

**VISUALLY-GUIDED WALKING REFERENCE MODIFICATION FOR  
HUMANOID ROBOTS**

**by  
KAAN CAN FIDAN**

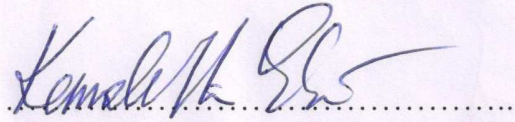
**Submitted to the Graduate School of Engineering and Natural Sciences  
in partial fulfillment of  
the requirements for the degree of  
Master of Science**

**Sabanci University  
August 2012**

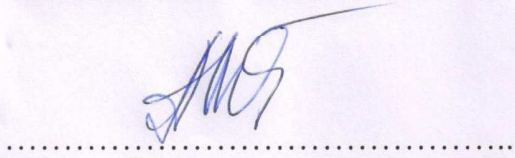
**VISUALLY-GUIDED WALKING REFERENCE MODIFICATION FOR  
HUMANOID ROBOTS**

APPROVED BY:

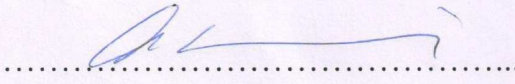
Assoc. Prof. Dr. Kemalettin ERBATUR  
(Thesis Supervisor)



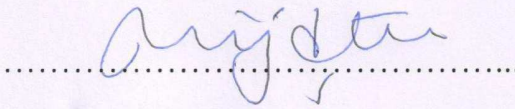
Prof. Dr. Asif SABANOVIC



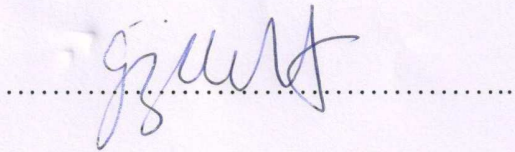
Prof. Dr. Aytül ERÇİL



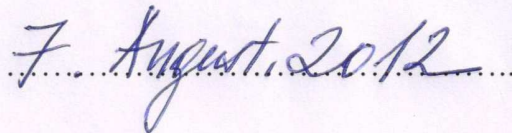
Assist. Prof. Dr. Müjdat ÇETİN



Assoc. Prof. Dr. Gözde ÜNAL



DATE OF APPROVAL:



## ABSTRACT

Humanoid robots are expected to assist humans in the future. As for any robot with mobile characteristics, autonomy is an invaluable feature for a humanoid interacting with its environment. Autonomy, along with components from artificial intelligence, requires information from sensors. Vision sensors are widely accepted as the source of richest information about the surroundings of a robot. Visual information can be exploited in tasks ranging from object recognition, localization and manipulation to scene interpretation, gesture identification and self-localization.

Any autonomous action of a humanoid, trying to accomplish a high-level goal, requires the robot to move between arbitrary waypoints and inevitably relies on its self-localization abilities. Due to the disturbances accumulating over the path, it can only be achieved by gathering feedback information from the environment.

This thesis proposes a path planning and correction method for bipedal walkers based on visual odometry. A stereo camera pair is used to find distinguishable 3D scene points and track them over time, in order to estimate the 6 degrees-of-freedom position and orientation of the robot. The algorithm is developed and assessed on a benchmarking stereo video sequence taken from a wheeled robot, and then tested via experiments with the humanoid robot SURALP (Sabanci University **R**obotic **R**ese**A**rch **L**aboratory **P**latform).

## ÖZET

İnsansı robotların, yakın gelecekte insanlara yardımcı olmaları beklenmektedir. Gezgin karakteristiğe sahip her robotta olduğu gibi, insansı robotlarda da otonom hareket kabiliyeti çevreyle etkileşimde büyük rol oynamaktadır. Otonomluk, yapay zeka öğeleriyle birlikte algılayıcı verisine ihtiyaç duyar. Robotun çevresi ile ilgili en zengin bilgiyi sağlayan algılayıcılar, görsel bilgi içeren kameralar olarak kabul görmektedir. Görsel bilgi, nesnelere tanıma, yerlerini belirleme ve hareket ettirme gibi uygulamaların yanında sahne anlamlandırılması, jest tanıma ve özkonumlandırma gibi problemlerin çözümünde kullanılabilir.

Bir insansı robotun üst seviye bir amaca hizmet etmek üzere gerçekleştireceği herhangi bir otonom hareket, bulunduğu çevre içerisinde belirli noktalara gitmesini gerektirmekte, ve dolayısıyla robotu özkonumlandırma yeteneğine bağımlı kılmaktadır. Hareketin sürdürüldüğü yol üzerinde gelen etkenlerin yarattığı hataların üstüste eklenmesi sonucu, çevreden bir geri beslemeye ihtiyaç duyulmaktadır.

Bu tez, görsel odometri tabanlı bir yürüyüş yörüngesi düzeltme algoritması sunmaktadır. Bahsedilen yöntemde bir stereo kamera çifti tarafından algılanan üç boyutlu noktalar zaman içinde takip edilerek kamera setinin 6 serbestlik dereceli konum ve oryantasyonu tahmin edilmektedir. Algoritma, geliştirilme aşamasında önceden kaydedilmiş videolar vasıtasıyla denenmiş ve son halini aldığı anda insansı robot SURALP (Sabancı Üniversitesi Robot Araştırmaları Laboratuvar Platformu) üzerinde test edilmiştir.

To my loving family.

## ACKNOWLEDGMENTS

Initially, I would like to express my gratitude towards my thesis advisor Prof. Kemalettin Erbatur, who always found patience to enable my enthusiasm. I was able to pursue my graduate study thanks to his belief in my abilities, and I will always be grateful for this chance. His eagerness to share his immense knowledge and experience, his ability to create a wonderful working environment and the value and time he granted to our often frivolous ideas made his guidance invaluable.

I would also like to thank the rest of my thesis committee: Prof. Asif Sabanovic, Prof. Aytül Erçil, Prof. Gözde Ünal and Prof. Müjdat Çetin for their detailed review, constructive criticism and overall interest that they have shown for my research.

I would like to name fellow SURALP team members Tunç Akbaş, Utku Seven, Emre Eskimez and Selim Özel. I could not imagine better companions to go through this journey. I would also like to thank all the souls dedicated to FENS 1093, especially İlker Sevgen, for their support and friendship.

I am sincerely thankful to Osman Rahmi Fıçıcı and Serhan Coşar who gave their unending support and helped me realize this thesis' work.

Last but not least, to my parents Zekai and Canan Fidan, my little sister Özgecan and my soulmate Nihan Aydın: Thank you for being in my life, I love you.

## TABLE OF CONTENTS

	Page
ABSTRACT	III
ACKNOWLEDGMENTS	VI
TABLE OF CONTENTS	VII
LIST OF FIGURES	IX
LIST OF TABLES	XI
1. INTRODUCTION	1
2. . A SURVEY ON EGOMOTION ESTIMATION AND ITS APPLICATIONS IN ROBOTICS	4
2.1. Methods and Application Areas	4
2.2. Literature Survey on Visual Odometry in Mobile Robotics	6
2.3. Literature Survey on the Self-Localization Problem in Humanoid Robots	13
3. THE EXPERIMENTAL HUMANOID SURALP	18
3.1. Hardware	18
3.2. Walking Reference Generation	22
3.2.1. Straight Walk	22
3.2.2. Walk on Circular Arc Shaped Paths	28
3.3. Basic Walking Control Algorithms	30
3.3.1. Joint Level Control	31
3.3.2. Foot Roll Control	31
3.3.3. Ground Impact Compensation	31
3.3.4. Early Landing Modification	32
4. 3D RECONSTRUCTION VIA STEREO VISION	34
4.1. The Pinhole Camera Model	34
4.2. Lens Distortion	38
4.3. Necessary Information for 3D Reconstruction	39
4.4. Stereo Camera Calibration	40
4.5. Stereo Rectification	41
4.6. 3D Reconstruction Using Rectified Stereo Camera Pairs	43
4.7. Feature Detection	45

4.8. Subpixel Corner Estimation	47
4.9. Stereo Matching	47
5. VISUAL ODOMETRY	53
5.1. Algorithm Workflow	53
5.2. Ground Truth Data	55
5.3. Tracking Corner Features in Time	57
5.4. Camera Pose Estimation	57
6. IMPLEMENTATION ON SURALP	74
7. CONCLUSIONS	79
REFERENCES	80
APPENDIX A - THE LEVENBERG-MARQUARDT METHOD	88
APPENDIX B - RANSAC	91



## LIST OF FIGURES

	Page
Figure 2.1. A computer generated image of a NASA Mars Exploration Rover	9
Figure 2.2. A quadrotor MAV with and RGB-D camera	11
Figure 2.3. An amphibious robot using stereo cameras	12
Figure 2.4. LittleDog and the stereo camera setup	13
Figure 2.5. Two commercial humanoid robots ASIMO and REEM-B	15
Figure 2.6. A photograph of HRP-2, HRP-3 and HRP-4 respectively	16
Figure 2.7. The humanoid robot H7	17
Figure 3.1. Humanoid robot SURALP, dimensions.	19
Figure 3.2. Kinematic arrangement of SURALP	20
Figure 3.3. The hardware architecture of SURALP	22
Figure 3.4. The linear inverted pendulum model	23
Figure 3.5. Forward moving ZMP references with pre-assigned double support phases.	24
Figure 3.6. x and y-direction COM and ZMP references	27
Figure 3.7. x and z-direction foot frame references in as expressed in the world frame.	28
Figure 3.8. SURALP CAD model on a arc shaped walking trajectory.	28
Figure 3.9. Mapping of foot placement locations of a straight walk onto the foot placement locations of a circular arc-following walk.	29
Figure 3.10. The walking controller block diagram	33
Figure 4.1. The pinhole camera model	35
Figure 4.2. The frontal pinhole camera model	35
Figure 4.3. The pixel and the image plane coordinate frames	37
Figure 4.4. Types of distortion	38
Figure 4.5. The stereo calibration process	40
Figure 4.6. Full-frontal-parallel and row-aligned stereo camera pair	41
Figure 4.7. The undistortion and rectification step	43
Figure 4.8. Triangulation with rectified stereo cameras	44
Figure 4.9. FAST corner features	46
Figure 4.10. Depth vs. disparity plot of the stereo camera rig on SURALP	49

Figure 4.11. Depth sensitivity vs. disparity plot	50
Figure 4.12. Zoomed depth vs. disparity plot	51
Figure 4.13. Stereo Correspondences found by SSD	51
Figure 5.1. Samples from the benchmark image sequence	56
Figure 5.2. Method I position estimation (dashed) and ground truth (solid) vs. time	64
Figure 5.3. Estimated angle (dashed) and the ground truth (solid) vs. time	65
Figure 5.4. Estimated (dashed) and ground truth (solid) robot trajectories	66
Figure 5.5. Method II position estimation (dashed) and ground truth (solid) vs. time	68
Figure 5.6. Estimated angle by Method II (dashed) and the ground truth (solid) vs. time	69
Figure 5.7. Estimated trajectory by Method II (dashed) and ground truth (solid)	69
Figure 5.8. Method III position estimation (dashed) and ground truth (solid) vs. time	72
Figure 5.9. Estimated angle (dashed) and the ground truth (solid) vs. time with Method III	73
Figure 5.10. Estimated trajectory by Method III (dashed) and ground truth (solid)	73
Figure 6.1. The arc walk curvature radius decision visualization	75
Figure 6.2. SURALP walking experiment with the yaw controller turned off	76
Figure 6.3. Inverse of the arc walk radius decided by the controller	77
Figure 6.4. SURALP walking experiment with the yaw controller turned on.	78

## LIST OF TABLES

	Page
Table 3.1. Length and weight information of links	20
Table 3.2. Joint actuator specifications	21
Table 3.3. Sensor system of SURALP	21
Table 5.1 Pseudo-code for solving constrained NLLS using penalty method	60
Table A.1. Pseudo-code for Levenberg-Marquardt algorithm	90
Table B.1. RANSAC pseudo-code	91

# Chapter 1

## 1. INTRODUCTION

The worldwide research interest towards humanoid robots has been growing for the past few decades. As the computational power and memory gets cheaper, scientists and engineers are encouraged to venture further possibilities to create more human-like machines. The humanoids are expected to assist and/or replace humans in many fields including, but not limited to, heavy-duty professions (e.g. mining, construction), service and healthcare industries, or even in our homes. Hence, mimicking the human form and functions is possibly advantageous as humans design their surroundings in favor of their own capabilities. But creating a human analogue from inanimate objects is an ambitious goal, especially because the human brain is a miraculous “machine”. Even if the computational power was not an issue, there is still a lot of ground to cover before a robot can match the versatility of a human-being.

Achieving complete (or at least sufficient with respect to their purpose) autonomy for robots should be the first stepping stone towards the ultimate goal. They should be able to make the necessary decisions and adjustments themselves to be able to help humans. A robot can be said to be autonomous if it can perform its desired tasks in an unstructured and unsupervised scenario. This criterion can be satisfied if necessary thresholds are passed in research fields like artificial intelligence and cognitive robotics. But applications in these fields also need reliable information to work on; hence progress in lower level data acquisition and information extraction tools is crucial.

Vision sensors, namely cameras, generate huge amounts of data depicting the environment. If these data are processed and converted to bits of meaningful information, higher level tasks can then be pursued. These sensors should be at utmost importance for humanoids. Any human environment, or a scenario including a human-robot interaction, inherently contains important visual cues as the human brain and anatomy is evolved to favor the visual data over other senses. With that being said, robust and generic vision applications are very rare for robots, as the sensor data

depends heavily on environmental conditions and the computational costs of machine vision algorithms are still sometimes out of reach. Nevertheless, a large amount of research effort worldwide is directed at these challenges and successful examples exist in applications including object recognition, localization and manipulation, human identification, gesture recognition and self-localization.

Self-localization is a problem that is encountered by every autonomous mobile robot that is trying to accomplish a given task that requires visiting an arbitrary number of waypoints. As a couple of application examples; a humanoid robot taking care of an ill person would have to go find the necessary medicine and bring it back to him/her, or a robot serving as a waiter in a restaurant would have to patrol between tables and occasionally go to the kitchen. Although these tasks seem trivial to a human-being, they are highly challenging problems for robots and gather attention from numerous researchers around the world.

Estimating the change in position and orientation over time from a moving sensor set is generally called *odometry*. In a mobile robot application, these sensor sets often include inertial measurement units (accelerometers and gyroscopes), cameras, laser rangefinders, LIDARs and sonar sensors. A more specific method relying solely (or heavily) on the data from the cameras is called *visual odometry*. Estimating the trajectory of a set of cameras in space (detached from robotics context) is often called *egomotion estimation* or *camera pose estimation*.

In this work, a visual odometry algorithm using a stereo camera pair is developed and applied to the humanoid robot SURALP to correct for trajectory deviations during a walk in a static scene. The visual odometry algorithm includes feature detection and matching between the stereo pair, 3D reconstruction, feature tracking via optical flow and nonlinear least squares techniques to estimate the absolute position and orientation of the camera set relative to its initial pose. The development stage needed a ground truth trajectory to compare; hence a benchmarking video sequence recorded on a wheeled robot is utilized. The comparison between the algorithm results against the ground truth trajectory is presented.

The organization of this thesis is as follows.

The next chapter contains a literature survey on egomotion estimation in general and its applications in robotics. Several examples of humanoid robots with self-localization methods are given.

Chapter 3 describes the experimental robot SURALP. The structure and hardware, walking reference generation and basic walking control routines are briefly presented.

3D reconstruction and stereo vision basics are introduced in Chapter 4. Pinhole camera model is presented to show the relation between 3D points and camera pixel coordinates of the related features in the images and the necessary information needed to reconstruct a 3D point from its stereo correspondences. Calibration, undistortion and rectification are discussed as the necessary steps before the actual 3D reconstruction can begin. Finally, last part of this chapter explains how features are found and matched in stereo images and converted to 3D world coordinates.

Chapter 5 contains an outline to the implemented visual odometry algorithm. Tracking of 3D points and the role of optical flow methods are presented. The camera pose estimation parameterized by a unit quaternion is defined as a nonlinear least squares problem with a nonlinear constraint and a few possible solutions are evaluated. The necessity of robust estimation techniques and an estimation refinement step including a probabilistic filtering (namely an Extended Kalman Filter) or a bundle adjustment process, are discussed. Finally, a comparison between possible solutions is carried out by assessing them using the benchmarking video sequence and its ground truth trajectory.

The implementation of the visual odometry algorithm on the humanoid robot SURALP and the obtained results are presented in the Chapter 6.

The last chapter summarizes the work carried out. Comments on the obtained results, conclusion remarks and directions for the future work are given in this part.

## **Chapter 2**

### **2. A SURVEY ON EGOMOTION ESTIMATION AND ITS APPLICATIONS IN ROBOTICS**

This chapter presents a literature survey on the egomotion estimation and its application to robotics to establish the ground of this thesis among related research areas. In the first section, several methods of egomotion estimation and the application areas in computer vision systems in general are pointed out. The following sections sample the mobile robotics and visual odometry research area and finally, going from general to specific, the state-of-the-art humanoid robots with self-localization abilities are identified.

#### **2.1. Methods and Application Areas**

The egomotion estimation is defined as the process of calculating the observer's motion using the visual data. Several solutions to this problem exist in the literature and there are multiple categories that divide them: motion estimation level (linear and angular velocities vs. the absolute position and orientation of the camera), camera setup (single vs. multiple views) and the motion of the scene (static vs. dynamic).

Early works in this field deal with relating image velocities to linear and angular velocities of a single camera. They commonly make two main assumptions; the apparent motion of the scene is generated by the camera itself and instantaneous velocities of the projected points can be observed.

One of the pioneer works is given in [1] where the authors decouple the observed motion from scene depth in order to create a bilinear constraint on linear and angular velocity of the camera. Their method is one of the many that relies on optical flow estimation for representing image velocities. Soon after, this work is followed by a series of methods [2-7] that are built on the estimation of the focus of expansion (FOE).

The observation is when the observer is purely translating, the projected motion vectors of the points in the scene converge to (or diverge from) a single point (FOE), and in the case of a complex motion FOE can be used to decouple the estimation of rotation and translation. Finally, [8] presents a notable work based on motion parallax and decoupling the translation from rotation from image deformations. A comparison in performance of some of these methods is given in [9].

Another approach is to use the epipolar constraint between the two views taken by a single calibrated camera at different time steps. These methods result in discrete-time estimations of the motion parameters (translational and rotational displacement of the camera between frames) and some examples are given in [10-13]. They are closely related to structure-from-motion (SFM) applications where the 3D structure of the scene and the motion parameters of the camera are simultaneously estimated. The SFM applications generally are finalized by an offline batch processing with bundle adjustment [14] to refine the resulting 3D reconstruction and the camera trajectory. The bundle adjustment is adopted by robotics community to be applied in real-time (which will be discussed in the next section) and also implemented in this thesis' work.

It is important to note that the methods introduced so far suffer from the ambiguity in translation and scale created by the projective geometry, since they do not benefit from an auxiliary view of the scene. In this thesis work, a stereo camera pair is utilized to overcome this problem; hence these methods are not considered.

A family of algorithms known as Perspective-N-Points (PnP) exists to solve for the absolute position and orientation of the camera given the projections of  $n$  known scene points to the image plane. Linear solutions exist in the literature for P3P [14] and P4P [15], but above 5 points the solution can be obtained using Direct Linear Transform (DLT) [16]. An important note on these algorithms is P3P solutions contain an ambiguity and result in 4 possible solutions, whereas P4P algorithms (and above) have a unique solution as long as the points are non-coplanar. Another major concern is computational efficiency as these algorithms are generally iterative. An efficient version (EPnP) is introduced in [17] to match the needs of a real-time system. Although these algorithms relate scene points to a single calibrated camera, a stereo system could be used to initially estimate the 3D scene structure, and solve for camera poses by tracking these features over time. The authors of [18] provide a visual odometry pipeline utilizing these algorithms and a performance comparison between them.



Up to this point, all of the above mentioned algorithms dealt with projections of the scene points onto the image plane, whereas with a multiple-view camera rig estimations of the 3D scene points are available at any time. Therefore, finding the transformation between the estimated 3D points in the camera frame and their known world frame coordinates is possible. This can be seen as a data alignment problem and a solution is described in [19]. This is the closest solution to the one implemented in this thesis work where the transformation between 3D points are estimated at each step. It also provides the insight that the translation and rotation between the point clouds can be decoupled by moving the centroids of them to the origin.

Another notable application, along with SFM and robotics, is the augmented reality (AR) systems. AR systems aim at inserting computer generated images (generally of 3D graphic models) onto real video sequences. Although they often adopt a marker with a known geometry placed into the scene to estimate the camera pose, there is a growing research interest in marker-less AR. These marker-less AR applications also benefit from egomotion estimation techniques and solutions provided are closely related to the ones used in robotics research. An example of such a system is given in [20] dealing with partially known dynamic scenes.

## **2.2. Literature Survey on Visual Odometry in Mobile Robotics**

Odometry is a term used in navigation and robotics for estimating the actual position and orientation of a moving vehicle/robot using the onboard sensor data. The simplest odometry is achieved by integrating the estimated velocity of the vehicle over time and this method is called *dead reckoning*.

For a wheeled robot, an example dead reckoning method could be measuring the wheel speeds and converting them to linear speed which is then integrated over time. But this method has an obvious flaw; we are assuming that the exact speeds of the wheels can be measured, and that the velocity of the ground is the same as the wheel's tangential velocity at the point of contact, which means no slipping. These assumptions very rarely (if not never) hold true. Hence we integrate the errors along with the estimations and the estimated position *drifts* from the actual one very rapidly. This drifting issue exists for any kind of odometry actually, so the performances of the methods are evaluated as the percentage of drift in a distance traveled.

Motion in 3D world has 6 degrees-of-freedom (DOF), so one needs to specify 3 position and 3 orientation components to fully define a body in space. Hence, full visual odometry has to estimate all 6 DOF to allow the robot to move freely. There are however, some works in the literature that benefit from a simplification of the motion parameters. These are generally wheeled or tracked robots, or automated cars which are assumed to move strictly on the ground plane. Examples of two wheeled robots with 3DOF (planar) visual odometry systems are given in [21], [22] and [23]. These robots adopt a single camera attached to their body watching ground movement and use optical flow fields gathered from the ground texture to estimate their motion. Very similar approaches are presented in [24] and [25] applied on automated cars. Although these approaches should yield more robust results than a full 6DOF estimation (because there are less parameters to estimate), their use is limited to these occasions and even then, the imperfections on the ground (level differences, bumps) could affect the outcome significantly.

Another popular approach is Monte Carlo Localization (MCL), and it is used to localize the robot given the map of its environment. This method is an application of the particle filters in robot localization and it is based on randomly sampling the space of all possible configurations, and updating the probability of correctness of each sample with coming sensor data. In [26], an application of this method on a shopping companion wheeled robot equipped with an omnidirectional camera is described. Their approach requires the robot to be led through the environment once as a training session to build the necessary map. Another such example is given in [27], where the robot is a quadruped competing in the RoboCup football challenge. The environment is a mini football field and it is encoded as a map into the robot. Although the method is extendable to localize the robot in 6DOF, sampling the possible configurations in the 6-dimensional space is costly (curse of dimensionality); hence these examples also estimate the 3DOF position and orientation of the robot on the ground plane.

