing the error and the distance covered in each

Run	Frames	Freq.	Distance	Error
1	4849	10Hz	418.1 m	5.25 m (1.25%)
2	1394	6Hz	100.2 m	1.49 m (1.48%)
3	1082	6Hz	90.4 m	0.5 m (0.56%)
4	357	6Hz	26.6 m	0.13 m (0.46%)
5	302	6Hz	26.4 m	0.23 m (0.91%)
6	364	6Hz	20.4 m	0.16 m (0.78%)
7	285	6Hz	15.8 m	0.12 m (0.78%)
8	256	6Hz	14.7 m	0.23 m (1.6%)
9	391	10Hz	12.5 m	0.21 m (1.67%)
10	419	10Hz	12.53 m	0.049 m (0.35%)
11	461	10Hz	13.66 m	0.058 m (0.43%)

In short, for all the experiments conducted and over the thousands of frames the algorithm was tested on, it proved to be robust and didn't fail except when the image comparison method failed. There were problems on reflective surfaces like polished flagstones where it is difficult to match images correctly. However, in real life scenarios this is usually not a problem since most surfaces, ranging from indoor and outdoor pavements to gravel and asphalt roads, contain enough features for robust performance of the image comparison algorithm.



Figure 6.1: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 461meters.

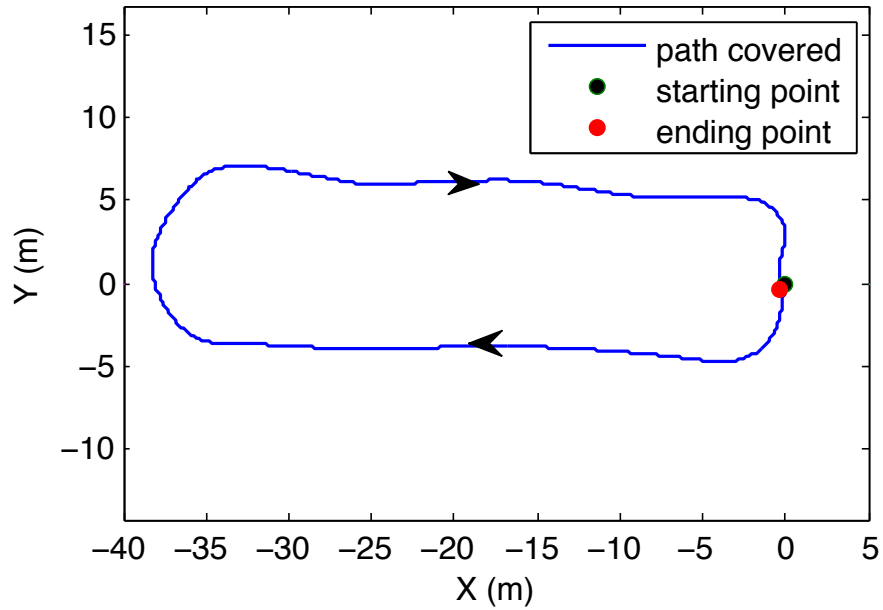


Figure 6.2: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 90.4 meters.

6.2 Results with the tilt sensors

After the tilt sensors were introduced, the results of the algorithm were compared to the path generated using a GPS and an IMU (the MTi-G IMU). Both results are displayed in fig. 8.12. As you can see in the figure, if we are to assess the final reached position, the GPS would definitely be better as it is generating an absolute measurement (not dependent on previous measurements) whereas the

vision algorithm calculates its position in an accumulative manner (which means that the error propagates throughout the whole sequence). However, the vision algorithm showed to be much more consistent locally than the GPS and the IMU coupled together. In other words, the vision algorithm is much more accurate and consistent on small distances but this accuracy starts to decrease as the length of the path increases to 1Km and more. The reason behind this decay is that a negligible error in the rotation estimation (in the order of 0.1 degrees) will have a huge impact on such long runs whereas as mentioned above, the GPS is estimating its position from scratch without using previously-calculated polluted data.

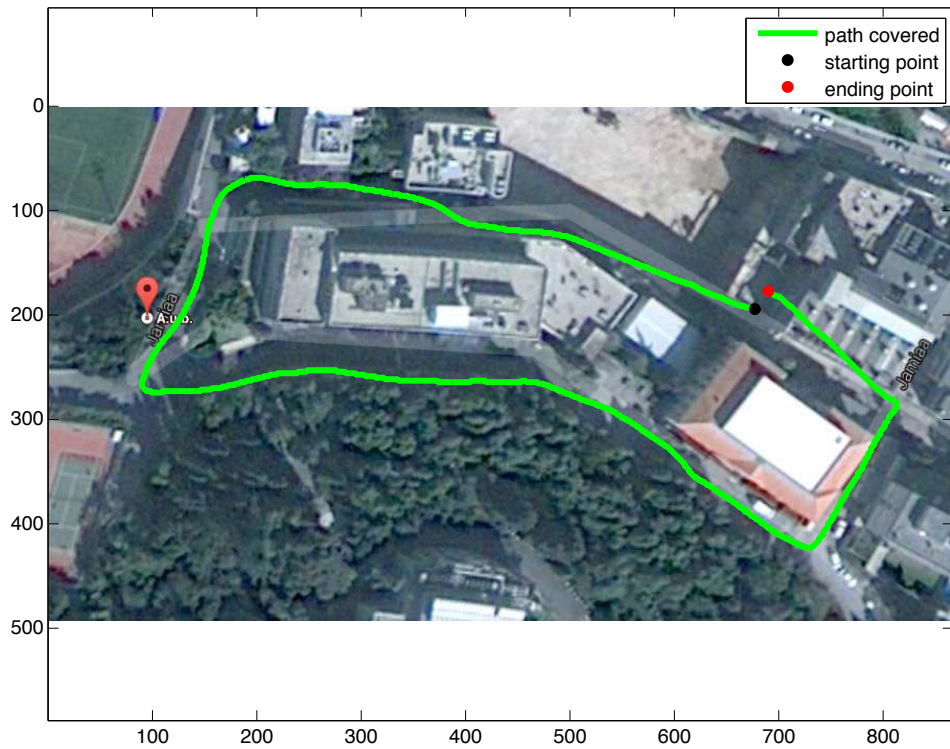


Figure 6.3: Above: Estimated trajectory on a non-planar path using my algorithm. Below: Estimated path using fused data from a GPS and an Inertial Measurement Unit (MTi-G IMU)

The basic limitation of the algorithm is that it is limited to mobile robots only. Moreover, it doesn't work on a rough-off road terrain because in such terrain the ground can't be considered to be locally planar which is a pre-requisite for the algorithm.