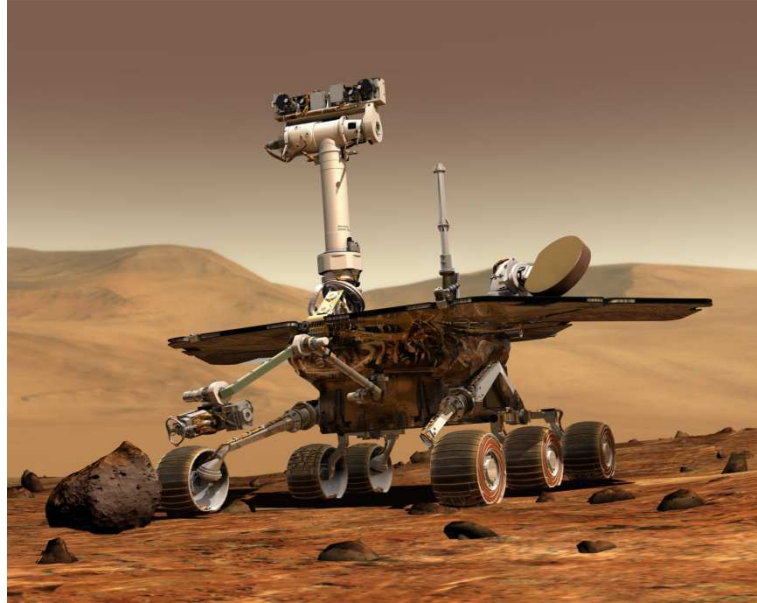
Apart from the previously discussed methods, the visual odometry techniques adopted by the robotics community share a large amount of common steps. They are based on reconstructing the 3D scene (dense or sparse) by triangulation and tracking the obtained 3D points over time to solve for the absolute position and orientation of the robot in the world coordinate frame. Random Sample Consensus (RANSAC) is also very popular for dealing with outliers created by false matches and tracking errors. Few of many examples of such systems in wheeled locomotion are given in [28] and [29].

Flying robots like quadrotors are generally designed for outdoor applications and rely on GPS data for self-localization. There is however a respectable amount of research aimed to apply visual odometry methods on these machines for either getting a finer estimation of the robot position, or to be used in GPS-denied indoor applications. Authors of [30] applied visual odometry methods on a quadrotor flying in an indoor environment and their results show the drastic drift in estimation for such rapidly moving machines even with a refining bundle adjustment process.

The bundle adjustment is a powerful technique and as mentioned before, it is widely used in SFM applications to refine the final 3D reconstruction and camera trajectory simultaneously. Although the systems of equations tend to get really large even for two consecutive frames, recently there has been a significant progress towards achieving online bundle adjustment. So called Sparse Bundle Adjustment (SBA) [31] method is based on abusing the sparse structure of the system created by the lack of interaction between the 3D scene points and solving the problem efficiently. The authors of [32] implemented SBA on a mobile robot with a sliding window approach to minimize the drift. Their sliding window resembles the batch operation of SFM applications using only a small number of consequent images. One key remark on their algorithm is that they do not estimate the egomotion using images before the bundle adjustment, they use the odometry estimation acquired from the other sensors of the robot (dead reckoning result) as an initial guess for SBA. Their results show that this approach is feasible and satisfactory.

Recently, there has been a very successful and renowned example of autonomous robots benefiting from visual odometry techniques: the NASA Mars Exploration Rovers *Spirit* and *Opportunity*. These robots spent 2 years on the surface of Mars, moving autonomously (as the communication delay does not allow human supervision). Their method is described in [33] and [34], and the evaluation of the system performance on Mars is published in [35]. They have adopted an Expectation Maximization (EM) method to calculate the robot trajectory in 6DOF, by defining a normally distributed error over the reconstructed 3D points. This error modeling technique is introduced in [36] and forms a basis for the necessary robust estimation of visual odometry over very long distances. Although the resulting drift in estimated position is minimal, it still exists as a threat against long distance navigation. Hence, they have relied on a sun sensor to crudely localize the absolute position of the robot on

the planet surface and prevented the drift from growing unboundedly [37]. A computer generated image of a NASA Mars Exploration Rover is shown in the Figure 2.1.



**Figure 2.1.** A computer generated image of a NASA Mars Exploration Rover

The drift in visual odometry is counter-intuitive in a sense that the 3D measurements seem to be coming from a static world and localization among them should be trivial. One has to take into account that the measurements of the 3D points in the world rely on the exact knowledge of the motion parameters of the camera set. This creates a “chicken or egg” problem because estimation of the camera pose also depends on the 3D scene information. That is to say, while estimating the 3D point locations and the egomotion of the camera set on the fly, the incoming 3D point coordinates contain errors caused by prior inaccurate camera motion parameters. Regarding this issue, there is a great deal of research effort spent to create Simultaneous Localization and Mapping (SLAM) systems.

SLAM researchers rightfully claim that localization and mapping form a coupled problem and cannot be solved separately. The provided solutions often contain information from various sensors including inertial measurement units (IMU), laser rangefinders, LIDARs and sonar sensors along with multiple cameras (color and grayscale). Hence SLAM solutions can be seen as higher level probabilistic algorithms

utilizing visual odometry estimations. Equivalently, the visual odometry techniques can be seen as local estimators and SLAM algorithms as global ones. In this thesis work, SLAM methods are not explored, but they are worth mentioning to establish the current research trends.

SLAM is first introduced in 1986 by the authors of [38] and [39], and actually spent a rather quite time during the 1990s with few researchers working on a solution because of the necessary computational power. In the early 2000s, advances in computer hardware enabled SLAM algorithms to be solved in acceptable time steps and the research interest grew exponentially.

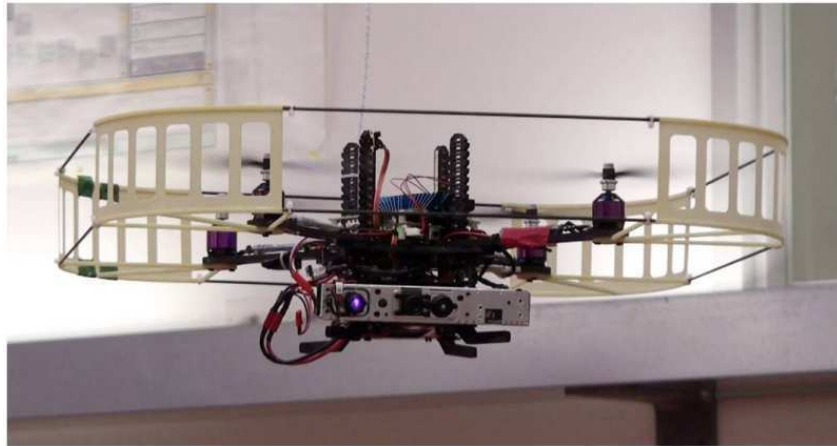
The key concept of SLAM is the so-called “loop closures”. These loop closures are the points on which the robot comes to a previously visited waypoint and recognizes the features from a memorized, permanent map. These maps, which typically consist of thousands of geometric features, are refined on these loop closure points to prevent unbounded growth of drift. The central problem of visual SLAM arises immediately as the recognition of 3D features from different points of view, which requires appropriate descriptors for each geometric feature. Because of the required computational power to process large maps, SLAM problem is often reduced to refining a bird’s eye view map of the environment and the robot’s motion parameters are expressed in 3DOF planar coordinates. Nevertheless, there are applications in the literature which work with full 3D maps. The refinement process is generally carried out with probabilistic methods like particle filters or Extended Kalman Filters (EKF). EKF-SLAM is especially popular because it creates a framework to merge data coming from various sensors along with modeling the robot dynamics.

Very few examples of the existing SLAM research and applications are given in this survey, because trying to cover the SLAM literature exhaustively would require a significant amount of time and effort and actually is out of focus of this thesis.

**Wheeled or tracked robots** are the ones that attract most SLAM researchers, because of the simplicity of locomotion. These robots allow engineers to make simplifying assumptions that prove to be valuable for online SLAM applications. An application of EKF-SLAM with a wheeled robot moving in an outdoor environment is given in [40]. They utilize data from vision sensors along with GPS and dead reckoning to find a statistically optimal map with EKF. The authors of [41] implemented a Rao-Blackwellised Particle Filter (RBPF) based SLAM on a wheeled robot in an indoor environment. The results of their work clearly show the difference between visual

odometry estimation and SLAM correction. These robots work with planar maps as they assume the robot trajectory (both in position and orientation) does not leave the ground plane.

**Flying robots** (namely quadrotors) are also considered as SLAM application candidates. An EKF-SLAM system on a quadrotor moving in GPS-denied indoor environments is presented in [42]. The existing sensors of the described system are a stereo camera rig, a laser rangefinder and an IMU. The EKF returns a state estimate in 4DOF (3 translation and yaw orientation) combining the sensor data. This is a similar simplification to a wheeled robot moving on the ground plane, only the plane that quadrotor is moving is assumed to be parallel to the ground rather than coincident. This assumption is achievable by their adopted feedback loop to correct for roll and pitch angles of the robot measured by the IMU. A very recent and notable SLAM example with a quadrotor micro air vehicle (MAV) is given in [43]. The presented system works with a single RGB-D (color and depth) camera to create a dense 3D model of the environment. This work shows the impact of advances in sensor technology on higher-level tasks. Their resulting 6DOF trajectory estimation and the environment map are impressive. A photograph of the MAV is shown in Figure 2.2.



**Figure 2.2.** A quadrotor MAV with and RGB-D camera

**Swimming robot** research in general is smaller than the other branches, so their SLAM applications are even rarer. The SLAM problem of swimming robots does not allow any simplifying assumption as the underwater world has inherently 6DOF. The

dynamic nature of the medium and the sensor behavior underwater add up to the challenge. Pioneer works on this problem are very recent (dating back to 2004) and are presented in [44-48]. The amphibious robot from [48] is shown in Figure 2.3.



**Figure 2.3.** An amphibious robot using stereo cameras

**Legged robots** subdivide into bipeds, quadrupeds and hexapods (with few exceptions). Visual odometry and SLAM applications in bipeds are analyzed in the next section. In the case of quadrupeds and hexapods, there is a surprisingly small amount of research aimed towards the SLAM problem. This lack of research interest could be caused by the purpose of such multi-legged structures. The multi-legged locomotion is inherently more stable than the bipedal one; hence the research areas of these types of robots are mainly shifted to stable walking and path planning in outdoor rough terrains. The outdoor scene is proven to cause a great amount of challenge against visual odometry techniques and the added erratic motion of the robot walking on rough terrain is a deal-breaker for vision systems.

An example work on quadrupeds aiming at modeling the rocky terrain while localizing the robot is given in [49]. They have mounted a stereo camera rig on a commercially available quadruped known as LittleDog to gather 3D points from the terrain. Their work differs from traditional SLAM approaches, because they use point cloud matching via Iterative Closest Point (ICP) algorithm to build a consistent 3D map of the rocky terrain and localize the robot. Although their results are very good, this approach is very computationally costly and not suitable for real-time requirements of

dynamic walking, so the described scenario requires the robot to move in discrete steps. The LittleDog and the stereo camera setup depicted in this publication are shown in Figure 2.4.



**Figure 2.4.** LittleDog and the stereo camera setup [49]

A very recent work on hexapods and SLAM is presented in [50]. A single camera is mounted on the hexapod and the visual data is combined with the reference trajectory to calculate the actual trajectory of the robot. Their results are not very clear, but they provide very useful insights. They claim that for such small robotic systems, the onboard computational power is always relatively low; so the proposed methods should be able to keep track of the robot even at very low frame rates. Another important issue pointed out is the effect of abrupt changes in motion due to the legged locomotion on the visual SLAM.

### **2.3. Literature Survey on the Self-Localization Problem in Humanoid Robots**

Studies show that humans rely on visual data to localize themselves in their environment. In a recent study [51], experiments with a number of human subjects are carried out to see if they can reach a target distance while blindfolded. The results show that even the subjects tried to repeat a trajectory they had walked minutes ago while they were seeing; the drift in direction and distance traveled was significant and

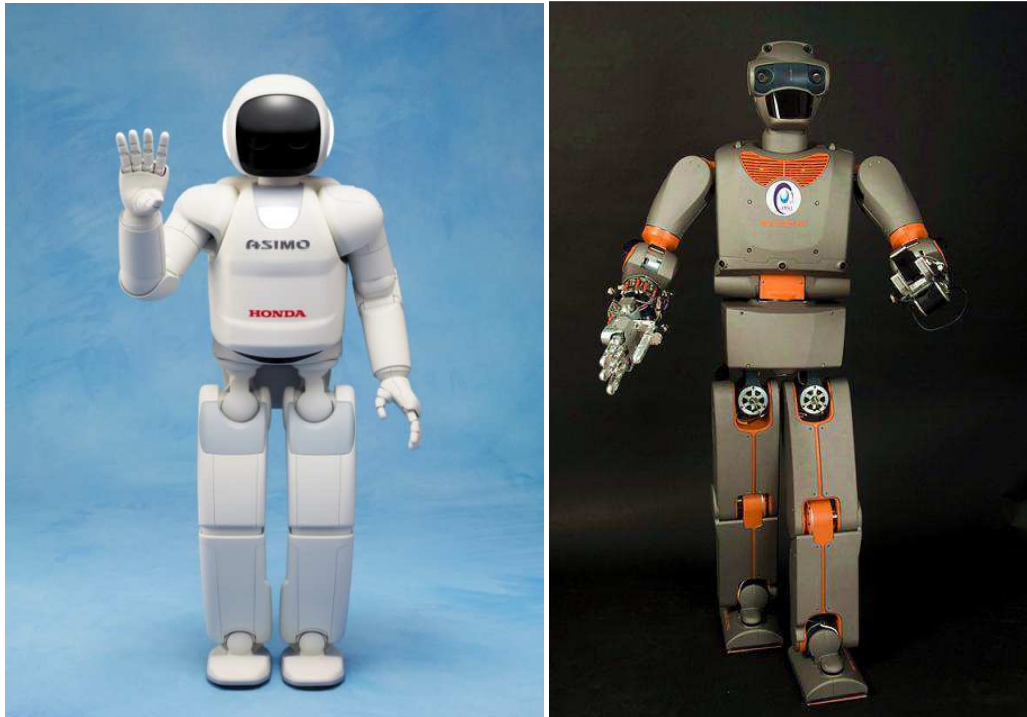


proportional to the length of the test. This result encourages the use of vision sensors in humanoid robots to generate feedback from the environment.

An extensive study on human perception of visual motion is given in [52]. The study claims there is evidence that the humans perceive movement from optical flow and use that information along with depth perception generated by binocular disparity to sense their egomotion as well as the motion of the other dynamic objects. These observations point out the similarities between the human perception of egomotion and the methods applied on a humanoid robot in this thesis.

In the research field of humanoid robotics, there are a number of bipedal walking machines, but few of them have full-body structure needed to perform high-level tasks. The humanoids with self-localization capabilities are even rarer and form an elite group among the other ones.

The self-localization problem of humanoids is very similar to the other types of mobile robots; but it is generally more challenging as the vibrations and abrupt motions caused by bipedal locomotion are significant. These disturbances highly affect the data gathered by any onboard sensor and threaten the health of any kind of odometry estimation. The magnitude of these disturbances and their effect on low-end vision sensors are clearly shown in [53]. In the implementation of this thesis work, the motion blur effect is not filtered but minimized by using cameras with high image acquisition frame rates and low exposure times. Even though the images are clean, estimating the motion parameters going under such high frequency changes is a challenging work.



**Figure 2.5.** Two commercial humanoid robots ASIMO and REEM-B

As mentioned before, there are only a handful of humanoid robots capable of localizing themselves in their environment. One of the most popular ones, ASIMO [54] is known to be able to perform tasks that require self-localization, but there are no publications in the literature explaining the details.

REEM-B [55] is another commercial humanoid robot that has the ability to build maps of indoor environments and localize itself using these maps. REEM-B has a stereo camera rig that can locate objects in 3D, but it is only used as auxiliary information for self-localization task. It uses two laser sensors located on its feet and walking trajectory reference to build 2D local maps, and DP-SLAM algorithm to merge the local maps into a global one. Once the map building process is finalized, 3DOF self-localization is performed using MCL. If the degree of confidence in localization drops under a certain level, visual landmarks recorded while building the map are compared to the camera data. Photographs of ASIMO and REEM-B are shown in Figure 2.5.

HRP family [56] of humanoid robots includes a few of the most advanced humanoid robots and they have been used by various research groups. A photo showing HRP-2, HRP-3 and HRP-4 is given in Figure 2.6.



**Figure 2.6.** A photograph of HRP-2, HRP-3 and HRP-4 respectively

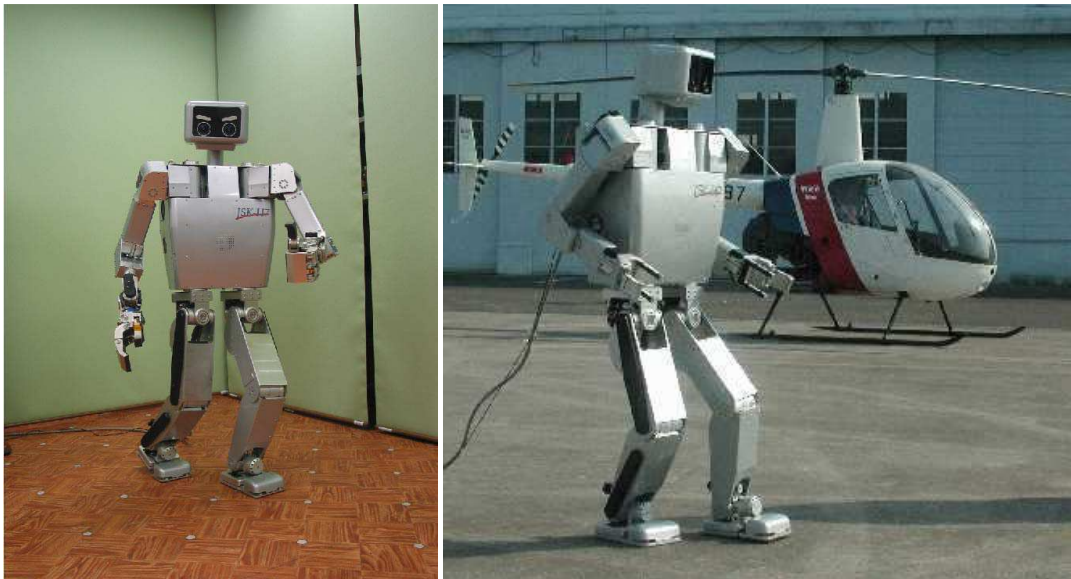
The authors of [57] and [58] present a visual EKF-SLAM method using a single camera applied on HRP-2. They point out that the necessary lateral motion of the body for a stable walking pattern can also be used to create a parallax effect between two frames taken with a single camera. Whereas the same robot is used in [59] and RBPF-SLAM method is applied on a stereo vision system to generate a grid-based map of the environment.

Another humanoid that has been the test bed of various successful self-localization algorithms is the H7 [60]. Figure 2.7 shows photographs of H7.

MCL approach has been implemented in a way very similar to that of REEM-B in [61]. The method uses stereo vision to reconstruct 3D points in the scene, and project the scene points to the ground plane to be used as a map for 3DOF MCL estimation.

The state of the art for visual self-localization in humanoid robots is accepted to be the work described in [62] and [63]. Their method uses stereo vision and visual odometry to create 3D local maps to be used in an EKF-SLAM context to estimate the trajectory of the robot in 6DOF. The visual odometry estimation method described in these publications form the basis of the work implemented in this thesis. So, their methods will be explained in more detail.

The system described in both publications starts by finding the dense depth map of the scene from the stereo cameras. Because dense depth maps often contain a high level of noise, they have provided a preprocessing technique to filter out the outliers in the depth data. The dense depth map is not used for visual odometry estimation, but to build a dense 3D map of the environment. The visual odometry calculation is carried out by tracking point features in the scene. They have used a Kanade-Lucas-Tomasi (KLT) feature tracker which is a fairly standard method based on finding features with Shi-Tomasi corner detector [64] and tracking them over time using Lucas-Kanade optical flow [65]. The estimation of the transformation between the 3D camera frame coordinates of the tracked features and their positions in the 3D world frame map is solved as a registration problem. Because of the errors in triangulation and tracking, they have devised a robust estimation method using RANSAC with a rigid body motion constraint. The difference in approaches described in both publications is that the method presented in [62] estimates the rotation and translation of the camera set simultaneously, while in [63], these motion parameters are decoupled and estimated separately. Further details on both of these approaches will be given in the Chapter 5.



**Figure 2.7.** The humanoid robot H7

The next chapter introduces the experimental humanoid robot SURALP and gives a brief overview on hardware and stable walking control methods utilized by the robot.

## Chapter 3

### 3. THE EXPERIMENTAL HUMANOID SURALP

SURALP is a human-sized full-body robot designed and constructed at Sabanci University Robotics Laboratory. This robot is introduced in this chapter with a special attention to its reference generation method for changing the direction of the walk on the fly.

#### 3.1. Hardware

A picture and dimensional drawings of the robot are shown in Figure 3.1. It is designed in human proportions with 29 DOF, including 6-DOF legs, 6-DOF arms, 1 DOF hands, a 2-DOF neck and a 1-DOF waist [66]. The kinematic arrangement is presented in Figure 3.2. The weight of the robot is 114 kg. Various dimensions are tabulated in Table 3.1. DC motors are used as actuators. Motor drivers are in the trunk. Belt-pulley systems transmit the motor rotary motion to Harmonic Drive reduction gears (Table 3.2). The sensor system of SURALP includes encoders measuring the motor angular positions, six-axes force/torque sensors positioned at the ankles and wrists, a rate gyro, an inclinometer, and a linear accelerometer mounted at the robot torso. Two CCD cameras are mounted to the head of the robot for visual information. Table 3.3 shows the sensor working ranges, mounting locations and allocated communication channels.

The control hardware of SURALP consists of a modular dSpace digital signal processing system in a backpack configuration Figure 3.3. The controller cycle time employed is 1 milliseconds.

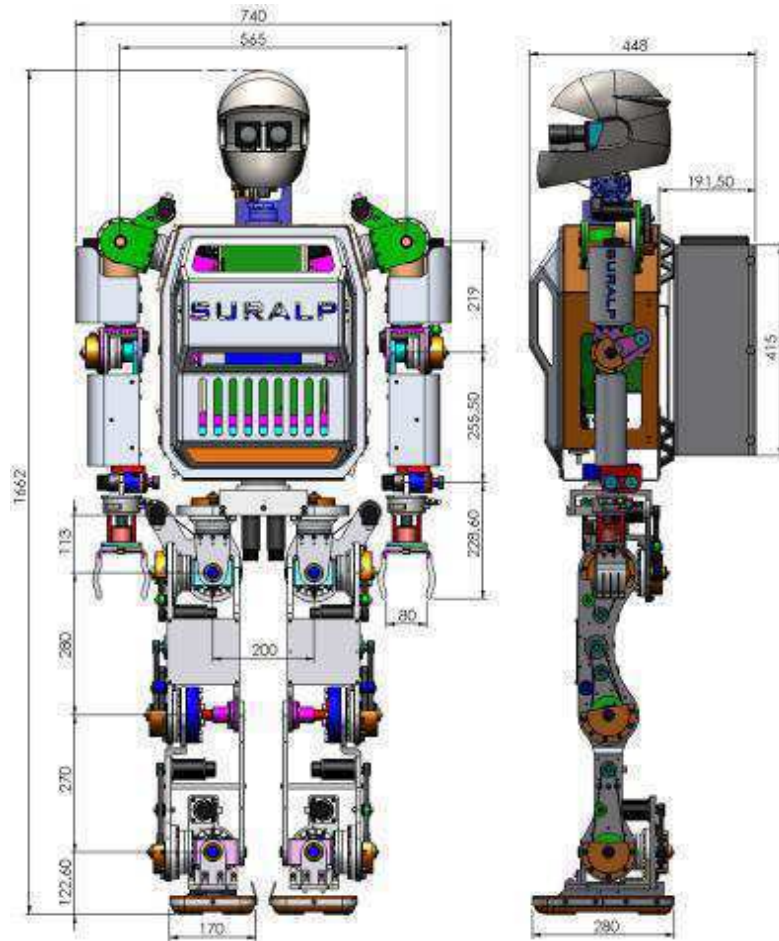
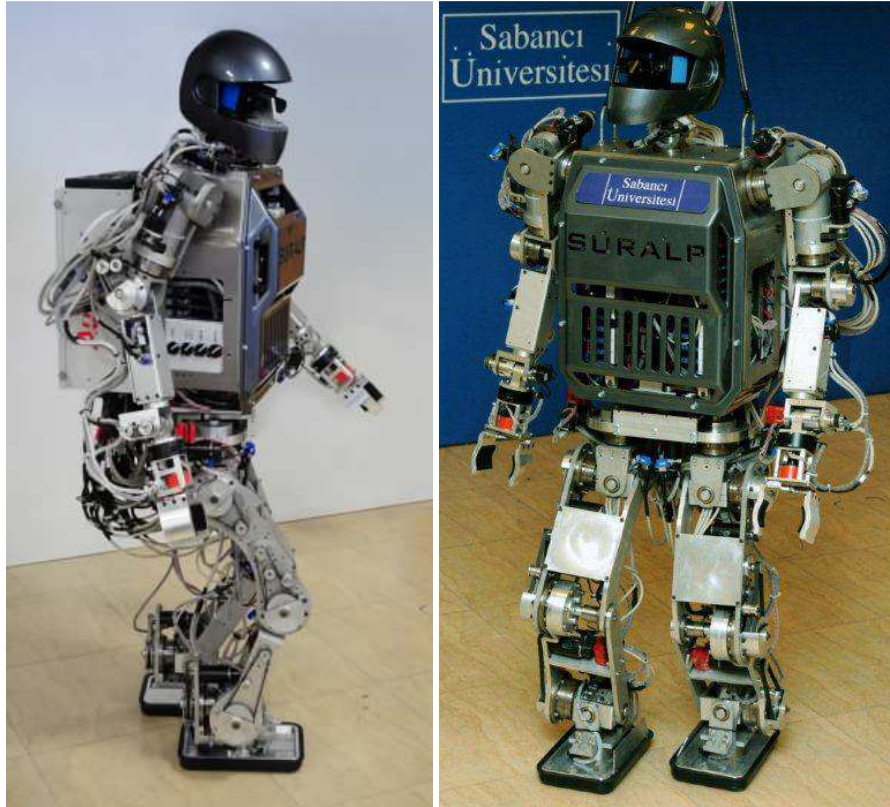
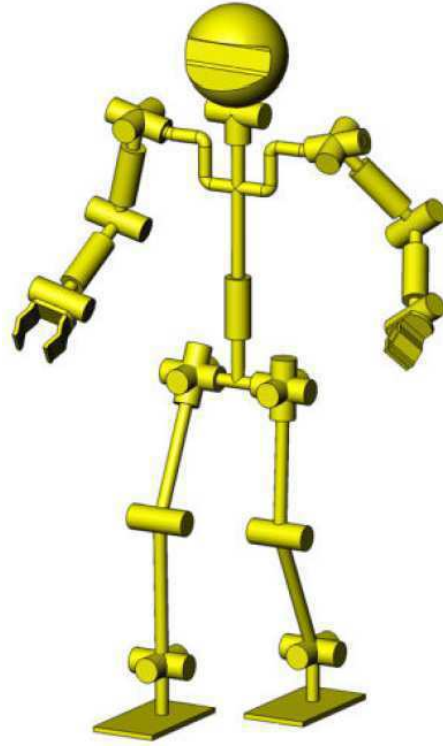


Figure 3.1. Humanoid robot SURALP, dimensions.



**Figure 3.2.** Kinematic arrangement of SURALP

**Table 3.1.** Length and weight information of links

Upper Leg Length	280mm
Lower Leg Length	270mm
Sole-Ankle Distance	124mm
Foot Dimensions	240mm x 150mm
Upper Arm Length	219mm
Lower Arm Length	255mm
Robot Weight	114 kg

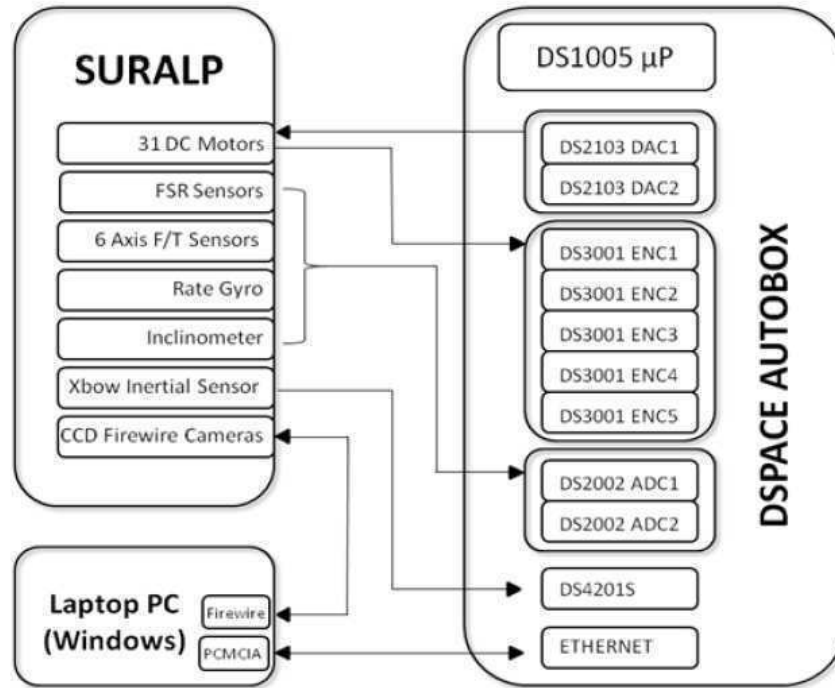
**Table 3.2.** Joint actuator specifications

<b>Joint</b>	<b>Motor Power</b>	<b>Pulley Ratio</b>	<b>HD Ratio</b>	<b>Joint Range</b>
Hip-Yaw	90W	3	120	-50 to 90 deg
Hip-Roll	150W	3	160	-31 to 23 deg
Hip-Pitch	150W	3	120	-128 to 43 deg
Knee 1-2	150W	3	160	-97 to 135 deg
Ankle-Pitch	150W	3	100	-115 to 23 deg
Ankle Roll	150W	3	120	-19 to 31 deg
Shoulder Roll 1	150W	2	160	-180 to 180 deg
Shoulder Pitch	150W	2	160	-23 to 135 deg
Shoulder Roll 2	90W	2	120	-180 to 180 deg
Elbow	150W	2	120	-49 to 110 deg
Wrist Roll	70W	1	74	-180 to 180 deg
Wrist Pitch	90W	1	100	-16 to 90 deg
Gripper	4W	1	689	0 to 80 mm
Neck Pan	90W	1	100	-180 to 180 deg
Neck Tilt	70W	2	100	-24 to 30 deg
Waist	150W	2	160	-40 to 40 deg

**Table 3.3.** Sensor system of SURALP

	<b>Sensor</b>	<b>Number of Channels</b>	<b>Range</b>
All joints	Incremental optic encoders	1 channel per joint	500 pulses/rev
Ankle	F/T sensor	6 channels per ankle	$\pm 660$ N (x, y-axes) $\pm 1980$ N (z-axis) $\pm 60$ Nm (all axes)
Torso	Accelerometer	3 channels	$\pm 2$ G
	Inclinometer	2 channels	$\pm 30$ deg
	Rate gyro	3 channels	$\pm 150$ deg/s
Wrist	F/T sensor	6 channels per wrist	$\pm 65$ N (x, y-axes) $\pm 200$ N (z-axis) $\pm 5$ Nm (all axes)
Head	CCD camera	2 channels	640x480 pixels (30 fps)





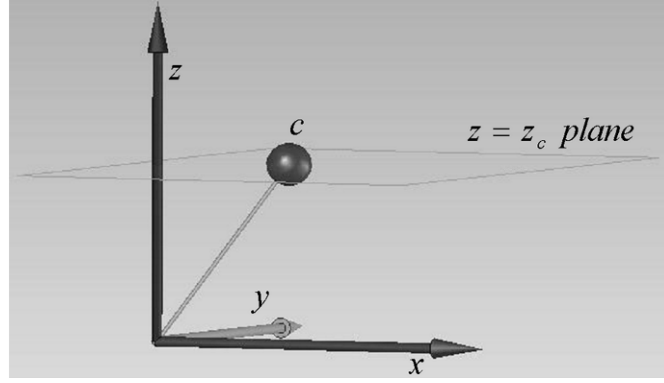
**Figure 3.3.** The hardware architecture of SURALP

### 3.2. Walking Reference Generation

A ZMP based walking reference generation technique is employed. Firstly, references for straight walk are considered. In order to change the direction of the walk, a modification which maps straight walk references on arc shaped paths is employed. The next two subsections discuss the straight walk references and the mapping for direction changes.

#### 3.2.1. Straight Walk

The Linear Inverted Pendulum Model (LIPM), with its simple structure, is suitable for reference generation purposes. A point mass is assigned to the robot center of mass (COM) and it represents the trunk of the robot. The point mass is linked to a stable contact point on the ground via a massless rod, which is idealized model of a supporting leg. With the assumption of a fixed height for the COM, a linear system which is decoupled in the  $x$  and  $y$  directions is obtained. The system described above is shown in Figure 3.4.  $c = (c_x \quad c_y \quad c_z)^T$  represents the position of the point mass.



**Figure 3.4.** The linear inverted pendulum model

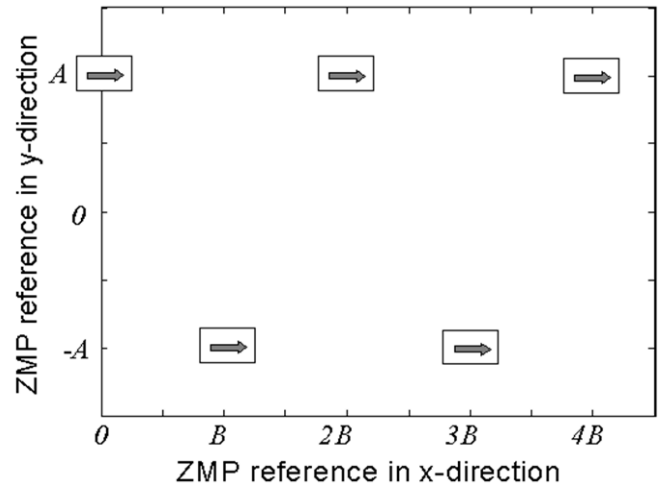
The ZMP is defined as the  $x$ - $y$  plane point on which no horizontal torque components exist. For the structure shown in Figure 3.4, the expressions for the ZMP coordinates  $p_x$  and  $p_y$  are [67]

$$p_x = c_x - (z_c/g)\ddot{c}_x, \quad (3.1)$$

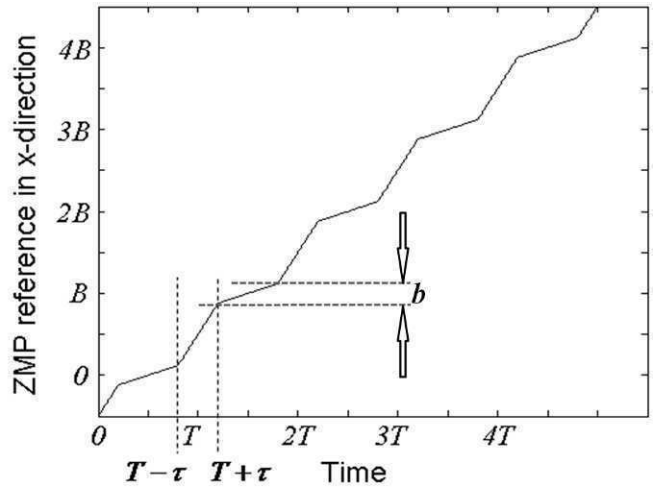
$$p_y = c_y - (z_c/g)\ddot{c}_y. \quad (3.2)$$

Here,  $z_c$  is the height of the plane where the motion of the point mass is constrained and  $g$  is the gravity constant. Suitable ZMP trajectories can be generated for reference generation purpose [68, 69]. As the stability constraint, the ZMP should always lie in the supporting polygon defined by the foot or feet touching the ground. Figure 3.5 shows a ZMP reference trajectory [70]. Firstly, support foot locations are chosen.  $A$  in the figure is the distance between the foot centers in the  $y$  direction,  $B$  is the step size and  $T$  is the half of the walking period in this figure.  $b$  defines the range of the ZMP motion under the sole. The double support phase is introduced by using the parameter  $\tau$ .

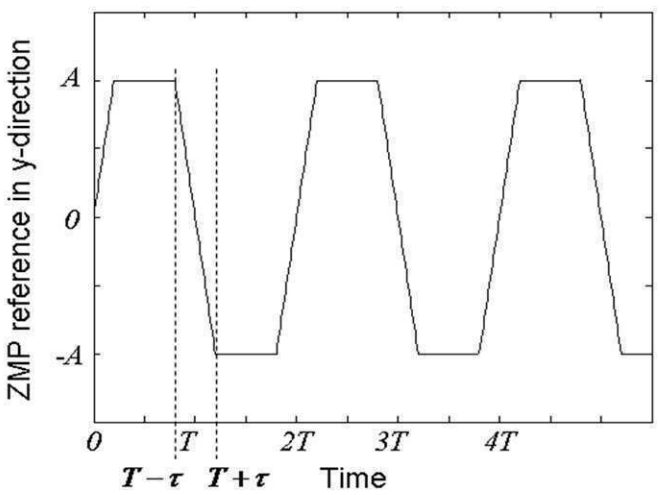
Having defined the curves, and hence the mathematical functions for the ZMP references, the next step is obtaining COM reference curves  $c_x^{ref}(t)$  and  $c_y^{ref}(t)$  from  $p_x^{ref}(t)$  and  $p_y^{ref}(t)$ , respectively. Fourier series' of the ZMP references  $p_x^{ref}(t)$  and  $p_y^{ref}(t)$  are used in this process to obtain Fourier series' for the  $x$  and  $y$  components of the COM trajectories.



a)



b)



c)

**Figure 3.5.** Forward moving ZMP references with pre-assigned double support phases.  
 a) Foot locations and forward moving ZMP in single support phases. b) The  $x$ -directional ZMP reference. c) The  $y$ -directional ZMP reference.

The obtained expression for the COM  $x$  directional component is [70]:

$$c_x^{ref} = \frac{B}{T} \left( t - \frac{T}{2} \right) + \frac{\alpha_0}{2} + \sum_{n=1}^{\infty} \alpha_k \cos\left(\frac{2\pi n t}{T}\right) + \beta_k \sin\left(\frac{2\pi n t}{T}\right) \quad (3.3)$$

The coefficients  $\alpha_0/2$ ,  $\alpha_k(1 + \pi^2 k^2 / \omega_n^2 T^2)$  (for  $k = 1, 2, 3, \dots$ ) of  $p_x^{ref}(t)$  are zero, and

$$\beta_k = \frac{\omega_n^2 T^2}{\pi^2 k^2 + \omega_n^2 T^2} \frac{2}{\pi k} \left\{ \sigma_1 \left[ -\tau \cos\left(\frac{2\pi k \tau}{T}\right) + \frac{T}{2\pi k} \sin\left(\frac{2\pi k \tau}{T}\right) \right] + \sigma_2 \left[ \tau \cos\left(\frac{2\pi k \tau}{T}\right) - \frac{T}{2} \left( \cos\left(\frac{2\pi k \tau}{T}\right) \right) - \frac{T}{2\pi k} \sin\left(\frac{2\pi k \tau}{T}\right) \right] \right\} \quad (3.4)$$

where  $\omega_n \equiv \sqrt{g/z_c}$ .

[70] finds the  $y$  directional component of the COM as

$$c_y^{ref}(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos\left(\frac{2\pi k t}{2T}\right) + b_k \sin\left(\frac{2\pi k t}{2T}\right). \quad (3.5)$$

$a_0/2$  and  $a_k(1 + (\pi^2 k^2) / (\omega_n^2 T^2))$  (for  $k = 1, 2, 3, \dots$ ) are zero. The remaining coefficients are obtained as

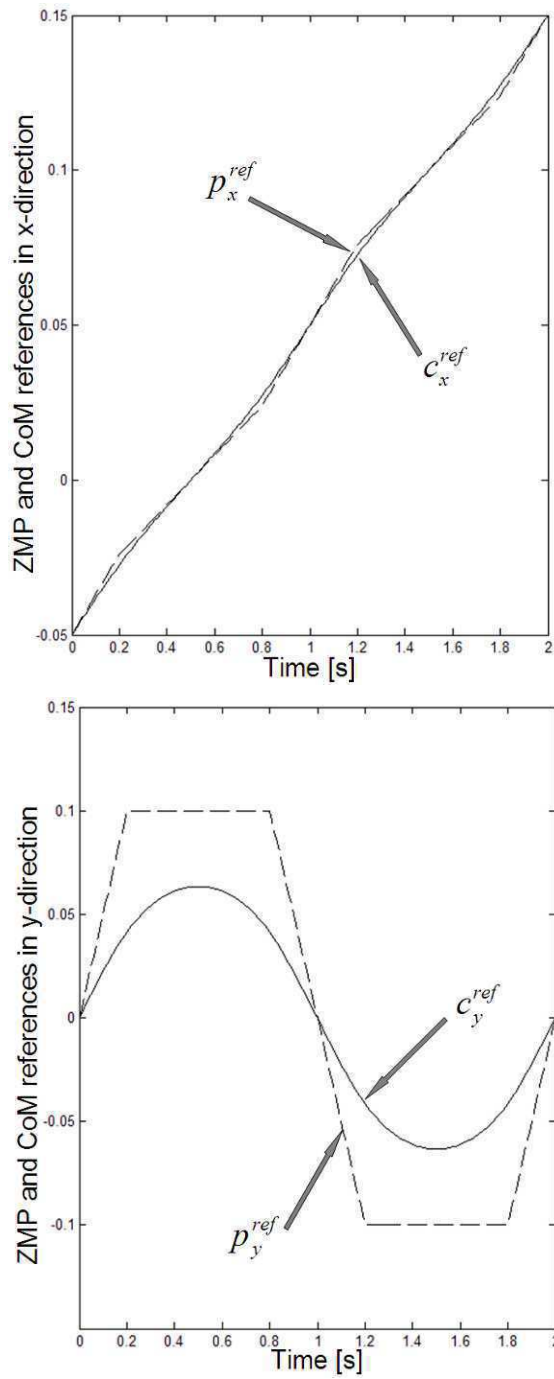
$$b_k = \begin{cases} \frac{\omega_n^2 T^2}{\omega_n^2 T^2 + \pi^2 k^2} \frac{2A}{\pi k} \left\{ \left[ \frac{2}{\tau} \left\langle \frac{T}{\pi k} \sin\left(\frac{\pi k \tau}{T}\right) - \tau \cos\left(\frac{\pi k \tau}{T}\right) \right\rangle \right] + \left[ \cos\left(\frac{\pi k \tau}{T}\right) - \cos\left(\frac{\pi k (T - \tau)}{T}\right) \right] \right\} & \text{if } k \text{ is odd} \\ 0 & \text{if } k \text{ is even} \end{cases} \quad (3.6)$$

The curves obtained for  $c_x^{ref}$  and  $c_y^{ref}$  are shown in Figure 3.6 together with the corresponding ZMP references defined in Figure 3.5. The infinite sums in (3.3) and (3.5) are approximated by finite sums of 24 terms. In Figure 3.6, the following parameter values are used:  $A = 0.1$  m,  $B = 0.1$  m,  $b = 0.04$  m,  $T = 1$  s and  $\tau = 0.2$  s.

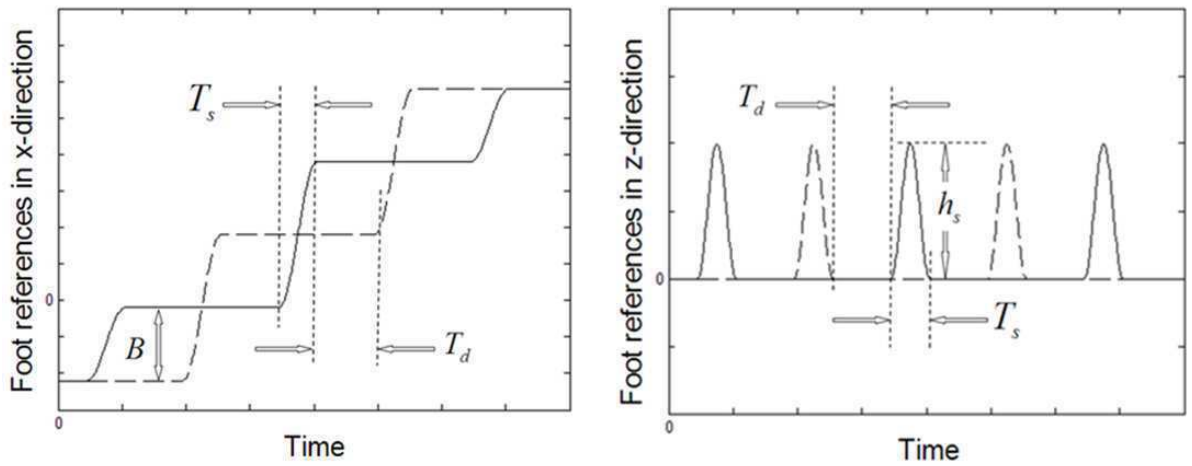
Foot position reference trajectories have to be designed too. Firstly foot placement timing and world frame foot position and orientation references are defined. Typical  $x$  and  $z$ -direction components of the foot trajectories used in this thesis are shown in Figure 3.7.  $T_d$  and  $T_s$  represent the double and single support periods, respectively.

( $T_d = 2\tau$ ,  $T_s = T - \tau$ .)  $B$  is the step size from Figure 3.5. The  $y$  direction trajectories are constant at  $-A$  and  $A$  for the right and left feet, respectively, where  $A$  is half of the foot to foot  $y$  direction distance also shown in Figure 3.5.  $h_s$  is the step height parameter. The foot orientation references are generated in such a way that the feet are parallel to the even ground.

The joint position references are obtained through inverse kinematics from COM and swing foot references defined in world frame coordinates. The process of reference generation is explained in detail in [70] and [71].