It was not possible to compare our method to conventional vision algorithms like the famous 8 points and 5 points Structure From Motion algorithms as these algorithms don't give acceptable results once all the features in the images belong to one plane (the ground plane in my case). This is a mathematical singularity for those algorithms as discussed earlier in the thesis and therefore it was not possible to test another algorithm on the same data set that we got from our experiments.

However, if we are to compare it to the results mentioned in the literature for monocular VOs, we observe that our algorithm is the first to reach an average accuracy of 0.9%. The highest accuracy to be reported before for conventional SFM is 2.5% on long runs and it is worth to note here that this accuracy was obtained in very specific conditions as neither me, nor other colleagues before me were able to re-produce them. Other researchers reported high accuracies of around 2% or even 1% but these were on small runs of less than 5 meters. Our algorithm attains such an accuracy for runs as long as 460 m which proves that it is much more robust and accurate on long distances than the other algorithms present in the literature.

Moreover, the algorithm is the first monocular VO algorithm without any iterative or probabilistic routine. The main steps in monocular VO algorithms are the following:

- Feature extraction
- Feature matching
- Removal of outliers
- Estimating the motion
- Optimizing the final results

The first four steps are common and inevitable for all algorithms whereas real time VOs usually skip the final step to reduce the cost. In the algorithm I presented above I use the conventional feature extraction and matching techniques. The main contribution of the thesis is in steps 3 and 4. Still, and since we are comparing the computational costs of the different VO algorithms, step 4 is to be disregarded because its computational cost is negligible compared to that of the other steps.

All monocular VO algorithms presented in the literature use the iterative RANSAC to remove outliers. They need a minimum of 160 RANSAC iterations (usually more) to remove outliers. However, my algorithm removes outliers with a single iteration by making use of the rigid position of the camera with respect to the ground and the inliers detection method. This means that the computational cost of step 3 is decreased.

In order to assess the impact of replacing RANSAC by the inliers detection method on the computational cost and the accuracy of the algorithm, both algorithms were implemented in MATLAB on an Intel(R) core(TM) i7-3940XM CPU @ 3.00 GHz 3.20 GHz processor with a 32 GB installed memory (RAM). Both algorithms were tested over 1200 frames and the results are presented in fig. 6.4 below. The computational cost was decreased by 58% while the accuracy was still the same for both algorithms.

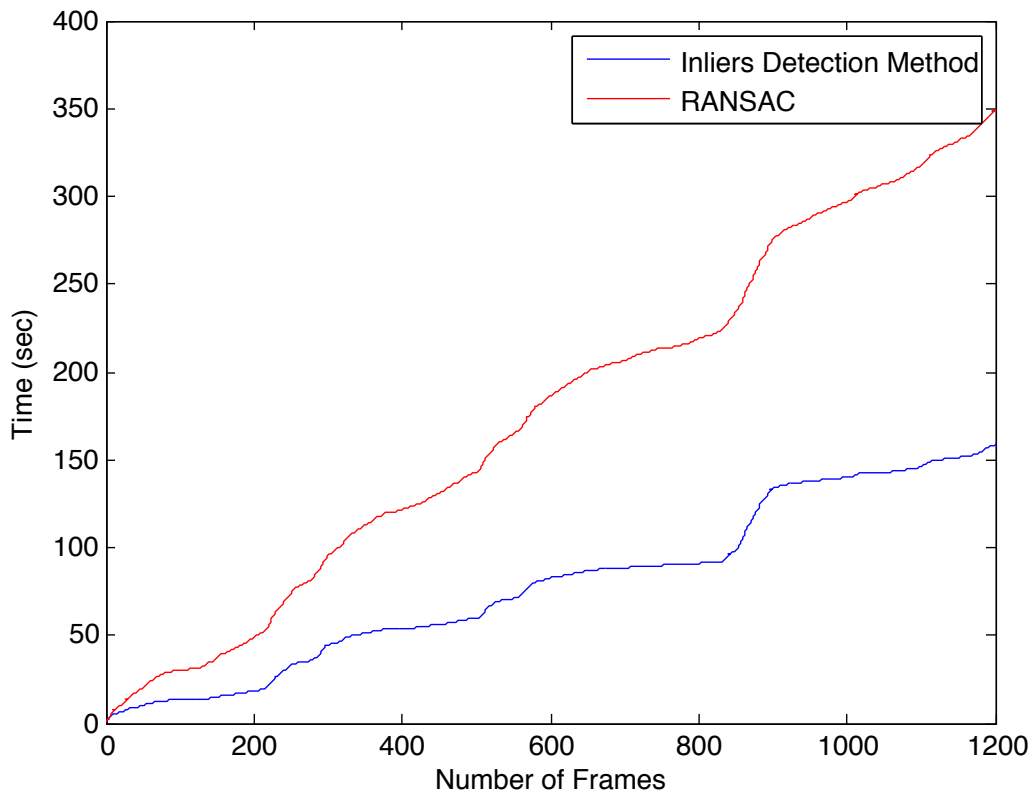


Figure 6.4: This figure shows the time consumed by each of the inliers detection method and RANSAC (with 160 iterations) versus the number of processed frames. An improvement by 58% is observed.

CHAPTER 7

CONCLUSIONS

In this thesis, we presented a hybrid visual odometry algorithm in which we implemented the inliers-detection method on a monocular camera and coupled that with the geometry of a virtual downward-looking camera to eliminate uncertainty in the depth. The significance of this approach is that it calculates the motion from a monocular camera without iterative or probabilistic techniques neither for motion estimation nor for outliers rejection. The result was an improvement in the accuracy with real-time speeds of implementation. The computational cost was decreased by 58% and the error at loop closure was as low as 1.25% on loops of approximately 418 meters.

Moreover, the vision algorithm was coupled with two inclinometers to calculate the full motion of the robot in the presence of slopes. Other ideas like auto calibration to continuously re-estimate the rotation between the camera and a virtual downward-looking camera can also be investigated. This is needed for a vehicle that is equipped with a suspension system like that of a car, where the body rotates with respect to the ground while negotiating sharp turns. Another potential proposition would be to use a particle filter with a laser range finder and force-fit the robot into a pre-defined map (or maybe a google satellite image). This would highly reduce the impact of the drift which results mostly from errors in the estimation of the rotation on long runs.

CHAPTER 8

APPENDIX

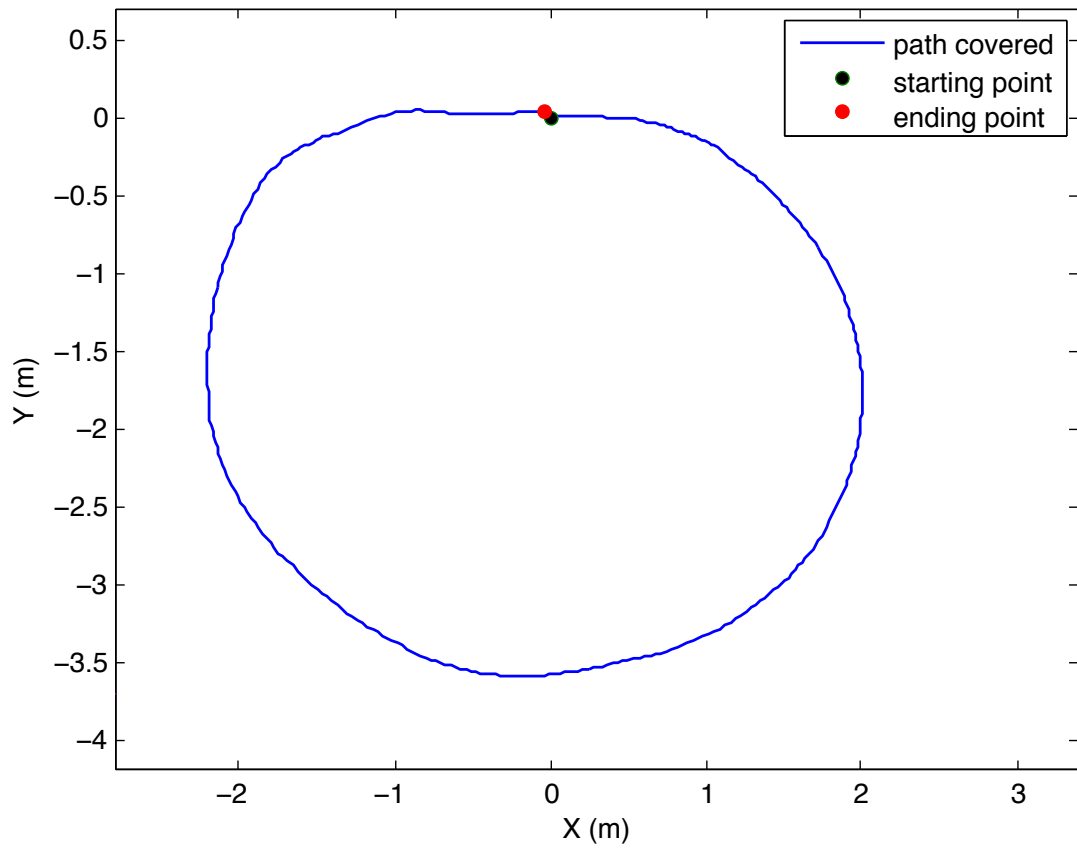


Figure 8.1: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 13.66 meters and an error of 0.43%.

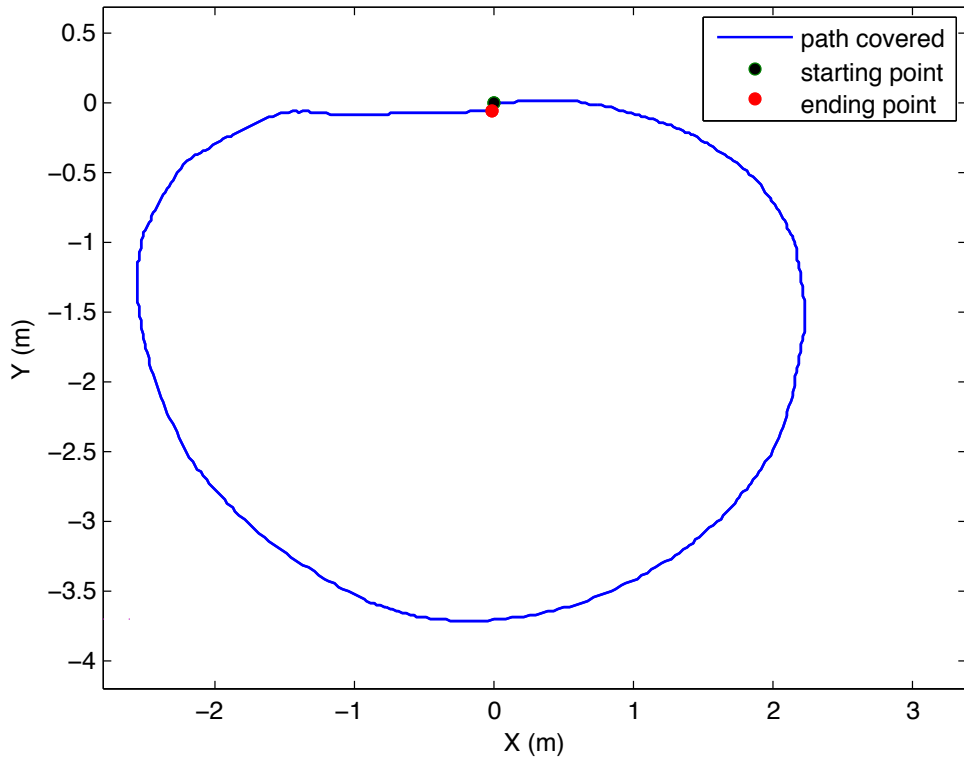
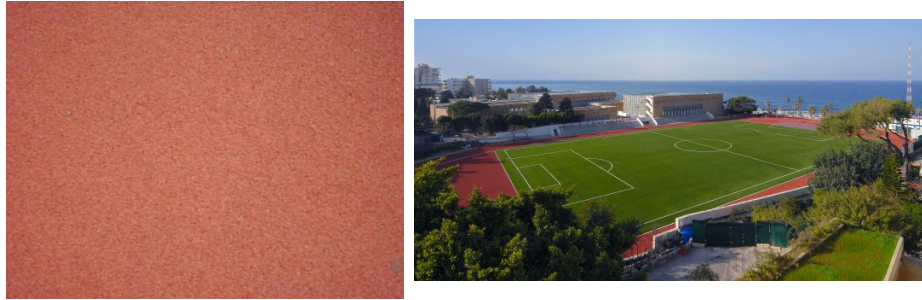


Figure 8.2: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 12.53 meters and an error of 0.35%.