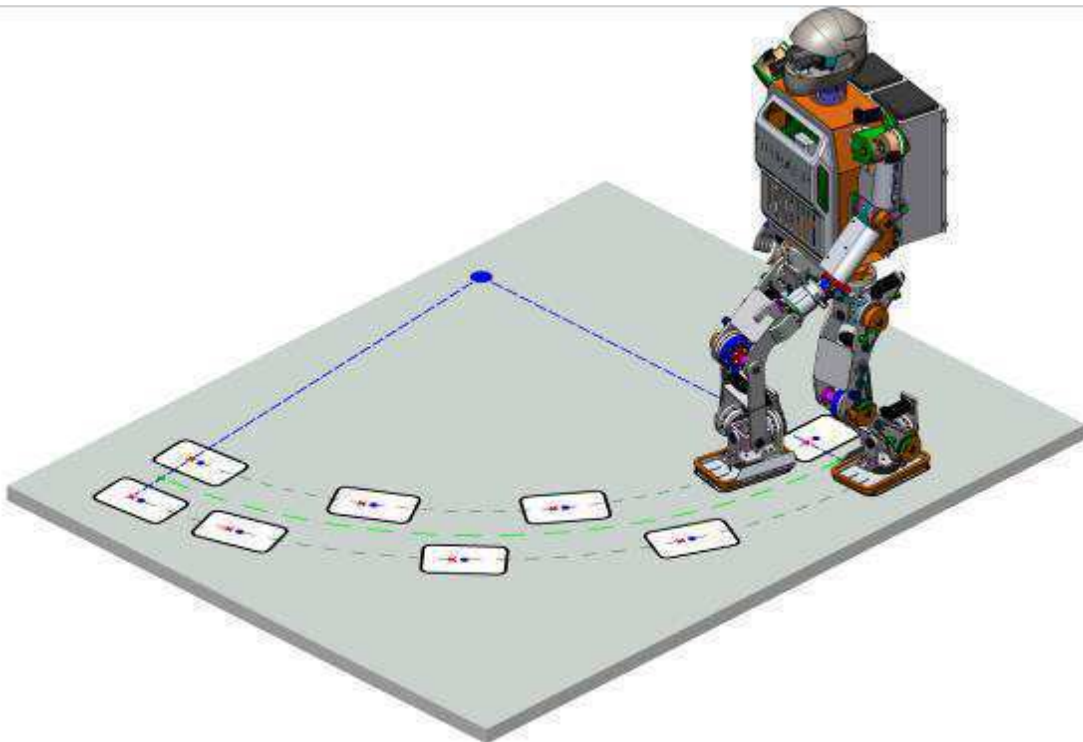
**Figure 3.6.**  $x$  and  $y$  -direction COM and ZMP references



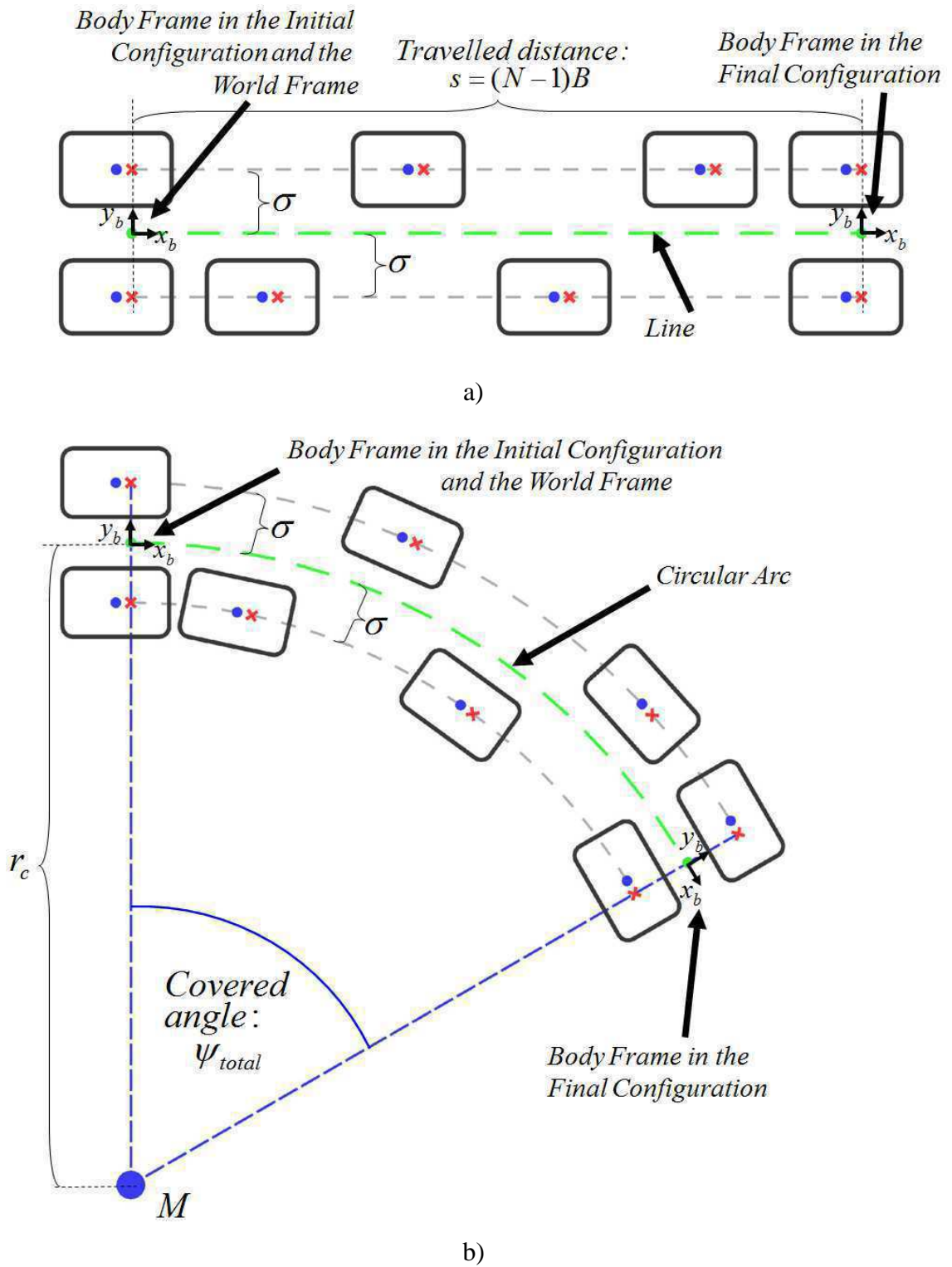
**Figure 3.7.**  $x$  and  $z$ -direction foot frame references in as expressed in the world frame. Solid curves belong to the right foot, dashed curves indicate left foot trajectories. (The  $y$ -directional foot references are not shown are equal to  $-A$  and  $A$ , respectively.)

### 3.2.2. Walk on Circular Arc Shaped Paths

The straight walk trajectory briefed in the previous section is exploited in this section to generate a walking reference which follows a circular arc as shown in Figure 3.8.



**Figure 3.8.** SURALP CAD model on a arc shaped walking trajectory.



**Figure 3.9.** Mapping of foot placement locations of a straight walk onto the foot placement locations of a circular arc-following walk. a) Straight walk b) Circular arc-following walk.



The method first considers the ground level line which connects the body frame origin projections on the ground at the beginning and at the end of the straight walk references (Figure 3.9.a). The right and left foot landing locations in the lateral direction are symmetric with respect to this line. The robot body and the feet are always kept parallel to it. The distance covered by the robot on this line can be computed as

$$s = (N - 1)B, \quad (3.7)$$

where  $N$  stands for the number of swings in the step sequence and  $B$  is the step size. This distance is mapped on a circular arc (Figure 3.9.b) to cover the angle

$$\psi_{total} = \frac{(N - 1)B}{r_c}. \quad (3.8)$$

Here,  $r_c$  is the radius of the turning circle. The body and feet are kept parallel to the arc. This, with (3.8), results in a turn of the robot to the right by the angle  $\psi_{total}$  in  $N$  steps. The smaller the radius  $r_c$  the acute is the turn. A very large radius corresponds to an almost straight walk.

In implementations with SURALP,  $r_c$  is a command variable interfaced to a human used through a joystick. Neutral joystick lateral position corresponds to a radius of 1000 m: This is a straight walk command. Other joystick lateral positions decrease the turning radius. Negative radius values are interpreted as commands for turning left. In this thesis, the joystick commands are replaced by radius reference values computed by the proposed visual path correction system.

### 3.3. Basic Walking Control Algorithms

The basic control actions, also presented in [66] are shortly described below. Figure 3.10 shows the block diagram of control actions.

### 3.3.1. Joint Level Control

The references for the leg joint positions are generated through inverse kinematics from Cartesian foot references and the ZMP based COM reference trajectory [70]. Independent PID controllers are used for joint position control. The PID controller gains are obtained via trial and error.

### 3.3.2. Foot Roll Control

The scheme computes ankle roll joint angle reference modifications in such a way that the feet are aligned parallel to the ground when they are in contact with the ground. The reference modification is the form of a first order filter applied on the foot to ground contact torques. The following reference modification law in the Laplace domain is employed for the two ankles separately.

$$\bar{\theta}_{roll}(s) = \theta_{roll}(s) + (K_{roll}/s + \lambda_{roll})\tau_{roll}(s), \quad (3.9)$$

Here  $s$  is the Laplace variable.  $\theta_{roll}$  is the ankle roll joint reference angle computed by inverse kinematics.  $\bar{\theta}_{roll}$  is the reference ankle roll angle after the reference modification.  $\tau_{roll}$  is the torque about the roll axis due to the interaction of the foot with the ground. This torque is measured by torque sensors positioned at the ankle in an experimental work.  $K_{roll}$  and  $\lambda_{roll}$  are low pass filter constants which are determined by trial and error in our approach. In the digital implementation, the Laplace domain transfer function in (4.18) is approximated by a difference equation.

### 3.3.3. Ground Impact Compensation

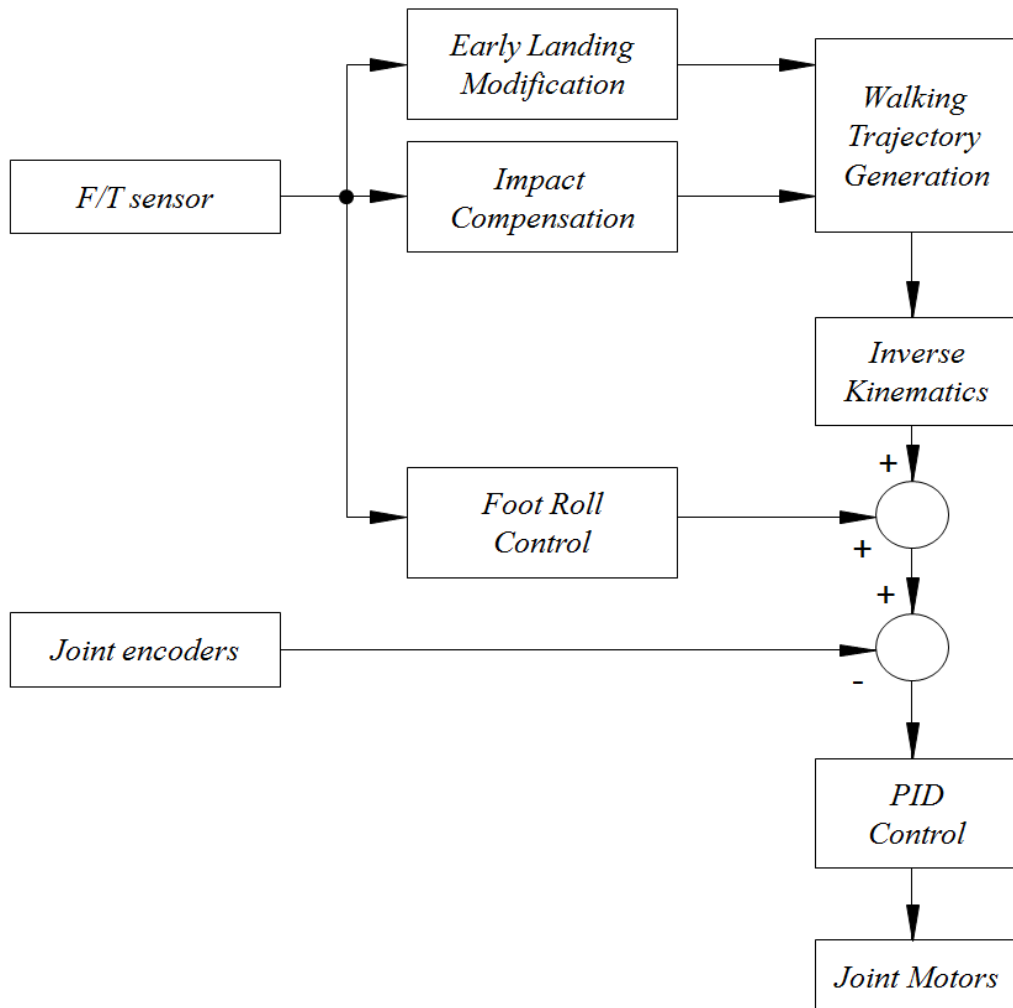
Another important problem in achieving stable walking is the impact generated at the landing of the swing foot. A shock absorbing control law is employed as a solution. This control law is activated with every landing of the swing foot. In effect, a virtual mass-spring-damper system is positioned between the hip and ankle. The following second order relation modifies the distance between the hip and sole of the landing foot.

$$\bar{l}(s) = l(s) - \frac{1}{m_l s^2 + b_l s + k_l} F_z(s) \quad (3.10)$$

Here  $l$  represents the hip-to-sole distance reference obtained from Cartesian foot reference trajectories.  $\bar{l}$  is its shock absorber modified version.  $F_z$  is the  $z$  direction component of the ground interaction force acting on the foot. Again, an ankle-mount force sensor measures this force.  $m_l$ ,  $b_l$  and  $k_l$  are the desired mass, damping and stiffness parameters of the mechanical impedance relation described in (4.19). These reference modification laws are applied for the two legs independently.

### 3.3.4. Early Landing Modification

One of the main problems of early landing of a swing foot is that when it is on the ground before the planned beginning of the double support phase, it will go on moving forward. In effect, the two feet on the ground will try to push the robot trunk in two different directions. The feet will slip; the robot will turn and possibly lose its balance. In order to avoid such a condition, the  $x$  - direction references are modified in the case of an early landing. Specifically, this modification “stops” the  $x$  direction references of the feet at their values they had at the instant of early landing. These references are kept fixed until the next walking cycle and start from their fixed values, whenever the planned  $x$  direction references (as expressed in the body frame) reach them again.



**Figure 3.10.** The walking controller block diagram

## Chapter 4

### 4. 3D RECONSTRUCTION VIA STEREO VISION

This chapter summarizes the 3D reconstruction algorithm needed for the visual odometry estimation technique described in the Chapter 5. Initially, the pinhole camera model used to relate the 3D points to pixel coordinates is introduced and the necessary information needed to invert the perspective projection is discussed. Brief summary of stereo camera calibration, undistortion and rectification steps are given as an offline procedure before the actual reconstruction is done. Finally, the online part of the algorithm is dissected into feature detection, finding stereo correspondences and disparity to depth conversion with calibrated and rectified stereo camera pairs.

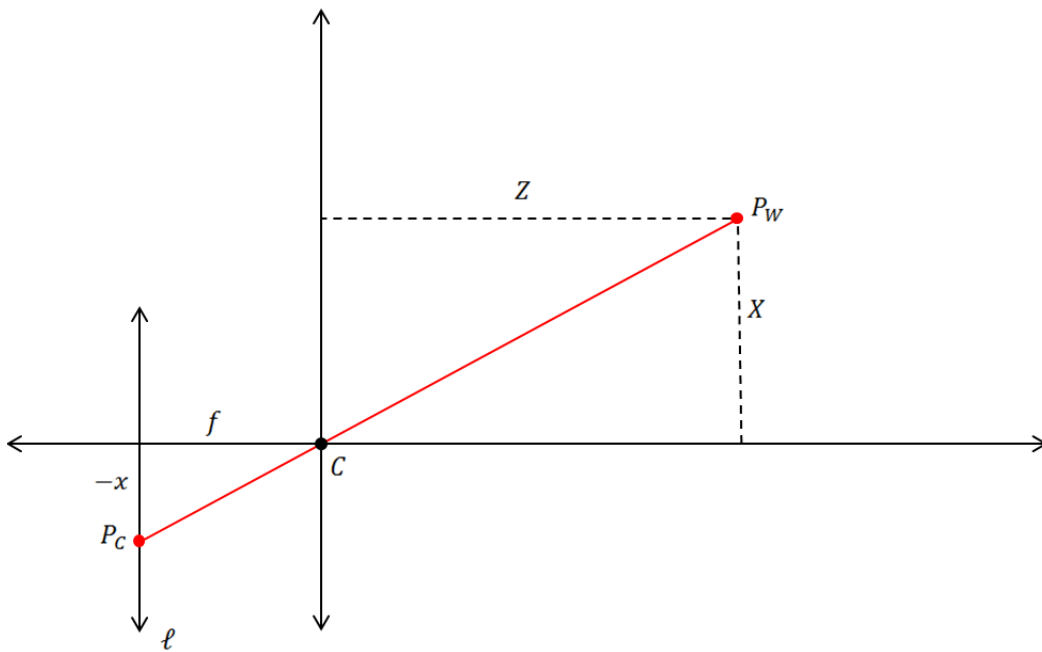
#### 4.1. The Pinhole Camera Model

The pinhole camera model [16] is a widely used simplification that relates the camera frame coordinates of the scene points to the projected points on the image plane of an ideal pinhole camera. It assumes that the scene points ( $P_W$ ) are connected to their projections on the image plane ( $P_C$ ) with lights of ray passing through the camera's center of projection ( $C$ ). The pinhole camera model is often replaced by the frontal pinhole camera model to avoid the inversion of the image plane coordinates in the camera frame. This substitution is also valid for this thesis and the stated equations are from the frontal case.

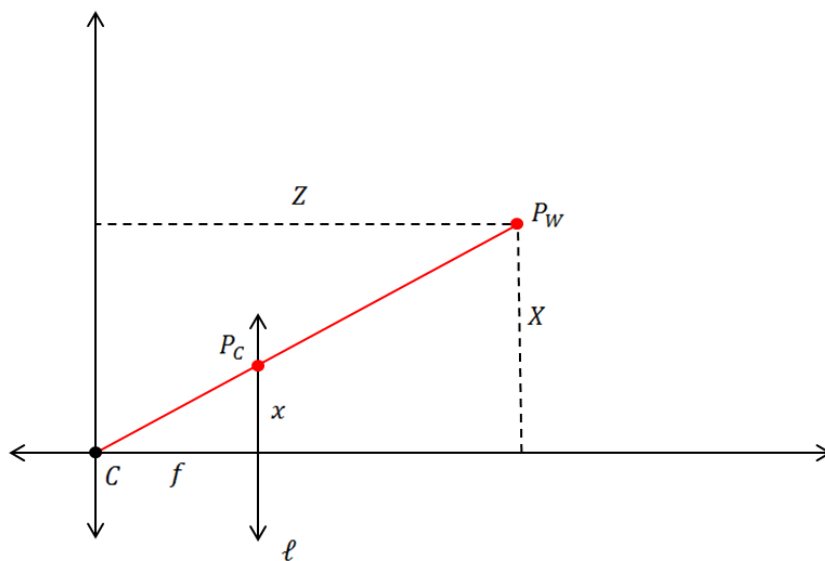
The pinhole and frontal pinhole camera models are shown in Figures 4.1 and 4.2 respectively. These models consider a 2D $\rightarrow$ 1D perspective projection case where the 2D world lies on the page and the image plane is replaced by a line ( $\ell$ ). Triangle similarity is used to derive the equation (3.1).

$$x = \frac{f}{Z}X \quad (4.1)$$

where  $x$  is the 1D image coordinate,  $X$  is the camera frame coordinate of the scene point in the same direction as  $x$ ,  $Z$  is the camera frame coordinate of the scene point in the direction of projection (*depth*) and  $f$  is the distance between the camera center and the image plane (*focal length*). The focal length is a property of the optic lens present in the camera setup, and is often fixed in machine vision applications.



**Figure 4.1.** The pinhole camera model



**Figure 4.2.** The frontal pinhole camera model

For the 3D→2D perspective projection case, since the coordinates are decoupled, this equation can be generalized onto the secondary axis to get the equation (4.2).

$$y = \frac{f}{Z}Y \quad (4.2)$$

Then, the perspective projection from a pinhole camera can be expressed in vector notation using homogeneous coordinates:

$$Z\mathbf{x} = \begin{bmatrix} f & & 0 \\ & f & 0 \\ & & 1 & 0 \end{bmatrix} \mathbf{X}_C \quad (4.3)$$

where  $\mathbf{x}$  and  $\mathbf{X}_C$  are defined as:

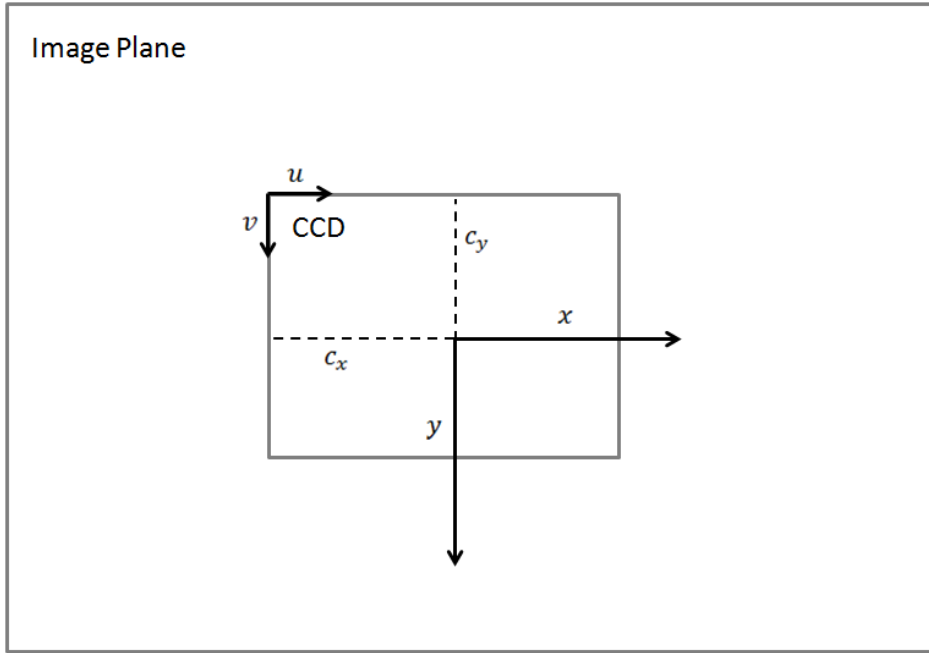
$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4.4)$$

$$\mathbf{X}_C = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (4.5)$$

The equation (4.3) defines a relationship between the image plane and the 3D coordinates of the point. Bear in mind that the pixel coordinates of a digital camera, are not the same as the image plane coordinates. The difference between the coordinate frames comes from various factors:

- The units in the image plane coordinates are the same as the real world units (e.g. *meters*); on the other hand the pixel coordinates are measured in *pixels*.
- The origin of the image plane coordinates is projection of the camera center (or the intersection of the plane with the principal axis), whereas the origin of the pixel coordinates is generally taken to be the top-left corner of the CCD.
- Cheap CCD sensors might not have perfectly rectangular pixels, so the transformation between the skewed pixel coordinates and the ideal image plane coordinates can be an affine one.

Figure 4.3 depicts the relation between the CCD sensor and the ideal image plane, where  $(u, v)$  are the pixel coordinates and  $(c_x, c_y)$  are the coordinates of the principal axis with respect to the pixel coordinate frame origin.



**Figure 4.3.** The pixel and the image plane coordinate frames

Then, the relation between the 3D camera frame and the corresponding CCD pixel coordinates of a scene point is often given as:

$$\lambda \hat{\mathbf{x}} = \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{X}_C \quad (4.6)$$

where

$$\hat{\mathbf{x}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (4.7)$$

$$\mathbf{K} = \begin{bmatrix} s_x f & \gamma & c_x \\ & s_y f & c_y \\ & & 1 \end{bmatrix} \quad (4.8)$$

$\lambda$  is called the inverse depth parameter and  $\mathbf{I}$  is a 3-by-3 identity matrix.  $\mathbf{K}$  is the intrinsic matrix of the camera and contains information about the focal length, the physical size of the pixels ( $s_x, s_y$ ), the skewness factor ( $\gamma$ ) and the position of the principal axis.