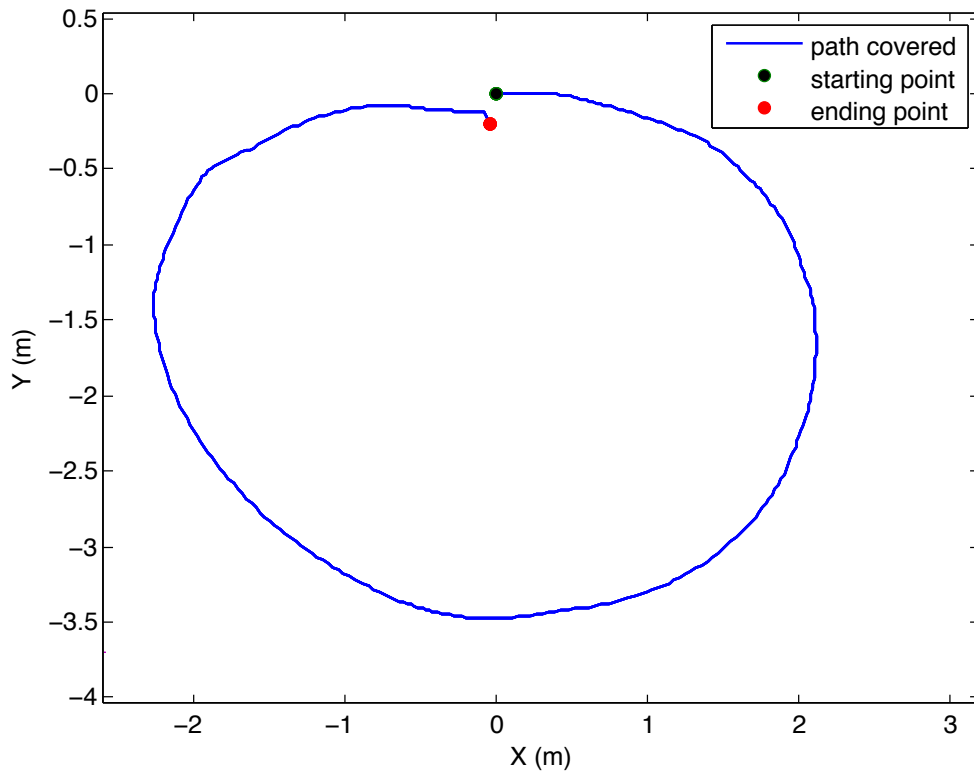
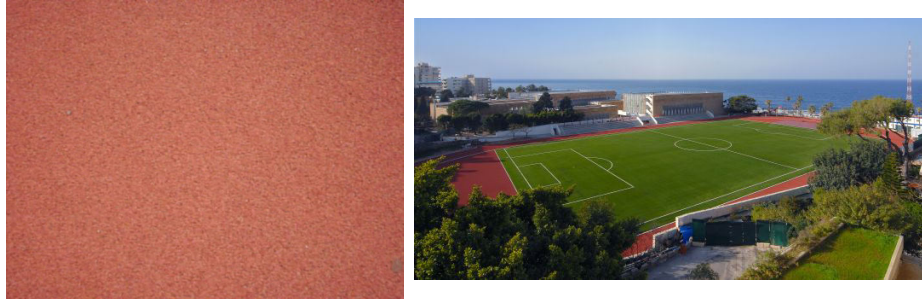


Figure 8.3: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 12.5 meters and an error of 1.67%.

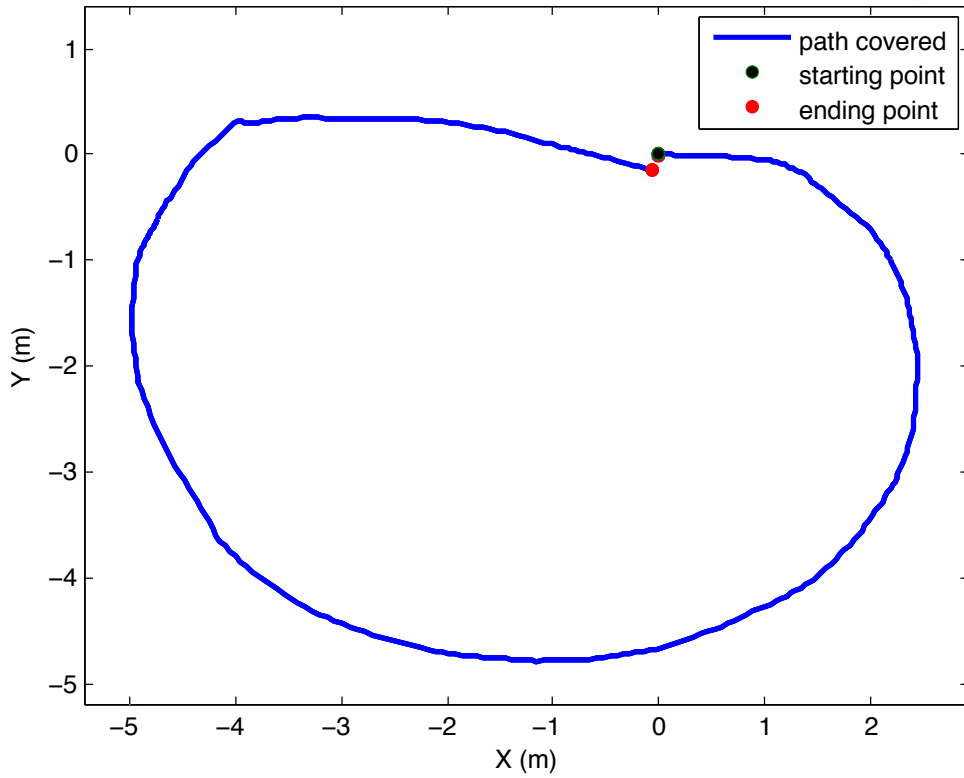


Figure 8.4: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 20.4 meters. The error is 0.78%.

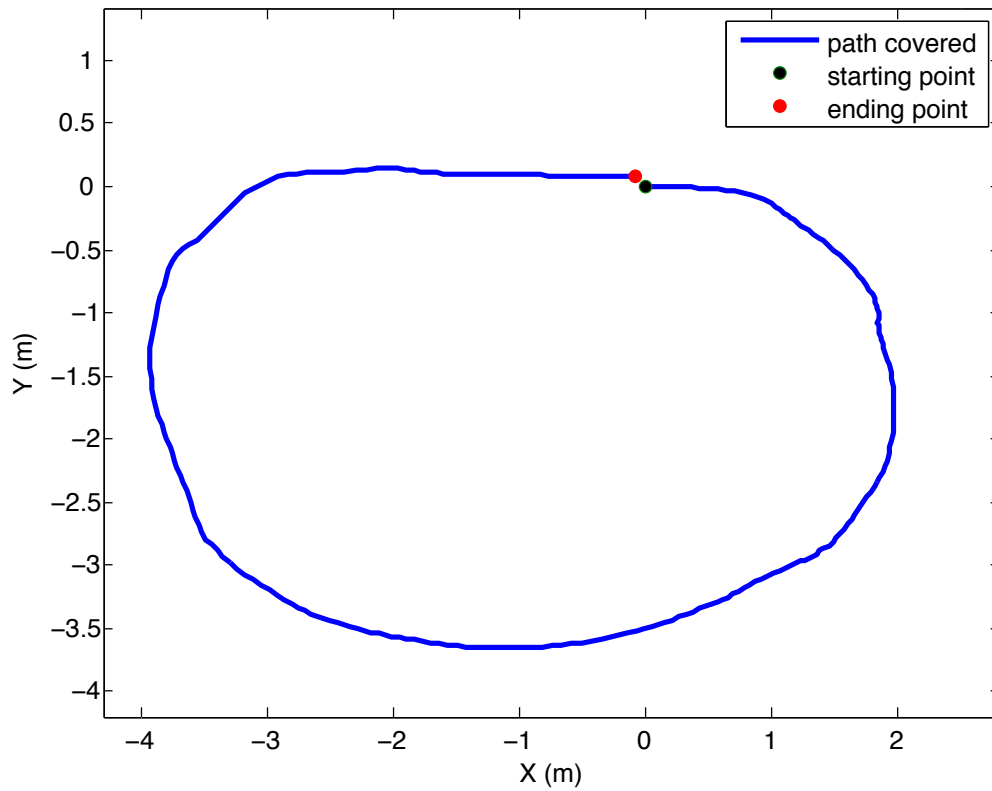


Figure 8.5: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 15.8 meters. The error is 0.78%

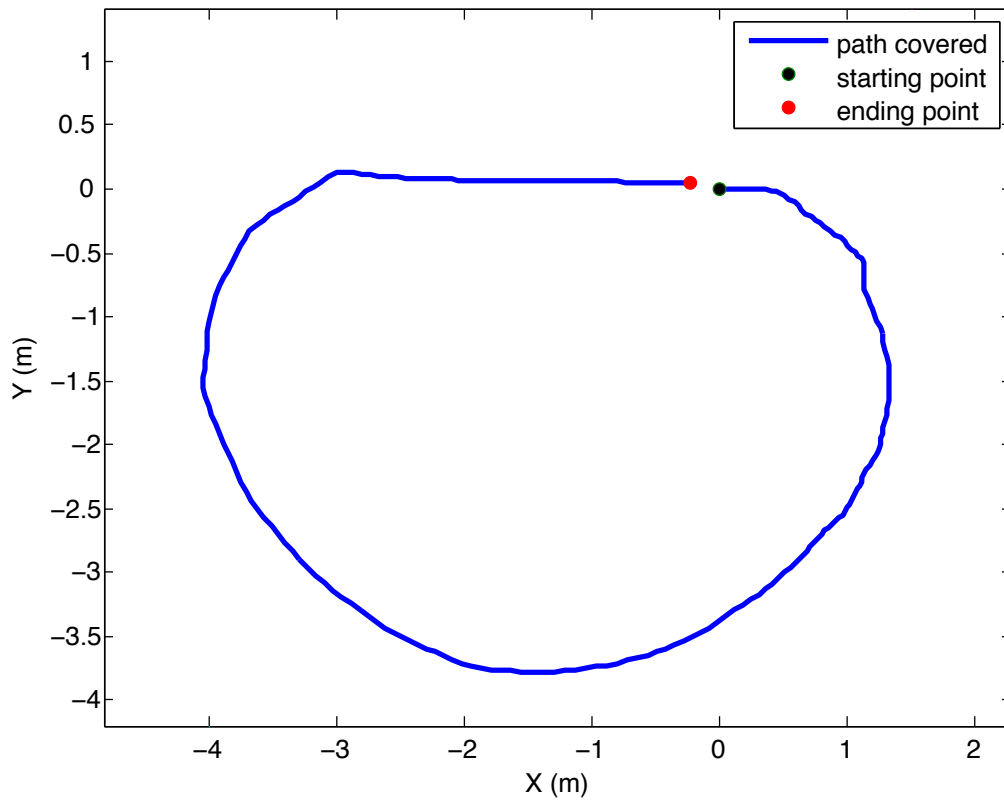


Figure 8.6: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 14.7 meters. The error is 1.6%