The equation (4.6) can be reduced into a more compact form by defining a 3-by-4 camera projection matrix  $\mathbf{P}$  as in:

$$\lambda \hat{\mathbf{x}} = \mathbf{P}\mathbf{X}_C \quad (4.9)$$



The model up to now dealt with the 3D points expressed in the camera frame. However, the need of a common world frame requires additional camera pose parameters. In the context of this thesis, these parameters are the ones to be estimated from visual odometry. The equation relating the world frame coordinates to camera frame ones is given in (3.10).

$$\mathbf{X}_c = \begin{bmatrix} \mathbf{R}_c^W & \mathbf{t}_c^W \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X}_w \quad (4.10)$$

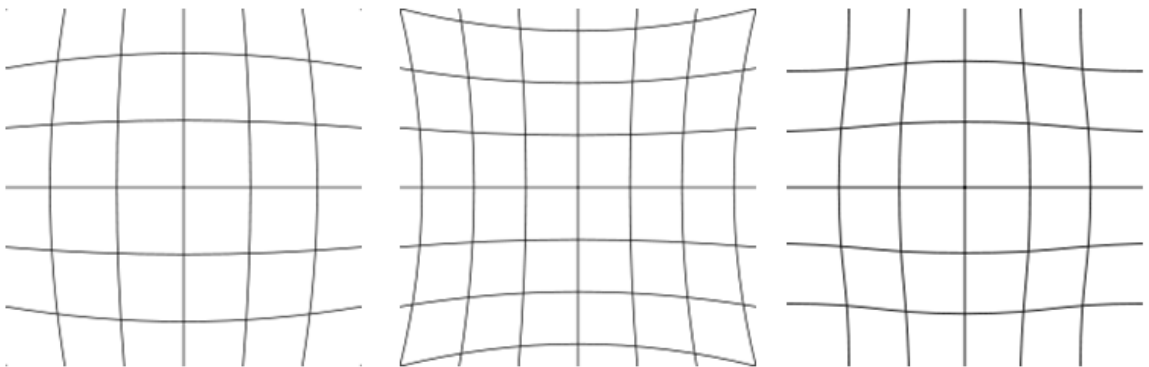
where  $\mathbf{X}_w$  is the homogeneous 3D coordinates of the scene point in the world camera frame and  $\mathbf{R}_c^W$  and  $\mathbf{t}_c^W$  are the rotation matrix and translation vector carrying the world coordinates to camera frame ones. The homogeneous transform matrix containing the  $\mathbf{R}_c^W$  and  $\mathbf{t}_c^W$  is called the extrinsic matrix.

Using the equations (3.9) and (3.10), we can write the equation (3.11) which finalizes the camera model by relating the world coordinates to pixel coordinates.

$$\lambda \hat{\mathbf{x}} = \mathbf{P} \begin{bmatrix} \mathbf{R}_c^W & \mathbf{t}_c^W \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X}_w \quad (4.11)$$

## 4.2. Lens Distortion

The real optic lenses cannot be manufactured perfectly and they introduce a distortion effect on the images. The simulated distortion types are shown in Figure 4.4.



**Figure 4.4.** Types of distortion (barrel, pincushion and mustache distortion respectively)

This effect has to be estimated and corrected to be able to get correct measurements of the 3D points. Brown's distortion model [72] is a widely used mathematical model to estimate and correct the lens distortion effect. The model fits a polynomial function depending on the distance of a pixel to the principal axis to get the distorted coordinates  $(u_d, v_d)$ .

The model is given in the following equations:

$$u_d = u + (u - c_x)(k_1 r^2 + k_2 r^4 + \dots) + [p_1(r^2 + 2(u - c_x)^2) + 2p_2(u - c_x)(v - c_y)](1 + p_3 r^2 + \dots) \quad (4.12)$$

$$v_d = v + (v - c_y)(k_1 r^2 + k_2 r^4 + \dots) + [p_2(r^2 + 2(v - c_y)^2) + 2p_1(u - c_x)(v - c_y)](1 + p_3 r^2 + \dots) \quad (4.13)$$

where

$$r = \sqrt{(u - c_x)^2 + (v - c_y)^2} \quad (4.14)$$

$k_1 \dots k_n$  are the radial and  $p_1 \dots p_n$  are the tangential distortion parameters. Although the model contains infinite series, for practical applications  $k_1, k_2$  and  $p_1, p_2$  are enough for most optic lenses.

### 4.3. Necessary Information for 3D Reconstruction

The pinhole camera and the lens distortion models allow us to reproject given 3D scene points onto our CCD sensor and find pixel coordinates, if we know the intrinsic and the extrinsic parameters of the camera.

On the other hand, the perspective projection is not an invertible process, so we cannot find a unique 3D scene point given the pixel coordinates in a single camera. Hence, auxiliary views of the scene are needed.

To sum up, at least 2 cameras with known intrinsic ( $\mathbf{K}$  matrix and the distortion coefficients) and extrinsic parameters are needed to fully reconstruct a 3D point free of any ambiguities.

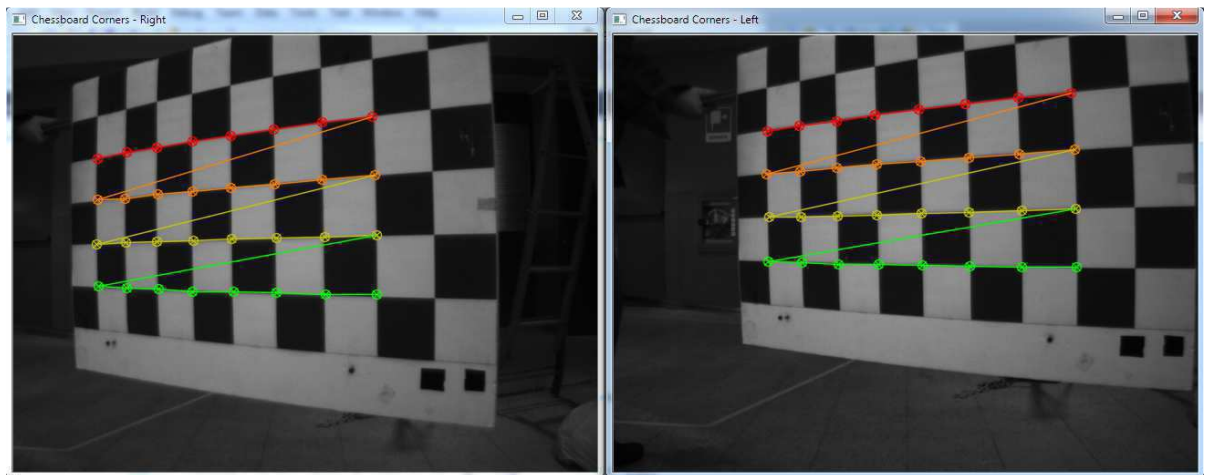
## 4.4. Stereo Camera Calibration

Camera calibration is the process of determining the intrinsic parameters of a camera. The widely used method for camera calibration is Zhang's camera calibration method given in [73]. It is carried out by gathering several images of an object with known geometry (traditionally a checkerboard pattern) to simultaneously solve for the distortion model coefficients and the intrinsic matrix.

The stereo calibration is preceded by separate calibration of the cameras to get the intrinsic parameters and then it estimates the relative rotation and translation between them. The method is similar to single camera calibration and uses a checkerboard pattern that is scene by both of the cameras simultaneously.

The estimated relative extrinsics are sufficient for reconstructing the 3D point in either one of the camera's coordinate frame.

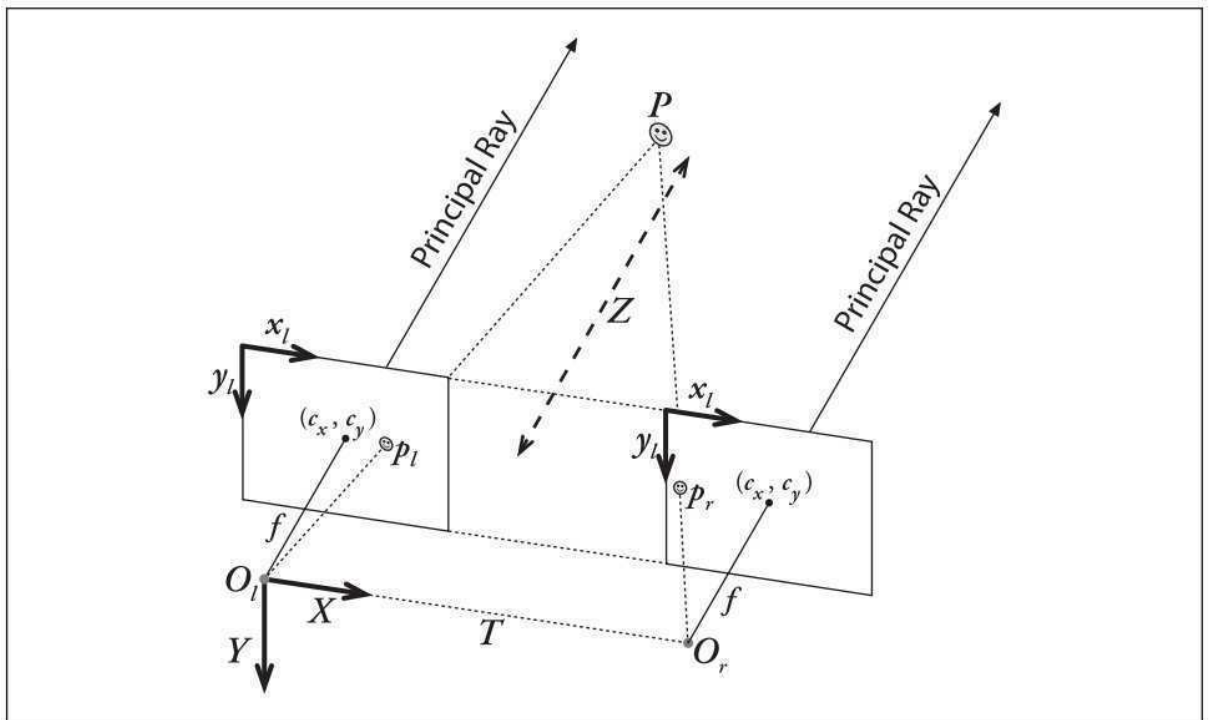
In the implementation of this thesis, the OpenCV version of the renowned "Bouguet's camera calibration toolbox for Matlab" [74] is used for both single and stereo camera calibration. A screenshot from the stereo calibration process is given in Figure 4.5.



**Figure 4.5.** The stereo calibration process

## 4.5. Stereo Rectification

*Full-frontal-parallel* configuration in stereo camera pairs defines a relative orientation that has epipoles at infinity. *Full-frontal-parallel and row-aligned* adds the constraint that the epipolar lines should be horizontal and hence the intrinsic matrices of the cameras should be identical (same pixel scale, focal length and principal axis coordinates). Figure 4.6 depicts this configuration.



**Figure 4.6.** Full-frontal-parallel and row-aligned stereo camera pair [75]

It is widely known that the efficient way of searching for stereo correspondences is looking for them on the epipolar lines [16]. The full-frontal-parallel and row-aligned camera pairs allow this search to be done on a single row of pixels which adds even more efficiency and robustness to finding matches between the camera images.

Although useful, this configuration is almost impossible to achieve with real camera sets. So, researchers have devised an algorithm, called stereo rectification, which creates a virtual camera pair in this configuration by applying a transformation on the images grabbed by the real camera pair [76]. The transformed images are then can

be used as they were grabbed by a full-frontal-parallel and row-aligned stereo camera rig.

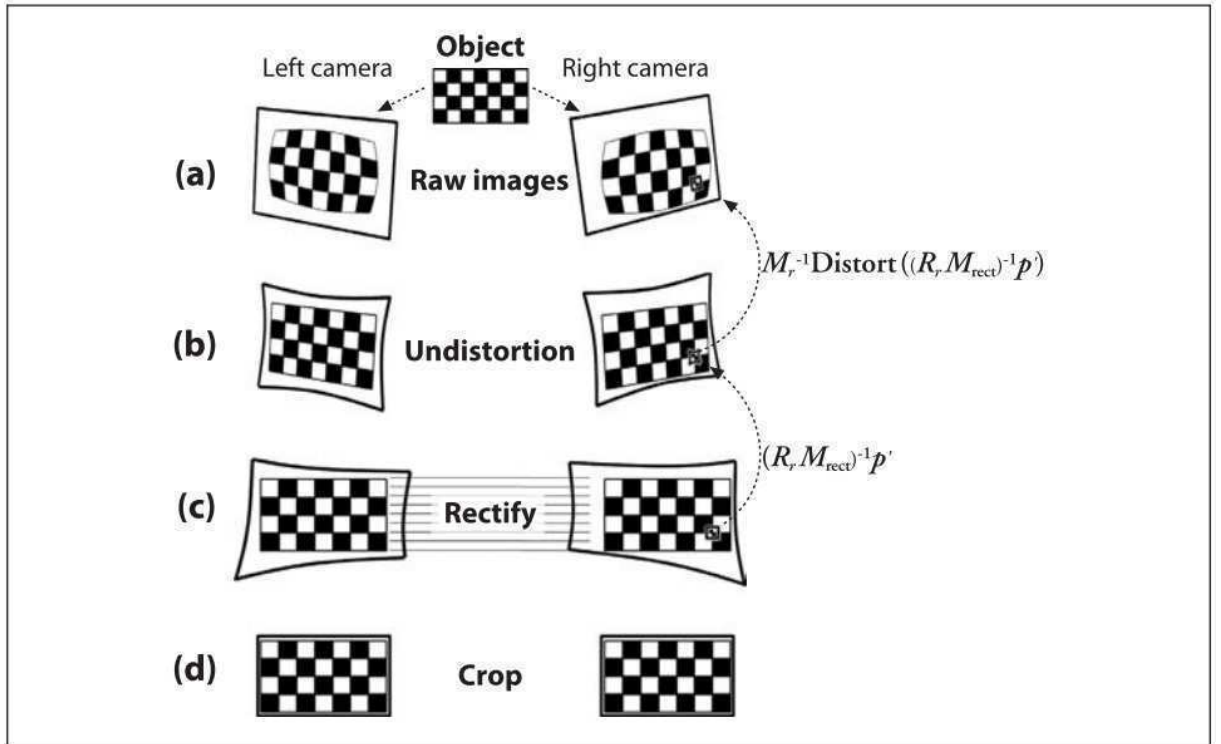
The rectified stereo camera set can be defined by a common focal length  $f$ , the principal axis location  $(c_x, c_y)$  and the translation between the cameras  $T$ . This absolute value of this translation is the length of the baseline, and the sign depends on which camera is chosen to be the main one (which defines the coordinate frame in which the 3D points are expressed). From here on, the right one is assumed to be the main camera. Following this assumption, the rectified projection matrices  $(\tilde{\mathbf{P}}_R, \tilde{\mathbf{P}}_L)$  are given in equations (4.15) and (4.16).

$$\tilde{\mathbf{P}}_R = \begin{bmatrix} f & c_x & 0 \\ & f & c_y & 0 \\ & & 1 & 0 \end{bmatrix} \quad (4.15)$$

$$\tilde{\mathbf{P}}_L = \begin{bmatrix} f & c_x & fT \\ & f & c_y & 0 \\ & & 1 & 0 \end{bmatrix} \quad (4.16)$$

It is important to note that the 3D reconstruction performed with rectified images results in 3D point coordinates expressed in the rectified camera frames.

The implementation of stereo rectification in the OpenCV creates an image warping map which contains information from rectification and undistortion estimations. This map is then can be used to transform the grabbed images (with a backwards bilinear interpolation) performing rectification and undistortion simultaneously. The result of this mapping is shown in Figure 4.7.



**Figure 4.7.** The undistortion and rectification step [75]

#### 4.6. 3D Reconstruction Using Rectified Stereo Camera Pairs

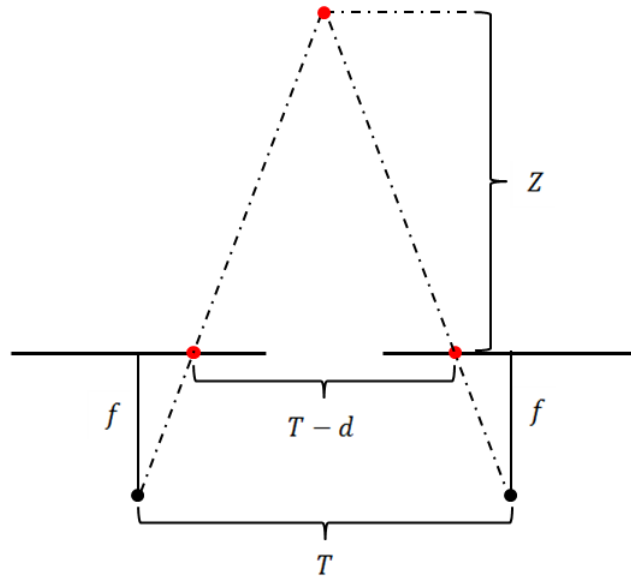
It is important to note that the 3D reconstruction algorithm explained in this section is designed to recover a sparse set of 3D points rather than a dense one. Dense 3D reconstruction algorithms aim to create depth maps covering every pixel in the images and they often contain a high level of noise.

3D reconstruction using multiple cameras with known intrinsic and extrinsic parameters is carried out by triangulation. The triangulation procedure is basically intersecting rays of light originating from the camera centers and passing through the image plane coordinates of the matching features. Rectified stereo camera pairs facilitate this triangulation step by introducing a concept called *binocular disparity*.

The binocular disparity ( $d$ ) is the difference between the horizontal coordinates of the stereo correspondences as in:

$$d = u_L - u_R \quad (4.17)$$

This disparity information is enough to recover depth from triangle similarity. The geometry of the problem is shown in Figure 4.8.



**Figure 4.8.** Triangulation with rectified stereo cameras

When this triangle similarity is solved, the inverse depth  $\lambda$  can be injected to the equation (4.9) to solve for the 3D coordinates. An efficient matrix-vector product method to perform this calculation is given in [75]. Using the disparity to augment the corresponding 2D homogeneous pixel coordinates in the right camera:

$$\bar{\mathbf{x}} = \begin{bmatrix} u_R \\ v_R \\ d \\ 1 \end{bmatrix} \quad (4.18)$$

The reconstruction of the 3D coordinates in the right camera frame  $\mathbf{X}_R$  is given in the following equations.

$$\bar{\mathbf{X}}_R = \mathbf{Q}\bar{\mathbf{x}} \quad (4.19)$$

$$\mathbf{Q} = \begin{bmatrix} 1 & & & -c_x \\ & 1 & & -c_y \\ & & 0 & f \\ & & -1/T & 0 \end{bmatrix} \quad (4.20)$$

$$\bar{\mathbf{X}}_R = \begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \\ W \end{bmatrix} \quad (4.21)$$

Then,

$$\mathbf{X}_R = \frac{\bar{\mathbf{X}}_R}{W} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (4.22)$$

The equation (4.22) normalizes the homogeneous coordinates to recover the 3D scene point.

It can be seen by inspection that the matrix  $\mathbf{Q}$  encodes all the necessary information to define the rectified stereo pair.

After this point, the 3D reconstruction algorithm only needs a feature point in the right camera image and a disparity value attached to it. The final two sections of this chapter address these issues.

#### 4.7. Feature Detection

There are a large number of tools for feature detection in the literature. Harris corner detector [77] is one of the oldest methods and it is still widely used. Shi-Tomasi [64] detects corners that are easier to track, and this method is actually the choice of the visual odometry algorithm of H7 [62-63]. Recent, state-of-the-art feature detection algorithms mostly address on finding scale and orientation invariant features for robust matching. SIFT [78] and SURF [79] are popular examples.

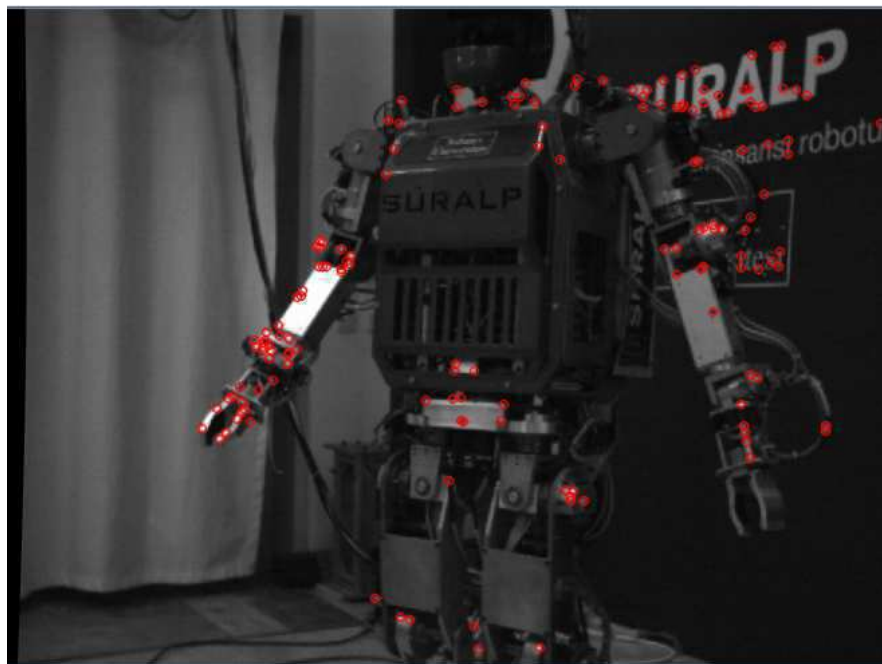
The problem of searching for stereo correspondences is easier than most feature matching problems, especially in a rectified system. The images are almost identical and there is a very good guess on where to search for matching features. This makes scale, orientation and illumination invariance properties offered by high-end descriptors simply redundant. Considering the trade-off between the computational cost and the quality of the resulting features, it is just not worth it.

FAST feature detector [80], which is a recent algorithm, has been the choice of the author of this thesis. This algorithm is specifically designed for real-time vision systems with relatively low computational power and it has gained instant popularity



among mobile device applications. Naturally, the trade-off between speed and quality applies to this situation also; the algorithm results in a relatively higher number of false corners. But, the epipolar constraint is deemed to be sufficient for eliminating the unreliable features. The algorithm is also much faster than the alternatives like Harris or Shi-Tomasi corner detectors.

The OpenCV implementation of the FAST feature detector has been adopted and utilized. An example set of resulting features is shown in Figure 4.9.



**Figure 4.9.** FAST corner features

The FAST feature detector has only one parameter, which is the threshold that needs to be surpassed for a pixel to be labeled as a corner feature. The threshold is related to the difference in intensity between the candidate corner features and their neighborhoods, so it can be seen that the low contrast parts in the image do not respond as corners.

## 4.8. Subpixel Corner Estimation

The FAST feature detector results in integer pixel coordinates that are estimated to contain corners. Considering that the camera resolutions are low (640x480) for real-time applicability, the integer coordinates for corners are too discrete to perform reliable 3D reconstruction.

OpenCV has an iterative subpixel corner refinement method [75] implementation and although it results in good estimations; the computational cost makes this step the bottleneck of the feature detection and matching step of the algorithm. Hence, the detected features are not refined into subpixel corners unless they can be matched to the left image. Once they are matched, the subpixel corner estimation steps in before the actual disparity to depth conversion happens.