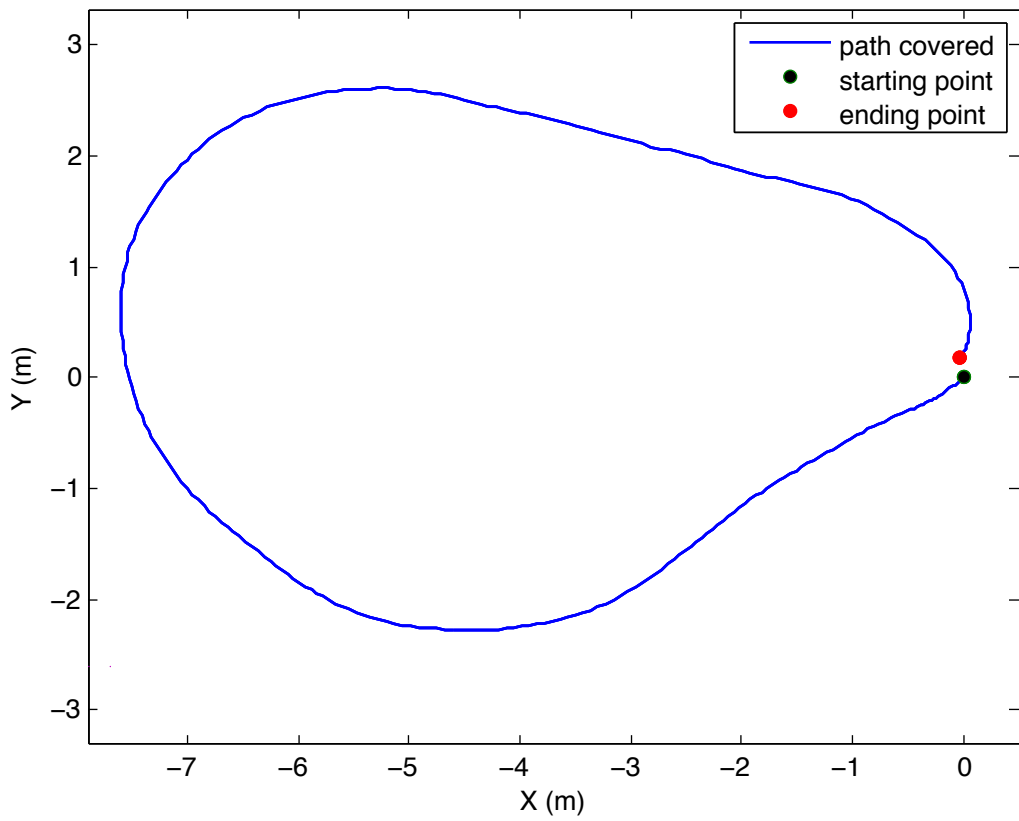


Figure 8.7: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 26.4 meters. The error is 0.91%

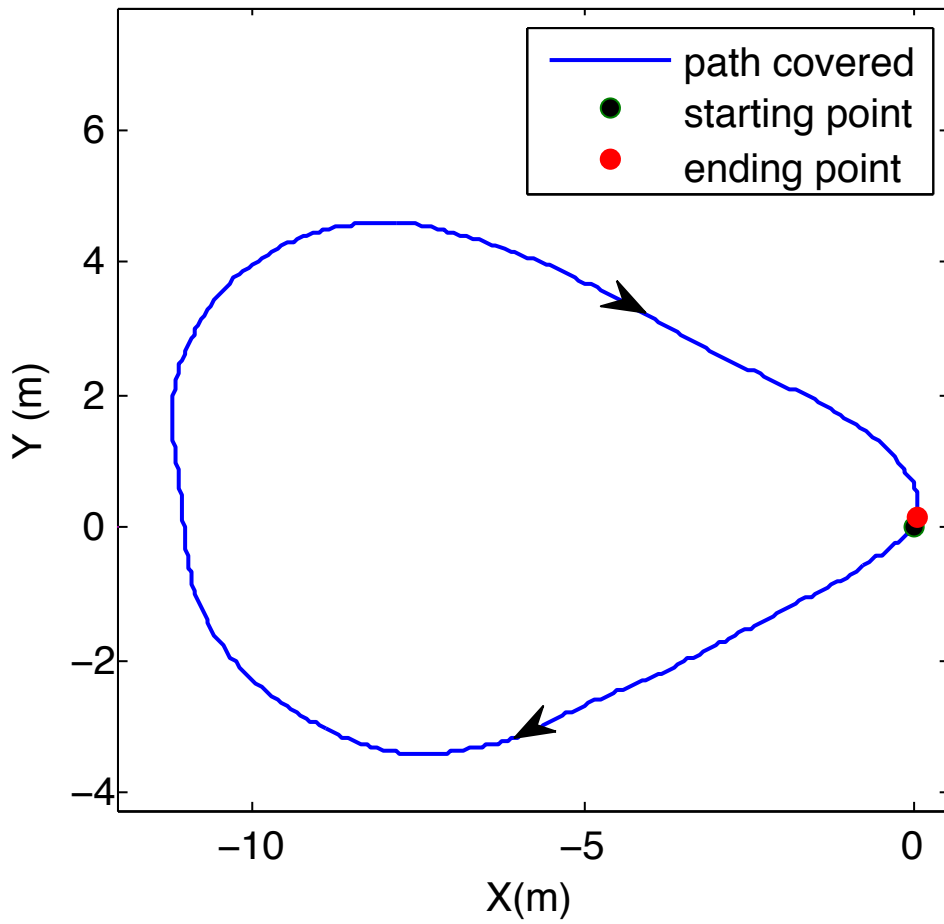


Figure 8.8: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 26.6 meters. The error is 0.46%