## 4.9. Stereo Matching

The implemented feature matching algorithm is tailored to work with features from rectified stereo images. The algorithm takes the features found in the right camera and starts a linear search on the horizontal epipolar line to the right, evaluating a sum of squared differences (SSD) cost function around a neighborhood. The linear search is also limited to a very few number of pixels between a minimum and a maximum allowed disparity. The candidate with the lowest SSD score is chosen to be the matching feature in the left image. If the minimum SSD score is above a certain threshold  $t$ , the algorithm decides that the best match candidate is unreliable and eliminates the feature from the right camera image.

The SSD score of a match candidate  $(u_L, v_L)$  against a corner feature in the right camera image  $(u_R, v_R)$ , for a given neighborhood size  $w$  is defined as:

$$M \triangleq (w - 1)/2 \quad (4.23)$$

$$SSD = \sum_{i=-M}^M \sum_{j=-M}^M [I_R(u_R + i, v_R + j) - I_L(u_L + i, v_L + j)]^2 \quad (4.24)$$

The notation  $I_C(u, v)$  in the equation (4.24) is the intensity level at the pixel coordinates  $(u, v)$  in the image grabbed by the camera  $C$ .

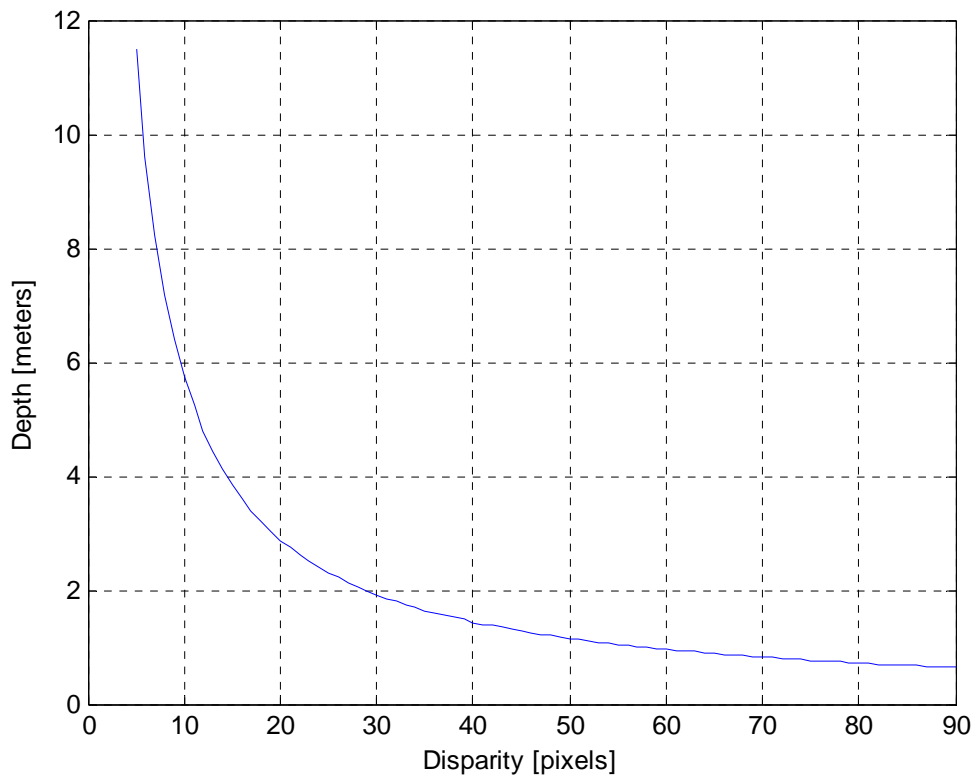
The SSD score is usually a very crude metric for robust feature matching and not feasible to be used when the match is searched on the whole image. Thanks to the rectification process and the limited search on the epipolar line this method proved to be sufficient for finding stereo matches.

The algorithm has a considerable amount of parameters, but tuning them to work indefinitely on a specific vision system (cameras and lenses) is possible.

The window size and the matching threshold are quite intuitive. They offer a trade-off between performance and reliability. As the window size parameter gets larger, the probability of falsely matching any other neighborhood drops significantly; but the computational cost increases. As for the threshold  $t$ , this parameter sets a confidence level target on the possible matches and as it gets larger a smaller amount of more robust feature matches are found.

The minimum and maximum disparity values to be searched are dependent on the baseline of the stereo system, the camera resolutions and the focal lengths of the optics used.

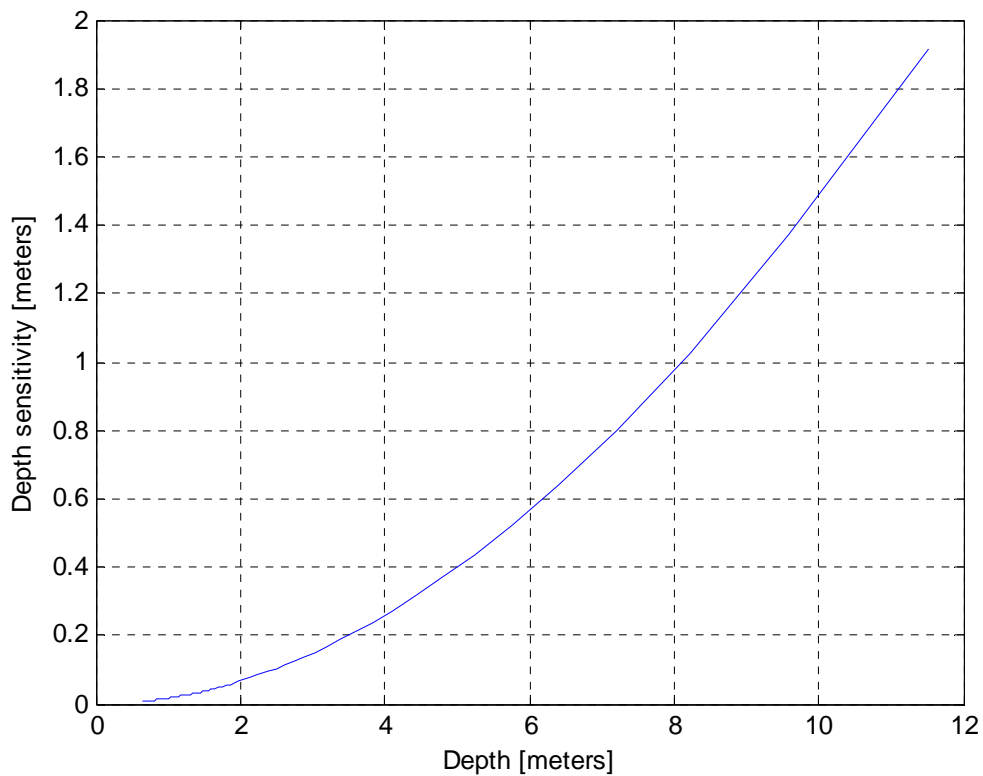
Since the disparity is inversely proportional to depth, minimum disparity sets an upper limit to the scene depth of searched points and similarly maximum disparity sets a lower limit. In order to choose these limits, plotting depth against disparity for the stereo vision system is useful. The depth vs. disparity plot of the stereo camera rig used on SURALP is shown in Figure 4.10.



**Figure 4.10.** Depth vs. disparity plot of the stereo camera rig on SURALP

Since the relation is nonlinear, the resolution in depth is varying and it gets coarser as the points go further away. This stereo camera set has a narrow baseline (6cm), very similar to that of humans and fairly wide-angle lenses (with focal lengths of 5mm).

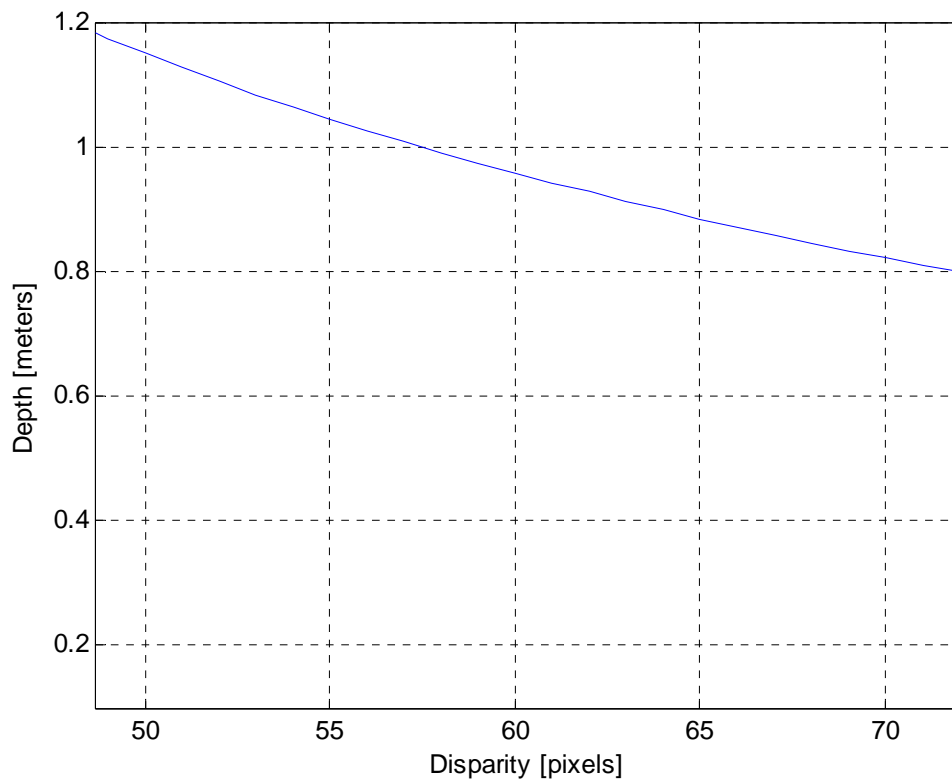
First important remark to make is the depth estimation changes drastically between pixels for far away objects. This is a dangerous behavior as even the subpixel level noise in estimation of the corner features can move the estimated 3D points in the order of meters and can destabilize the visual odometry estimation. The finite difference of the depth vs. disparity function is taken to clearly see the error sensitivity of the depth estimation and is shown as a plot against depth in Figure 4.11.



**Figure 4.11.** Depth sensitivity vs. disparity plot

Looking at the two figures, 6 meters of depth is chosen as the upper limit and hence the minimum disparity has been decided to be 10 pixels.

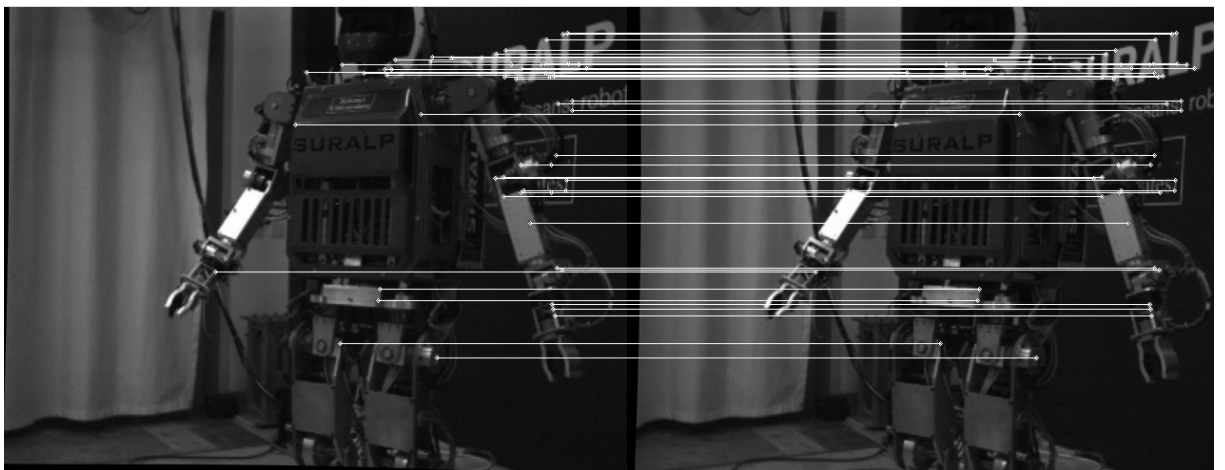
As for the maximum disparity value, the first concern is where in depth the optic lenses are focused. High frequency corner features cannot come from unfocused depths and any matched feature would probably be an outlier. Another major issue is the asymptotic behavior of the curve; as the points get closer to the camera set more and more disparity occurs between them. This adds an exponential increase in computational cost to search for nearer points. Figure 4.12 shows the change in depth between disparities 50 and 70.



**Figure 4.12.** Zoomed depth vs. disparity plot

The plot shows that 20 more pixels have to be evaluated to explore points  $\sim 0.4\text{m}$  nearer to the camera set, and this ratio gets larger really fast. Hence, a minimum depth of  $\sim 1\text{m}$  is deemed to be sufficient and the maximum disparity is chosen to be 60.

A screenshot of the resulting stereo correspondences is shown in Figure 4.13.



**Figure 4.13.** Stereo Correspondences found by SSD

Note that the epipolar lines are horizontal and the correspondences could be found on the same row. This result shows the success of the calibration, undistortion and the rectification steps.

Another important remark should be given on the necessity of the time synchronization between the cameras. When the image acquisition signals of the cameras are not synchronized, the grabbed images could (and actually would) belong to different camera pair poses. As this configuration would be different than the one estimated during the calibration steps, the initialized rectification maps would no longer transform the images into a full-frontal-parallel and row-aligned form.

This section finalizes the 3D Reconstruction via Stereo Vision chapter. The next chapter describes the proposed solution to the visual odometry estimation problem.

## Chapter 5

### 5. VISUAL ODOMETRY

This chapter introduces the proposed solution for the 6DOF visual odometry estimation problem. First two sections provide an outline of the algorithm and present the evaluation method used in the development stage. The following sections explore the details on tracking of the 3D points, camera pose estimation, robust estimation with RANSAC and bundle adjustment as an estimation refinement technique.

It is important to note that the scene is assumed to be static, and all the apparent motion of the scene features are created by the egomotion of the camera pair. Egomotion estimation in dynamic scenes requires further precautions to be taken and is out of focus of this thesis' work.

The core of the proposed algorithm is the camera pose estimation step. The 2 methods from [62] and [63] along with a novel approach proposed by the author of the thesis are presented. The results with a SLAM benchmarking video are shown.

#### 5.1. Algorithm Workflow

The offline part of the algorithm starts with stereo camera calibration and rectification. Once the intrinsic and extrinsic parameters are estimated, undistortion and rectification maps are initialized.

The online visual odometry estimation deals with 4 images at each cycle; the previous right and left and the current right and left images. The camera pose related to the previous image set is assumed to be known, since at each step the camera pose is updated according to the current images and the current images are copied on the previous ones.

Initially the world coordinate frame is coincident with the right camera frame; hence all the future estimations of the camera poses are relative to the initial pose. A



general outline of one cycle of the algorithm that is applicable to all 3 implemented methods is as following:

1. Grab new images from both cameras.
2. Apply the undistortion and rectification map.
3. If there are not enough tracked 3D points,
  - i. Detect new features in the previous right camera image.
  - ii. Find their matches on the previous left camera image.
  - iii. Estimate the subpixel corners.
  - iv. Reconstruct 3D points expressed in the right camera frame.
  - v. Convert the camera frame coordinates to the world frame using the current estimation of the camera pose.
4. Track the image features corresponding to the 3D points in the next frame.
  - i. Eliminate any point that could not be tracked (out of view, occlusions, tracking failures).
  - ii. Eliminate any point whose image coordinates in both cameras do not lie on the same row.
5. Reconstruct the tracked points from their new pixel coordinates.
6. Find the optimal camera pose that aligns the new 3D camera frame points to their world coordinate frame expressions.
7. Refine the estimated camera pose.
8. Replace the previous images with the current ones.
9. Replace the previous pixel coordinates of the tracked features with the new ones.

The major difference between this visual odometry algorithm and a SLAM framework occurs in the 4<sup>th</sup> step of the algorithm. The features that cannot be tracked are immediately deleted and forgotten; whereas in a SLAM application they are stored in a permanent map. This is why a visual odometry solution is a local estimator and suffers from inevitable drift.

In the following sections 3 different methods covering the 6<sup>th</sup> and the 7<sup>th</sup> steps of this outline will be presented.

## 5.2. Ground Truth Data

The development of such a system requires a ground truth for comparing the results and debugging accordingly. On the other hand, establishing a ground truth for visual odometry applications is a challenging task. Since it could not be obtained from the humanoid robot SURALP, a SLAM benchmark dataset from Rawsseed's Project [81] is used.

The used capture session is coded "Bicocca\_2009-02-25b" and it contains various sensor data gathered from a moving wheeled robot. The provided sensory information includes data from 3 grayscale cameras, 1 color camera, SONAR, 4 laser rangefinders (2 sets of different brands, located front and rear), IMU and robot odometry.

The robot moves through an indoor environment which is lit artificially mostly by fluorescent lights and the scene is static throughout the capture session.

The ground truth data is gathered by external sensory systems, but since it is a dataset aimed at benchmarking SLAM algorithms, only 3DOF planar trajectory of the robot is provided. The time synchronization of the sensor data and the ground truth is achieved via Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (PTP).

For the evaluation of the work described in this thesis, the right and left grayscale cameras' image sequences are used to estimate the visual odometry. The stereo calibration of the cameras is carried out using the provided calibration session images. The results are compared with the ground truth. Although the ground truth contains only 3DOF information, no such assumption is made and egomotion is estimated in 6DOF, the comparison is then carried out between the matching planar motion parameters.

The image sequences contain 26,335 pairs of images which span around 29 minutes. Although the robot mostly moves around in an environment rich with image features (narrow halls, libraries ...etc.), there are more than several occasions where the robot faces a blank wall and turns. These short periods with no image features are enough to disrupt a visual odometry or even a visual SLAM algorithm. Since the visual odometry algorithm that is being tested using these sequences is not expected to work over long distances, a relatively short sequence of images is chosen as a test bed.



**Figure 5.1.** Samples from the benchmark image sequence

A 600 frame sequence starting from the 7200<sup>th</sup> frame is proved to be a valid test for the visual odometry algorithm. During this ~40 seconds, robot moves around in a

library full of image features and performs two sharp 90-degree clockwise turns to follow an almost rectangular trajectory.

A summary of the chosen image sequence grabbed from the right camera is shown in Figure 5.1.

### 5.3. Tracking Corner Features in Time

The features are tracked between the previous and the current left and right images using the Pyramidal implementation of the Lucas-Kanade Optical Flow [82]. This method is designed for the OpenCV library and gathered quick interest and became a standard for sparse optical flow estimation in various applications.

This optical flow algorithm is chosen because it is more robust to large amount of motion between the frames compared to the other well-known optical flow estimators. Considering that the visual odometry algorithm is designed to be implemented on a humanoid robot, abrupt changes in motion caused by the walking dynamics should be expected. On the other hand, the designed algorithm contains iterative optimization techniques, where the convergence time could be slow when the initial guesses are far from the local minimum. This would create some jumps between the frames and could possibly cause the other optical flow estimators to fail.

### 5.4. Camera Pose Estimation

The camera pose estimation is approached as a registration problem aiming to find the optimal rigid body transformation that aligns the camera frame coordinates of the 3D points to their world coordinate frame expressions. This approach is also used in the visual odometry algorithm of the H7 [62-63].

From here on the right camera frame is to be referred as simply the camera frame, as it is chosen to be the main camera in the stereo pair.

The relation between the camera frame coordinates ( $\mathbf{X}_C$ ) and the world frame ones ( $\mathbf{X}_W$ ) are given in the equation (5.1).

$$\mathbf{X}_W = \mathbf{R}_W^C \mathbf{X}_C + \mathbf{t}_W^C \quad (5.1)$$

Note that this equation describes the inverse transformation of the equation in (4.10), and the coordinates are not expressed in the homogeneous form.

Estimating a 6DOF rigid body transformation ( $\mathbf{R}_W^C$  and  $\mathbf{t}_W^C$ ) is well-known to require measurements of at least 3 non-collinear points and their transformed locations. In other words, the points should define a unique plane in the 3D world. Conversely, given 3 such points, not all possible target locations define a rigid body transformation. The equation (5.2) describes the basic rigid body transformation constraint where  $\bar{\mathbf{X}}^i$  are transformed versions of the 3D points  $\mathbf{X}^i$ .

$$\|\mathbf{X}^i - \mathbf{X}^j\| = \|\bar{\mathbf{X}}^i - \bar{\mathbf{X}}^j\| \quad \forall i, j \quad (5.2)$$

The equation simply points out that a rigid body transformation should not change the distances between any points going through the same transformation. This constraint is used in [62-63] to detect outliers in the data.

Considering the case where 3 points define an exact rigid body transformation, the augmented system of equations contains 6DOF and 9 equations (3 points, 3 components in each of them), hence the system resembles an overdetermined system where actually 3 of these equations are inherently linearly dependent.

Bearing in mind that with a set of 3D points estimated from image features, one could never get a perfect set that agrees on a rigid body transformation. So solving for  $\mathbf{R}_W^C$  and  $\mathbf{t}_W^C$  should be addressed as a least squares problem. The least squares formulation with  $N$  points is stated in the equation (5.3).

$$\underset{\mathbf{R}_W^C, \mathbf{t}_W^C}{\operatorname{argmin}} \sum_{i=1}^N \|\mathbf{X}_W^i - (\mathbf{R}_W^C \mathbf{X}_C^i + \mathbf{t}_W^C)\|^2 \quad (5.3)$$

The problem with this formulation is that it requires all 9 elements of the rotation matrix to be estimated separately for a mere 3DOF orientation representation. Another issue is that since the points will never define an exact rotation, the estimated matrix elements will probably not form a rotation matrix. So the estimated matrix should then be reprojected onto  $SO(3)$ .

Orientation parameterization is a widely encountered problem. Although *axis-angle representations* and *exponential maps* perform adequately, the *unit quaternions* are shown to be better orientation representatives. They do not suffer from any representation singularities, and are argued to be more numerically stable.

Let  $\mathbf{q}$  be a unit quaternion representing the  $\mathbf{R}_W^C$ .

$$\mathbf{q} \triangleq \begin{bmatrix} q_s \\ \mathbf{q}_v \end{bmatrix} \quad (5.4)$$

$$\mathbf{q}_v \triangleq \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \quad (5.5)$$

The equation (5.6) defines the relation between  $\mathbf{q}$  and  $\mathbf{R}_W^C$ .

$$\mathbf{R}_W^C = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2(q_x q_y - q_s q_z) & 2(q_s q_y + q_x q_z) \\ 2(q_s q_z + q_x q_y) & 1 - 2q_x^2 - 2q_z^2 & 2(q_y q_z - q_s q_x) \\ 2(q_x q_z - q_s q_y) & 2(q_s q_x + q_y q_z) & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix} \quad (5.6)$$

The least squares problem is then redefined in the equation (5.7).

$$\operatorname{argmin}_{\mathbf{q}, \mathbf{t}_W^C} \sum_{i=1}^N \|\mathbf{X}_W^i - (\mathbf{R}_W^C(\mathbf{q})\mathbf{X}_C^i + \mathbf{t}_W^C)\|^2 \quad (5.7)$$

One could also use the unit quaternion as a rotation operator utilizing the quaternion multiplication. But that method is not considered in this thesis as defining the rotation matrix as a function of the unit quaternion is enough for parameterization.

The solution to this nonlinear least squares problem is carried out by the *Levenberg-Marquardt* method. The details on the theory and implementation of a generic Levenberg-Marquardt algorithm are given in the Appendix A.