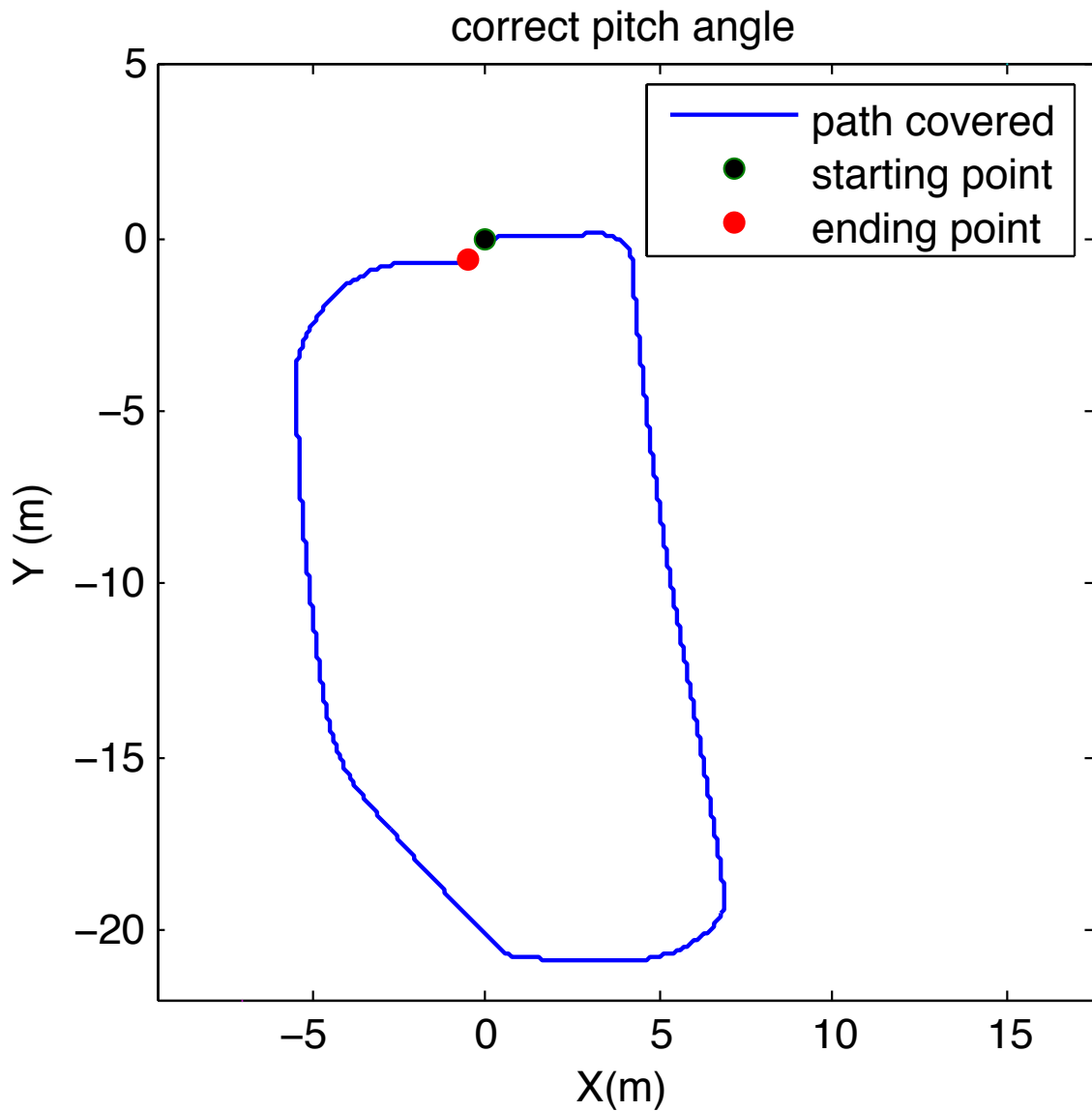


Figure 8.9: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 100.2 meters. The error is 1.48%

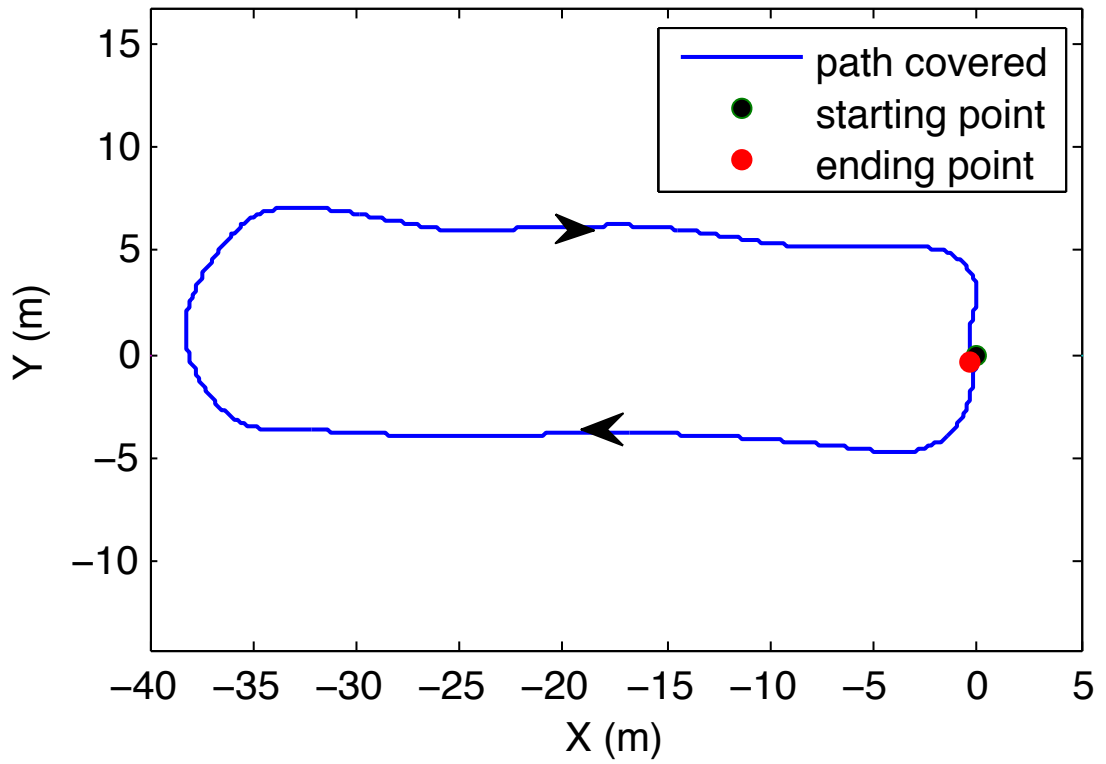


Figure 8.10: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 90.4 meters. the error is 0.56%

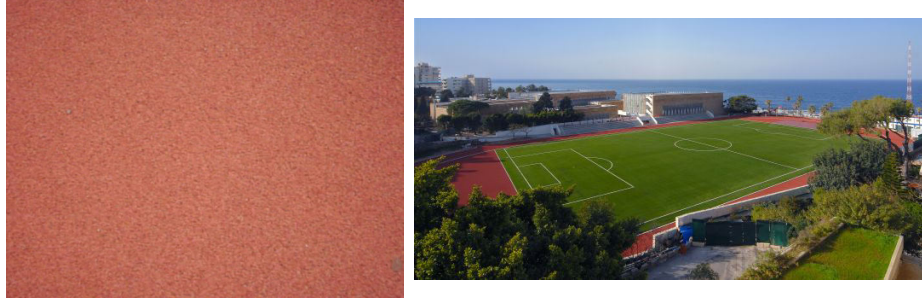


Figure 8.11: Top: picture of the experimental environment (right) and sample texture of the ground (left). Bottom: estimated trajectory using our system for a total run of 418.1 meters. The error is 1.25%