Reviving the concerns about estimating a rotation matrix and keeping it in  $SO(3)$  in the process;  $\mathbf{R}_W^C(\mathbf{q})$  is a rotation matrix as long as  $\mathbf{q}$  is a unit quaternion. Hence, the least squares definition given in (5.7) is actually incomplete. The complete version should include the unit quaternion constraint to result in a nonlinear least squares optimization problem with a nonlinear constraint as given in the equation (5.8).

$$\operatorname{argmin}_{\mathbf{q}, \mathbf{t}_W^C} \sum_{i=1}^N \|\mathbf{X}_W^i - (\mathbf{R}_W^C(\mathbf{q})\mathbf{X}_C^i + \mathbf{t}_W^C)\|^2 \quad \text{s.t. } \|\mathbf{q}\| = 1 \quad (5.8)$$

The authors of [83] address this unit quaternion constraint issue. Although their solution is not adopted in the work of this thesis, they give quite useful insights on the subject.

The proposed solution is using a penalty method to iteratively force the system to converge to a minimum where the constraint is satisfied. The unit quaternion constraint is embedded into the cost function  $S$  using a weighting coefficient  $\alpha$  as shown in the equation (5.9).

$$S = \sum_{i=1}^N \|\mathbf{X}_W^i - (\mathbf{R}_W^C(\mathbf{q})\mathbf{X}_C^i + \mathbf{t}_W^C)\|^2 + \alpha \|1 - \|\mathbf{q}\|\|^2 \quad (5.9)$$

The initial value of  $\alpha$  starts relatively small and gets larger at each iteration until the obtained solution is acceptable. The pseudo-code for the penalty method using Levenberg-Marquardt is shown in the table 5.1

**Table 5.1** Pseudo-code for solving constrained NLLS using penalty method.

---

```

Constrained_NLLS(alpha_initial, alpha_max,
alpha_growth, q_error_max)
alpha ← alpha_initial
beta ← initial guess on position and orientation
feasible ← false
failed ← false

while (alpha < alpha_max && not feasible && not
failed)
    Levenberg_Marquardt(alpha, beta)
    if(Levenberg_Marquardt failed)
        failed ← true
    else
        q_current ← extract quaternion from beta
        q_error ← 1-norm(q_current)
        if(q_error < error_max)
            feasible ← true
        else
            alpha ← alpha*alpha_growth

if(alpha >= alpha_max)
    failed ← true
else
    extract q and t from beta

return q, t, failed

```

---

The Levenberg-Marquardt method requires the cost function  $S$  given in (5.9) to be rearranged on vectors  $\mathbf{Y}$  and  $\mathbf{F}$  containing all  $N$  points and the weighted unit quaternion constraint as shown in the equation (5.10).

$$S = \|\mathbf{Y}_I - \mathbf{F}_I\|^2 \quad (5.10)$$

$$\mathbf{Y}_I = \begin{bmatrix} \mathbf{X}_W^1 \\ \mathbf{X}_W^2 \\ \vdots \\ \mathbf{X}_W^N \\ \sqrt{\alpha} \end{bmatrix} \quad (5.11)$$

$$\mathbf{F}_I = \begin{bmatrix} \bar{\mathbf{X}}_C^1 \\ \bar{\mathbf{X}}_C^2 \\ \vdots \\ \bar{\mathbf{X}}_C^N \\ \sqrt{\alpha} \|\mathbf{q}\| \end{bmatrix} \quad (5.12)$$

$$\bar{\mathbf{X}}_C^i = \mathbf{R}_W^C \mathbf{X}_C^i + \mathbf{t}_W^C \quad (5.13)$$

As mentioned before, 3 different methods for camera pose estimation will be presented in this section, hence  $\mathbf{Y}_I$  and  $\mathbf{F}_I$  are subscripted  $I$  to avoid confusion in notation with the other methods.

The parameter vector  $\boldsymbol{\beta}$  to be updated by the Levenberg-Marquardt iterations is given in the equation (5.14).

$$\boldsymbol{\beta}_I = [t_x \quad t_y \quad t_z \quad q_s \quad q_x \quad q_y \quad q_z]^T \quad (5.14)$$

Since the rigid body transform equation (5.13) only contains a small amount of parameters, the Jacobian matrix  $\mathbf{J}$  is derived analytically by symbolic derivation and it is hardcoded into the implemented Levenberg-Marquardt routine.

The authors of [62] suggest that this estimation has to be carried out in a *RANSAC* manner for robustness against outliers in the data. A generic implementation of *RANSAC* is given in the Appendix B.

The gross outliers arise from false matches between the right and left images, but it has been observed that these outliers occurred rarely and even then, they are quickly eliminated because they cannot maintain the epipolar constraint when the robot is moving (see the 4<sup>th</sup> step of the algorithm workflow).



The main contribution of RANSAC comes from the fact that during the course of the robot, different sets of 3D points are added when the number of tracked points falls below a certain threshold. So, different sets are estimated at separate time steps with different errors in camera pose estimation; hence they do not really belong to the same point cloud. When a single point cloud estimated in the camera frame is tried to be aligned to a skewed set of points, the least squares estimation may converge to nonsense values. RANSAC finds the largest set that is consistent in the mixed point cloud.

Although RANSAC finds the largest set with no problem, if the number of consistent groups in the data is large and their populations are low, none of the estimated models can contain enough number of potential inliers to be considered as a valid hypothesis. Bundle adjustment method [14] is then used from time to time to merge these different sets into a single consistent point cloud.

Bundle adjustment is the process of minimizing the reprojection error to simultaneously adjust the projection parameters along with the reconstructed 3D points. The minimization of the reprojection error given is traditionally done via Levenberg-Marquardt method. The parameters to be estimated generally include all the intrinsic and extrinsic parameters of the cameras and the 3D points. But in this work, the intrinsic parameters and the geometric relation between cameras are assumed to be constant and known as they were estimated by the stereo calibration step; hence only the motion parameters of the stereo camera pair are refined along with the 3D points. Since the unit quaternions are used to represent the orientation of the camera pair, the bundle adjustment is also done using the penalty method solving the constrained nonlinear least squares problem given in (5.14).

$$\underset{\mathbf{q}, \mathbf{t}_W^C, \mathbf{X}_W^i}{\operatorname{argmin}} \sum_{i=1}^N \|\tilde{\mathbf{x}}_R^i - \hat{\mathbf{x}}_R^i\|^2 + \sum_{i=1}^N \|\tilde{\mathbf{x}}_L^i - \hat{\mathbf{x}}_L^i\|^2 \quad \text{s. t. } \|\mathbf{q}\| = 1 \quad (5.14)$$

where  $\tilde{\mathbf{x}}_R^i$  and  $\tilde{\mathbf{x}}_L^i$  are the observed pixel coordinates of the point  $\mathbf{X}_W^i$  in the rectified right and left images, and  $\hat{\mathbf{x}}_R^i$  and  $\hat{\mathbf{x}}_L^i$  are the reprojections of the  $\mathbf{X}_W^i$  onto the rectified right and left cameras using the following equations.

$$\hat{\mathbf{x}}_R^i = \frac{\tilde{\mathbf{P}}_R \begin{bmatrix} \mathbf{R}_C^W & \mathbf{t}_C^W \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X}_W^i}{\lambda_R^i} \quad (5.15)$$

$$\hat{\mathbf{x}}_L^i = \frac{\tilde{\mathbf{P}}_L \begin{bmatrix} \mathbf{R}_C^W & \mathbf{t}_C^W \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x}_W^i}{\lambda_L^i} \quad (5.16)$$

where  $\tilde{\mathbf{P}}_R$  and  $\tilde{\mathbf{P}}_L$  are the rectified projection matrices which are defined in the Chapter 4.

Please note that  $\mathbf{R}_C^W$  and  $\mathbf{t}_C^W$  define the inverse of the rigid body transformation estimated from (5.8) and their relation is given in (5.17) and (5.18).

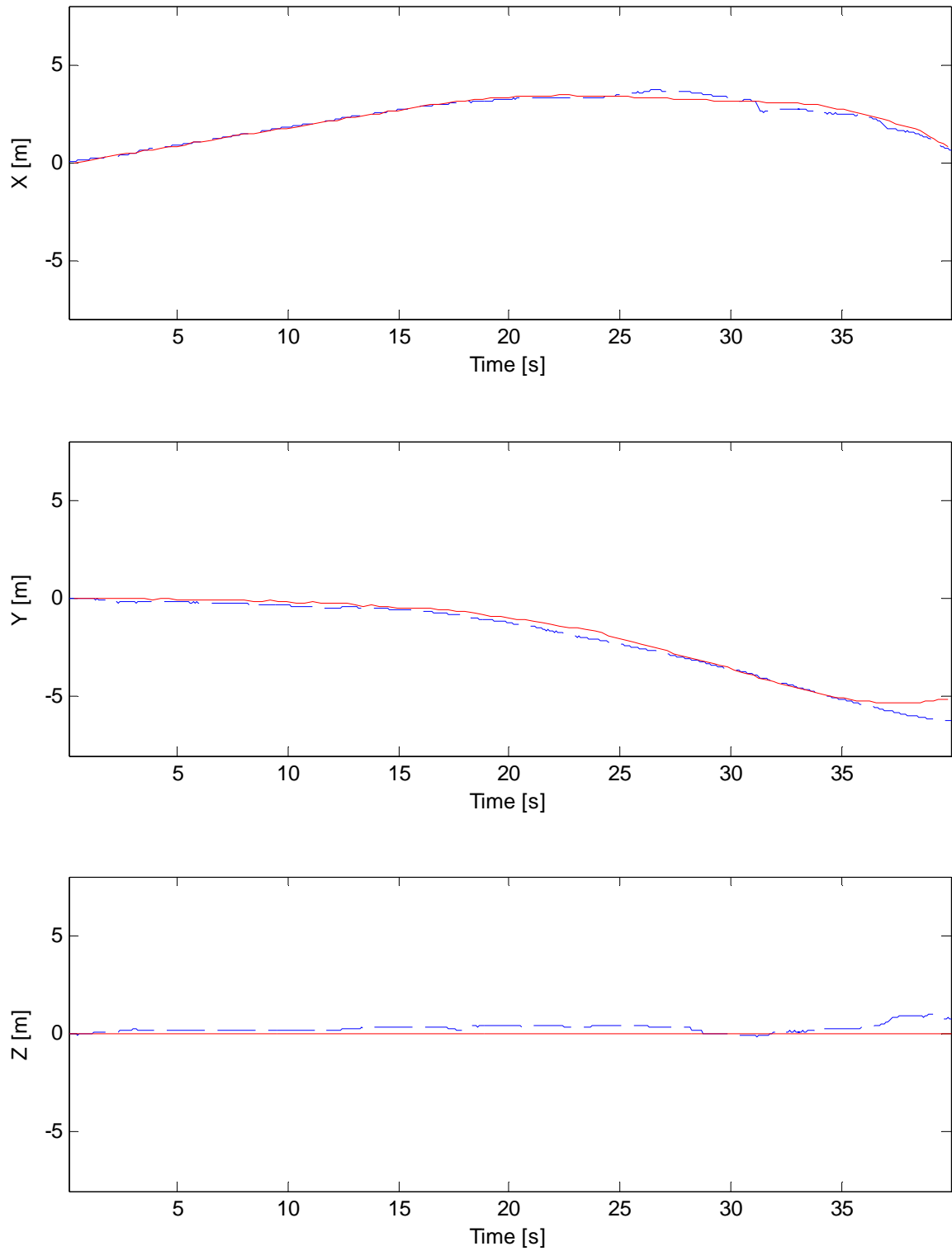
$$\mathbf{R}_C^W = (\mathbf{R}_W^C)^T \quad (5.17)$$

$$\mathbf{t}_C^W = -(\mathbf{R}_W^C)^T \mathbf{t}_W^C \quad (5.18)$$

One last note on the bundle adjustment process is the Jacobian matrix needed for the Levenberg-Marquardt iterations is not derived analytically because the expressions were too large to be hardcoded. Computing it numerically with discrete differentiation over the parameters is a common approach in bundle adjustment applications and it is done so in this thesis.

To sum up, solving the constrained nonlinear least squares problem given in the equation (5.8), using the parameter vector (5.13) in a RANSAC context, and refining the results via bundle adjustment will be referred as the *Method I*. The results with the benchmark image sequence for Method I are given below. The solid lines belong to the ground truth and the dashed ones are the estimated values.

Figure 5.2 shows the estimated position in x, y and z-axis in time. The ground truth does not contain z-axis information, but since the robot is a wheeled one,  $z = 0$  line is drawn on the figure for the sake of completeness.

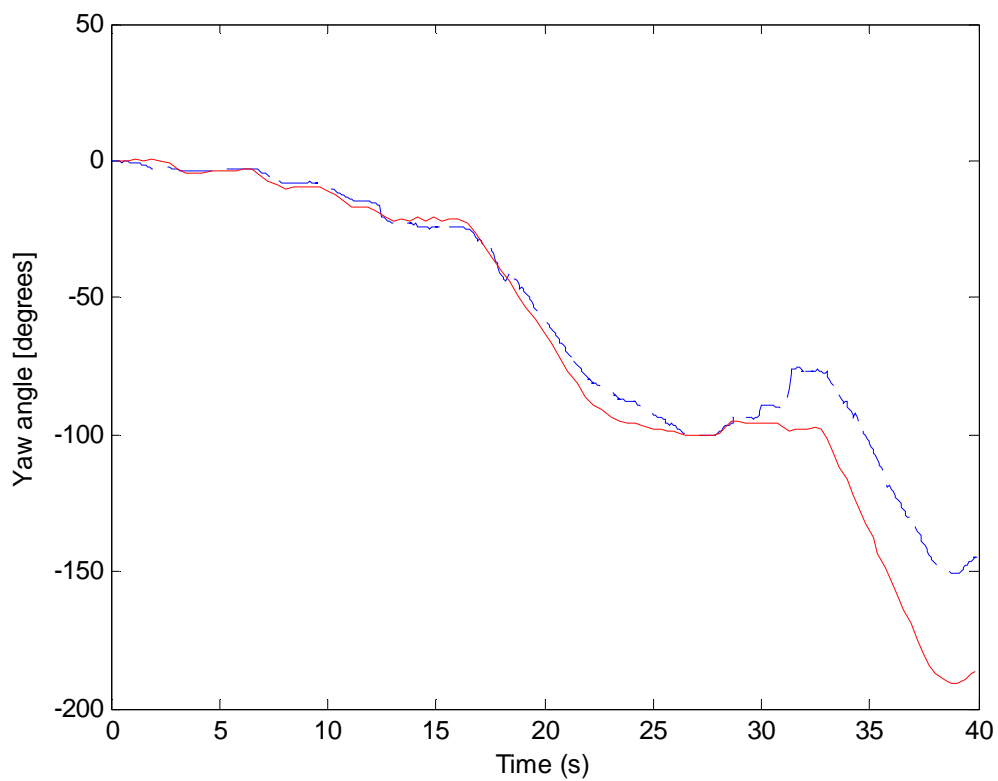


**Figure 5.2.** Method I position estimation (dashed) and ground truth (solid) vs. time

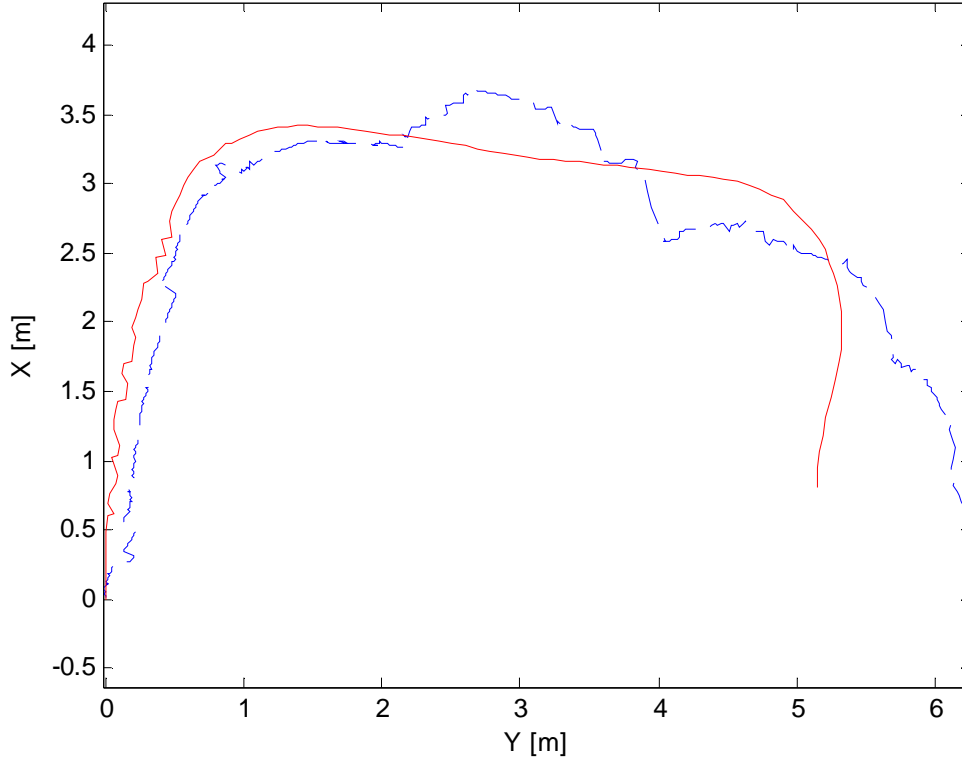
The comparison in orientation is more challenging than the position. The comparison is chosen to be made between the ground truth orientation angle (out of plane, yaw) and the angle contained in the unit quaternion's axis-angle representation.

Figure 5.3 shows both the ground truth and the estimated angle against time. The drift in orientation estimation can be seen clearly on this plot. Considering that the robot moves only forward and not in lateral directions, the drift in orientation estimation hinders the position estimation greatly.

Robot's ground truth trajectory and its estimation are shown in Figure 5.4. Although the position estimations look good when plotted against time, this figure gives a more complete understanding of the data and shows the drift in position over time.



**Figure 5.3.** Estimated angle (dashed) and the ground truth (solid) vs. time



**Figure 5.4.** Estimated (dashed) and ground truth (solid) robot trajectories

*Method II* is the implementation of the approach described in [63]. The problem formulation and aim are actually the same with the previous method, but the camera pose estimation is carried out in a decoupled way.

$\mathbf{X}_W^*$  and  $\mathbf{X}_C^*$  are defined as the centroids of the two point clouds to be aligned as in

$$\mathbf{X}_W^* = \frac{1}{N} \sum_{i=1}^N \mathbf{X}_W^i \quad (5.19)$$

and

$$\mathbf{X}_C^* = \frac{1}{N} \sum_{i=1}^N \mathbf{X}_C^i. \quad (5.20)$$

Then, the centroids are aligned on the origin to eliminate the translation, and estimate only the rotation from nonlinear least squares. The new, translated point clouds  $\tilde{\mathbf{X}}_W^i$  and  $\tilde{\mathbf{X}}_C^i$  are defined as

$$\tilde{\mathbf{X}}_W^i = \mathbf{X}_W^i - \mathbf{X}_W^* \quad (5.21)$$

and

$$\tilde{\mathbf{X}}_C^i = \mathbf{X}_C^i - \mathbf{X}_C^*. \quad (5.22)$$

Hence, the least squares formulation for rotation estimation between the new point clouds is given as

$$\operatorname{argmin}_{\mathbf{q}} \sum_{i=1}^N \|\tilde{\mathbf{X}}_W^i - \mathbf{R}_W^C(\mathbf{q}) \tilde{\mathbf{X}}_C^i\|^2 \quad \text{s. t. } \|\mathbf{q}\| = 1. \quad (5.23)$$

The solution to this optimization problem is handled identically to the one in (5.8), changing the  $\mathbf{F}$  and  $\boldsymbol{\beta}$  vectors to estimate only the rotation as in

$$\boldsymbol{\beta}_I = [q_s \quad q_x \quad q_y \quad q_z]^T \quad (5.24)$$

$$\mathbf{F}_{II} = \begin{bmatrix} \bar{\mathbf{X}}_C^1 \\ \bar{\mathbf{X}}_C^2 \\ \vdots \\ \bar{\mathbf{X}}_C^N \\ \sqrt{\alpha} \|\mathbf{q}\| \end{bmatrix} \quad (5.25)$$

where

$$\bar{\mathbf{X}}_C^i = \mathbf{R}_W^C \tilde{\mathbf{X}}_C^i. \quad (5.26)$$

Then the translation estimation is given as

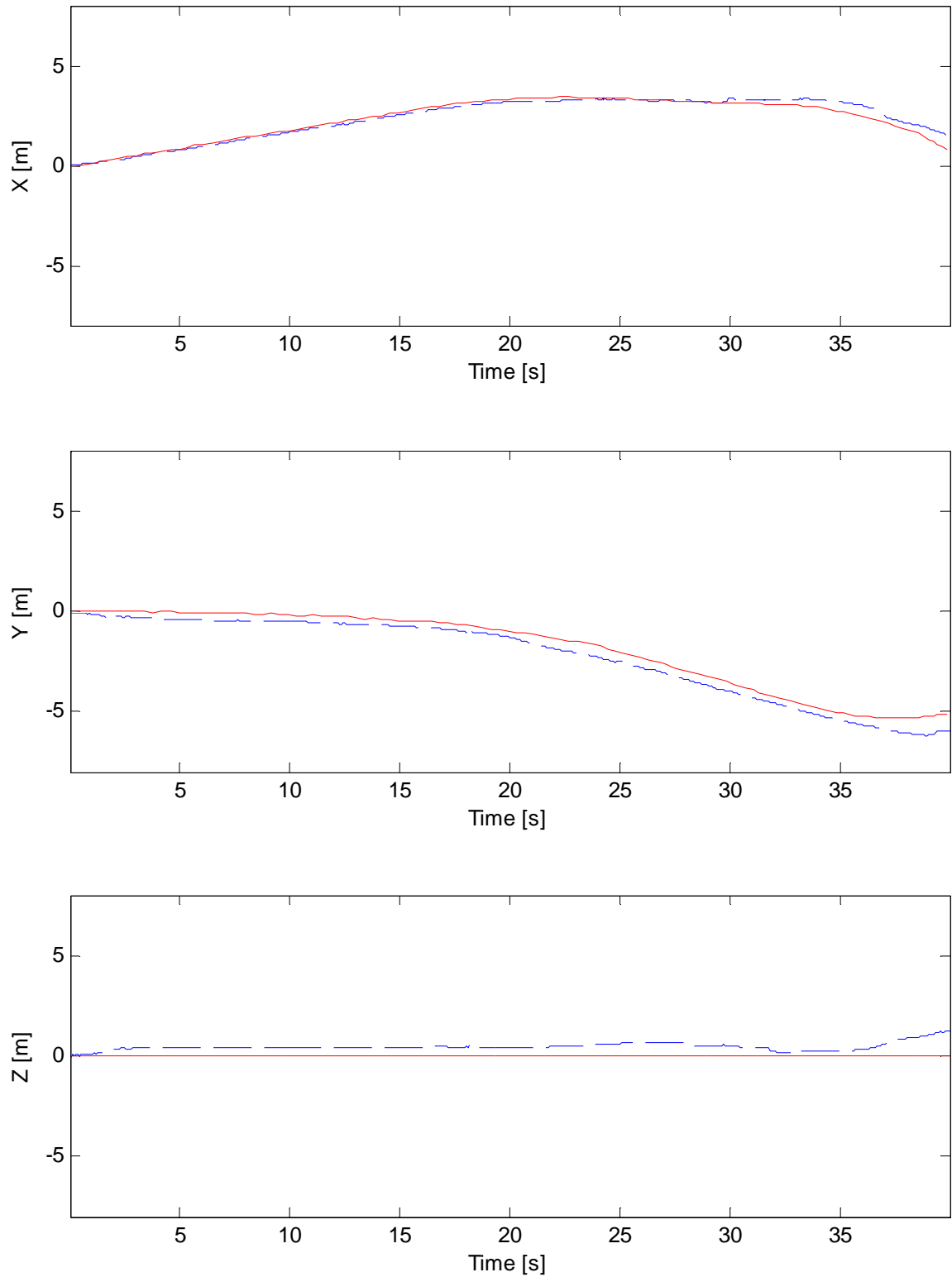
$$\mathbf{t}_W^C = \mathbf{X}_W^* - \mathbf{R}_W^C \mathbf{X}_C^*. \quad (5.26)$$

This new formulation ends up with a smaller system only estimating 4 parameters in Levenberg-Marquardt, which would possibly converge faster than the problem stated in (5.8). Also, the translation estimation is clearly related to the average translation of the points in closed form. In an ideal rigid body transformation, the points would share exactly the same translation and the average translation would be same as the original translation. For an estimated point cloud, the translation component is not the same for every point, so taking the average corresponds to a least squares solution.