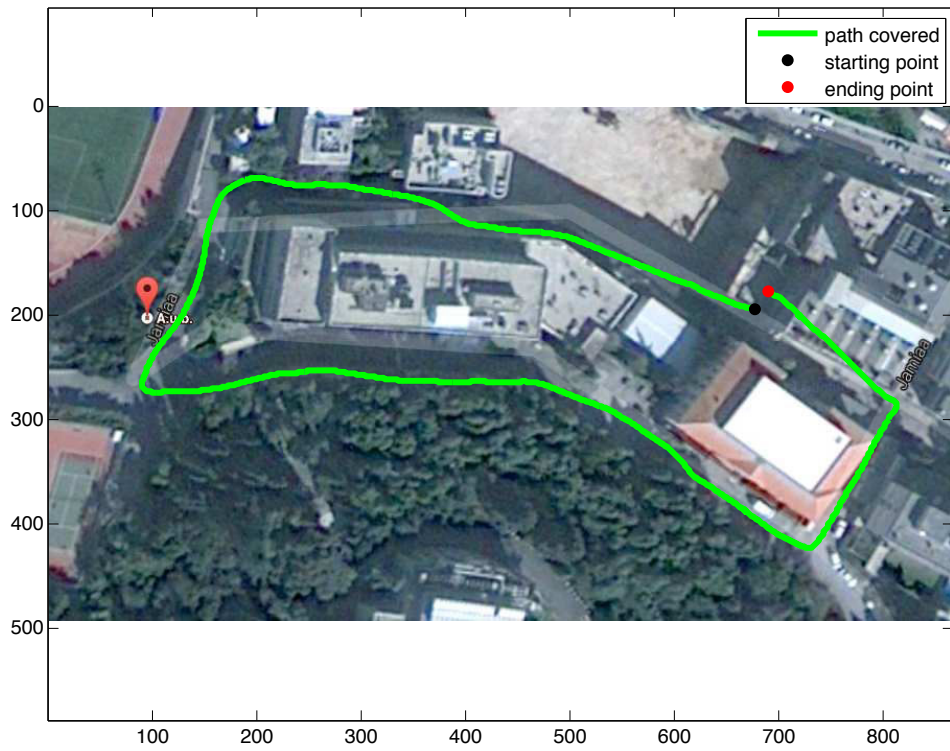


Figure 8.12: Above: Estimated trajectory on a non-planar path using my algorithm. Below: Estimated path using fused data from a GPS and an Inertial Measurement Unit (MTi-G IMU)

REFERENCES

- [1] Adept. <http://www.mobilerobots.com/researchrobots/p3at.aspx>.
- [2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, 1987.
- [3] H. Badino, D. Huber, and T. Kanade. Visual topometric localization. In *Intelligent Vehicles Symposium*, pages 794–799, 2011.
- [4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*, 2006.
- [5] P. Biber and T. Duckett. Experimental analysis of sample-based maps for long-term slam. *The International Journal of Robotics Research*, 28(1):20–33, 2009.
- [6] J. Borenstein, H. R. Everett, and L. Feng. *Where am I? Sensors and Methods for Mobile Robot Positioning*.
- [7] Jean-Yves Bouguet. http://www.vision.caltech.edu/bouguetj/calib_doc/.
- [8] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping: part i. *Robotics and Automation Magazine*, 13(2):99–110, 2006.
- [9] E. Eade and T. Drummond. Monocular slam as a graph of coalesced observations. In *International Conference on Computer Vision (ICCV)*, 2007.
- [10] David Fernandez and Andrew Price. Visual odometry for an outdoor mobile robot. In *IEEE Conference on Robotics, Automation and Mechatronics*, 2004.

- [11] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM, Volume 24, Issue 6, Pages 381 - 395*, 1981.
- [12] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part ii matching, robustness, optimization, and applications. In *Robotics & Automation Magazine, IEEE*, 2012.
- [13] P. Furgale and T. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of field robotics*, 27(5):534–560, 2010.
- [14] Roland Goecke, Akshay Asthana, Niklas Pettersson, and Lars Petersson. Visual vehicle egomotion estimation using the fourier-mellin transform. In *Intelligent Vehicles Symposium*, 2007.
- [15] C. Harris and J. Pike. 3d positional integration from image sequences. In *Proc. Alvey Vision Conf.*, 1988.
- [16] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151, 1988.
- [17] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.
- [18] H. Hirschmuller, P.R. Innocent, and J.M. Garibaldi. Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. In *Control, Automation, Robotics and Vision (ICARCV)*, 2002.
- [19] Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *International Conference on Intelligent Robots and Systems*, 2008.
- [20] Tim Kazik and Ali Haydar Goktogan. Visual odometry based on the fourier-mellin transform for a rover using a monocular ground-facing camera.

- In *IEEE International Conference on Mechatronics (ICM)*, 2011.
- [21] Qifa Ke and Takeo Kanade. Transforming camera geometry to a virtual downward-looking camera: Robust ego-motion estimation and ground-layer detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.
- [22] G. Klein and D. W. Murray. Parallel tracking and mapping on a camera phone. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009.
- [23] Konologie. Towards lifelong visual maps. In *IEEE Conference on Intelligent Robots and Systems (IROS)*, pages 1156–1163, 2009.
- [24] H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. In *Nature*, vol. 293, no. 10, pp. 133–135, 1981.
- [25] Steven Lovegrove, Andrew J. Davison, and Javier Ibanez-Guzman. Accurate visual odometry from a rear parking camera. In *Intelligent Vehicles Symposium*, 2011.
- [26] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.
- [27] H. McDonald, M. Kaess, C. Cadena, J. Neira, and J. Leonard. 6-dof multi-session visual slam using anchor nodes. In *European Conference on Mobile Robots (ECMR)*, pages 69–76, 2011.
- [28] D. Nister. Preemptive ransac for live structure and motion estimation. *ICCV*, 199-206, 2003.
- [29] D. Nister. An efficient solution to the five-point relative pose problem. Technical report, Princeton, 2006.

- [30] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.
- [31] Navid Nourani-Vatani and Paulo Vinicius Koerich Borges. Correlation-based visual odometry for ground vehicles. In *Journal of Field Robotics Issue Journal of Field Robotics Volume 28, Issue 5, pages 742–768*, 2011.
- [32] Sony Corporation of America. <http://www.sony.com/index.php>.
- [33] G. Ros, A. Sappa, D. Ponsa, and A. Lopez. Visual slam for driverless cars: a brief survey. In *Intelligent Vehicles Symposium*, 2012.
- [34] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, October 2005.
- [35] R. Royer, M. Lhuillier, M. Dhome, and J. Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision (IJCV)*, 74(3):237–260, 2007.
- [36] D. Scaramuzza. Visual odometry [tutorial]. In *Robotics & Automation Magazine, IEEE*, 2011.
- [37] Jean-Philippe Tardif, Yanis Pavlidis, and Kostas Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *International Conference on Intelligent Robots and Systems*, 2008.
- [38] DAVID TITTERTON and JOHN WESTON. *Strapdown Inertial navigation Technology 2nd Edition*. THE INSTITUTION OF ENGINEERING AND TECHNOLOGY, 2004.

- [39] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision (IJCV)*, 9(2):137–154, 1992.
- [40] C. Valgren and A. Lilienthal. Sift, surf and seasons: Long-term outdoor localization using local features. In *European Conference on Mobile Robots (ECMR)*, pages 253–258, 2007.
- [41] Hui Wang, Kui Yuan, Wei Zou, and Qingrui Zhou. Visual odometry based on locally planar ground assumption. In *International Conference on Information Acquisition*, 2005.
- [42] Guoshen Yu and Jean-Michel Morel. Asift: An algorithm for fully affine invariant comparison. *Image Processing On Line*, 2011.