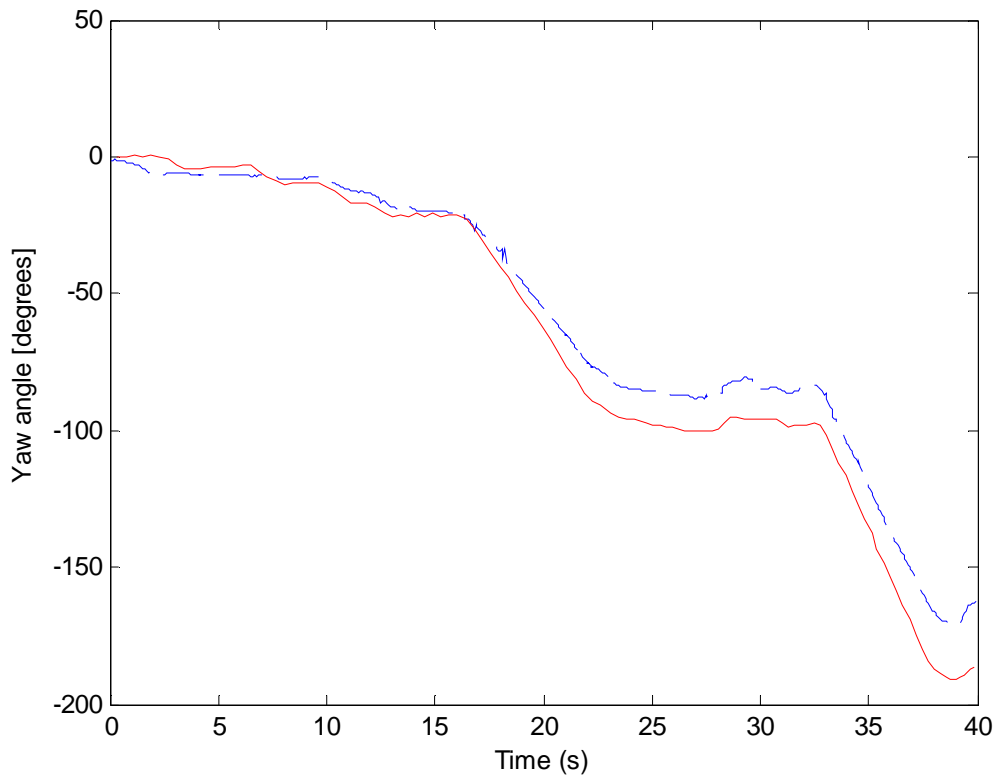
This approach is used in RANSAC model estimations and the system is bundle adjusted when needed as in Method I. The results of the Method II with the benchmark image sequence are given in Figures 5.5, 5.6 and 5.7.

The results are very similar to those of Method I. Although they look a bit better on the documented runs, the nondeterministic nature of RANSAC prevents a clear cut

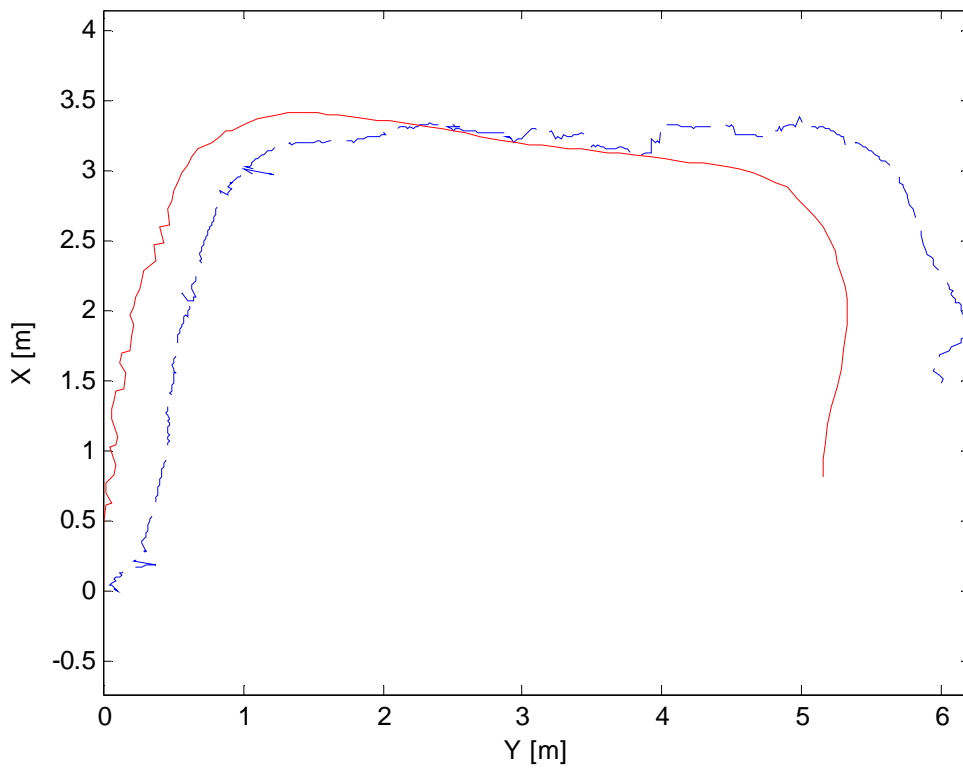
comparison between the two methods. That is to say, the differences between different runs of the same method are of the same magnitude as the shown difference between the two methods.



**Figure 5.5.** Method II position estimation (dashed) and ground truth (solid) vs. time



**Figure 5.6.** Estimated angle by Method II (dashed) and the ground truth (solid) vs. time



**Figure 5.7.** Estimated trajectory by Method II (dashed) and ground truth (solid)



Although these results are satisfactory, this is partly because the speed and efficiency of the algorithm does not affect the quality of offline processing on a pre-recorded video. This would not be the case with a real-time visual odometry task. When the time difference between the frames gets larger, initial guesses on camera pose parameters get farther away from the actual ones and cause Levenberg-Marquardt to converge to nonsense values.

The execution times of the two aforementioned methods are very similar. The cycles in which bundle adjustment is not needed are completed at 5~6 FPS on average on a notebook computer, but the bundle adjustment adds a huge load on the algorithm and the video typically freezes for a few seconds. This is not acceptable for a humanoid robot application.

The proposed *Method III* is a novel approach developed by the author of this thesis. It aims to eliminate the need for RANSAC and the bundle adjustment to get much faster cycle times at acceptable performance.

The idea is correcting the camera frame 3D point estimations so that they will align on a perfect rigid body transformation originating from the tracked world camera coordinates. So random sampling would not be needed to find a valid rigid body motion and keeping the estimation from converging to nonsense values. The method comes down to finding more robust compromises between the point clouds added at different time steps. Since different sets can be used together, the bundle adjustment process is also eliminated.

Although bundle adjustment seems to correct for the mistakes done by the egomotion estimation along the trajectory, it only finds possible solutions given the initial guess and arranges the “temporary map” of the environment accordingly. Since the initial guess on motion parameters is incorrect, it actually corrupts the correct 3D world points. The cause of the drift in position and orientation estimation is this corruption of the temporary map.

The proposed method solves for the same nonlinear least squares problem stated in (5.8) with an approach inspired by the bundle adjustment. The camera frame 3D point estimates are also estimated with the initial guess provided by the 3D reconstruction step. The problem is restated in (5.27).

$$\operatorname{argmin}_{\mathbf{q}, \mathbf{t}_W^C, \mathbf{X}_C^i} \sum_{i=1}^N \|\mathbf{X}_W^i - (\mathbf{R}_W^C(\mathbf{q})\mathbf{X}_C^i + \mathbf{t}_W^C)\|^2 \quad \text{s.t. } \|\mathbf{q}\| = 1 \quad (5.27)$$

The solution method is the same penalty method with Levenberg-Marquardt iterative minimization algorithm. The only difference with the prior method is the parameter vector  $\boldsymbol{\beta}$ , which is given as

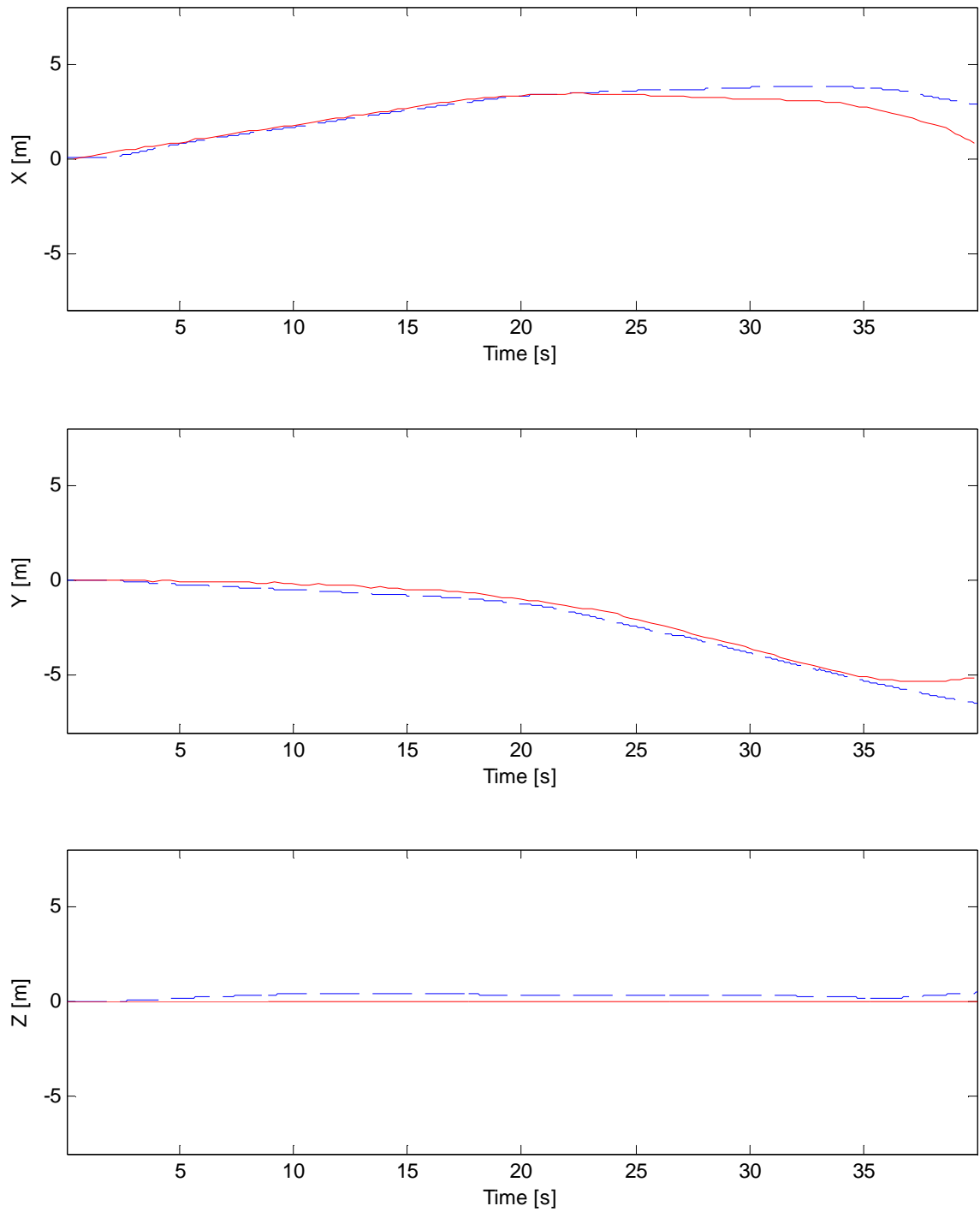
$$\boldsymbol{\beta}_{III} = [t_x \quad t_y \quad t_z \quad q_s \quad q_x \quad q_y \quad q_z \quad \mathbf{X}_C^1{}^T \quad \mathbf{X}_C^2{}^T \quad \dots \quad \mathbf{X}_C^N{}^T]^T. \quad (5.28)$$

The parameter vector is of size  $7 + 3N$ ,  $N$  being the number of points included in the process. The system to be optimized is considerably larger and similar to the size of a bundle adjustment process when carried out with the whole set of tracked points. Considering that the camera frame points are reconstructed at each cycle, the refined versions will never be used again, so one could perform this minimization over a subset of the tracked points. The selection of the number of points to be included in the camera pose estimation process becomes a tool for performance optimization.

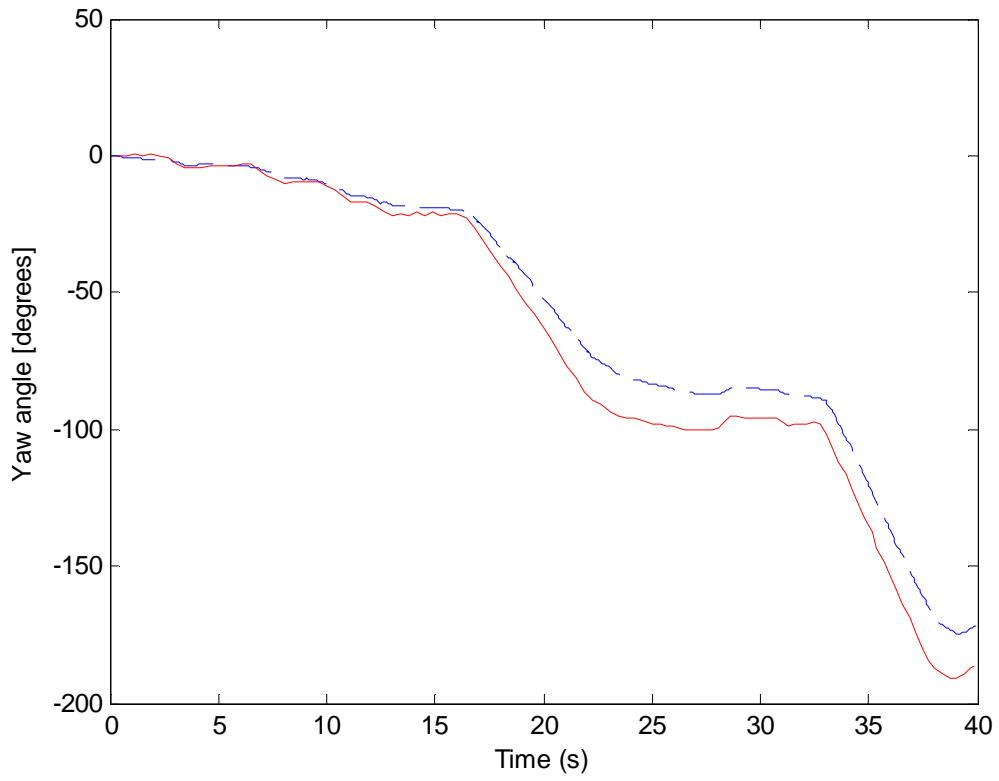
The results of Method III are given in Figures 5.8, 5.9 and 5.10. The method's performance in speed exceeded the previous ones with an execution at 12 FPS on average and the estimated trajectory is fairly smoother. The drift is increased considerably, but still manageable.

Note that by changing the motion parameters and the reconstructed camera frame points, one can find infinitely many solutions. The algorithm could easily move the camera frame points exactly on the world frame ones to get identity transformation, but it does not do so because it converges to the closest feasible solution to the initial guess. As the initial guess on the motion parameters comes from the previous camera pose estimation, the algorithm finds a compromise by transferring some of the motion to the 3D points and this accelerates the drift in estimation.

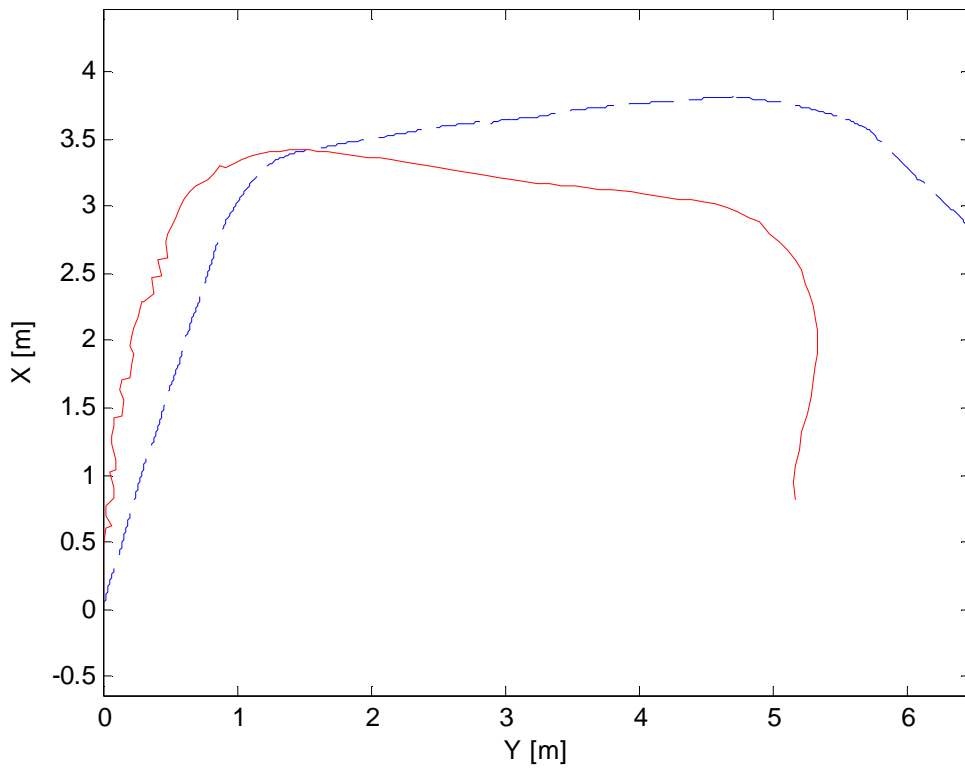
Visual odometry algorithms are never the complete solution for the self-localization problem, and the drift is inevitable. In a complete solution provided by a SLAM application, visual odometry estimation of the motion parameters and the local map would be refined anyway. So, fast and smooth visual odometry estimation could be more viable to be used in a SLAM context rather than a slow and more accurate one.



**Figure 5.8.** Method III position estimation (dashed) and ground truth (solid) vs. time



**Figure 5.9.** Estimated angle (dashed) and the ground truth (solid) vs. time with Method III



**Figure 5.10.** Estimated trajectory by Method III (dashed) and ground truth (solid)

## Chapter 6

### 6. IMPLEMENTATION ON SURALP

The planned walking path of a humanoid robot cannot always be exactly realized. Disturbance forces and torques coming from the ground contact often add up and detour the robot from its trajectory. This effect can be seen on humans trying to walk blindfolded also, since without any feedback from the environment, the direction of the walk cannot be controlled.

The scenario, in which the visual odometry estimation is tested, is based on this problem. A control method to correct for the orientation around the z-axis (out of ground plane, referred as yaw) is devised based on the orientation estimation of the visual odometry algorithm.

The control method acts on the arc walk curvature radius  $r_c$  (measured in meters) which is mentioned in the Chapter 3.

The yaw estimate  $\gamma$  and a desired yaw orientation  $\gamma_d$  is used to form a yaw error  $e_\gamma$  as in

$$e_\gamma = \gamma_d - \gamma. \quad (6.1)$$

The control action  $r_c$  is simply calculated as

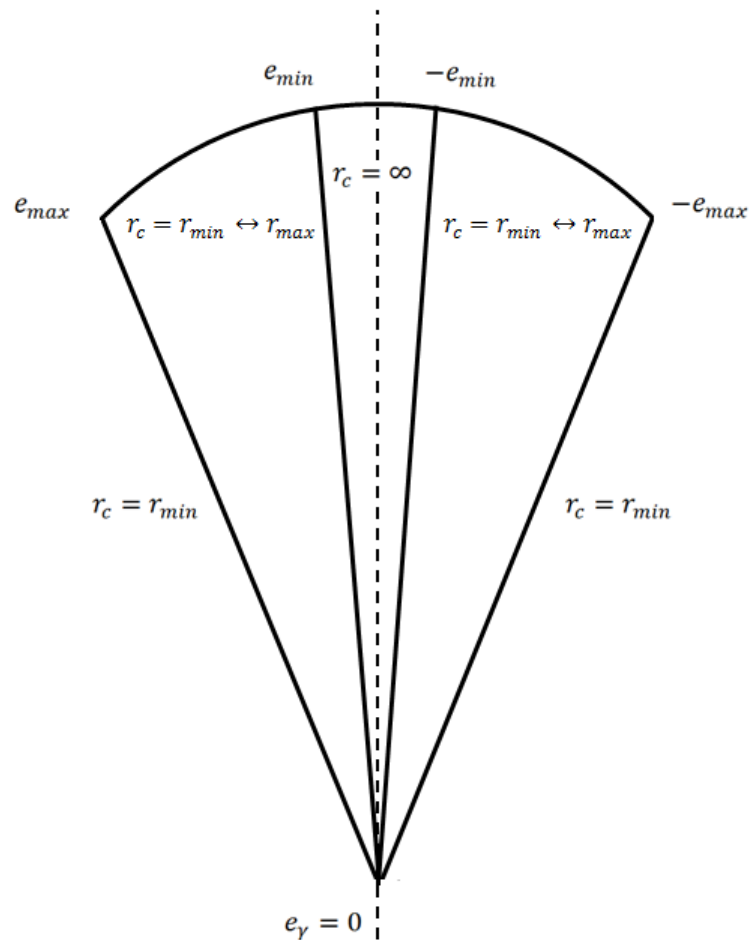
$$r_c = \begin{cases} \infty & \text{if } |e_\gamma| \leq e_{min} \\ r_{max} - \frac{r_{min}}{e_{max} - e_{min}} (e_\gamma - e_{min}) & \text{if } e_{min} < e_\gamma \leq e_{max} \\ -r_{max} - \frac{r_{min}}{e_{max} - e_{min}} (e_\gamma + e_{min}) & \text{if } -e_{max} \leq e_\gamma < -e_{min} \\ \text{sgn}(e_\gamma)r_{min} & \text{if } |e_\gamma| > e_{max} \end{cases} \quad (6.2)$$

The control input is a bit counter-intuitive as  $r_c$  is inversely proportional to the angular velocity created on the yaw axis. For a perfectly straight walk,  $r_c$  must be infinity, since this is not possible to program, a practical substitute for infinity is chosen to be 1000m.

$e_{min}$  defines a dead zone in the controller to prevent the noise on yaw estimation to overact on the walk trajectory. So if the error in yaw is smaller than a predefined control parameter, the controller does not act to correct the trajectory.

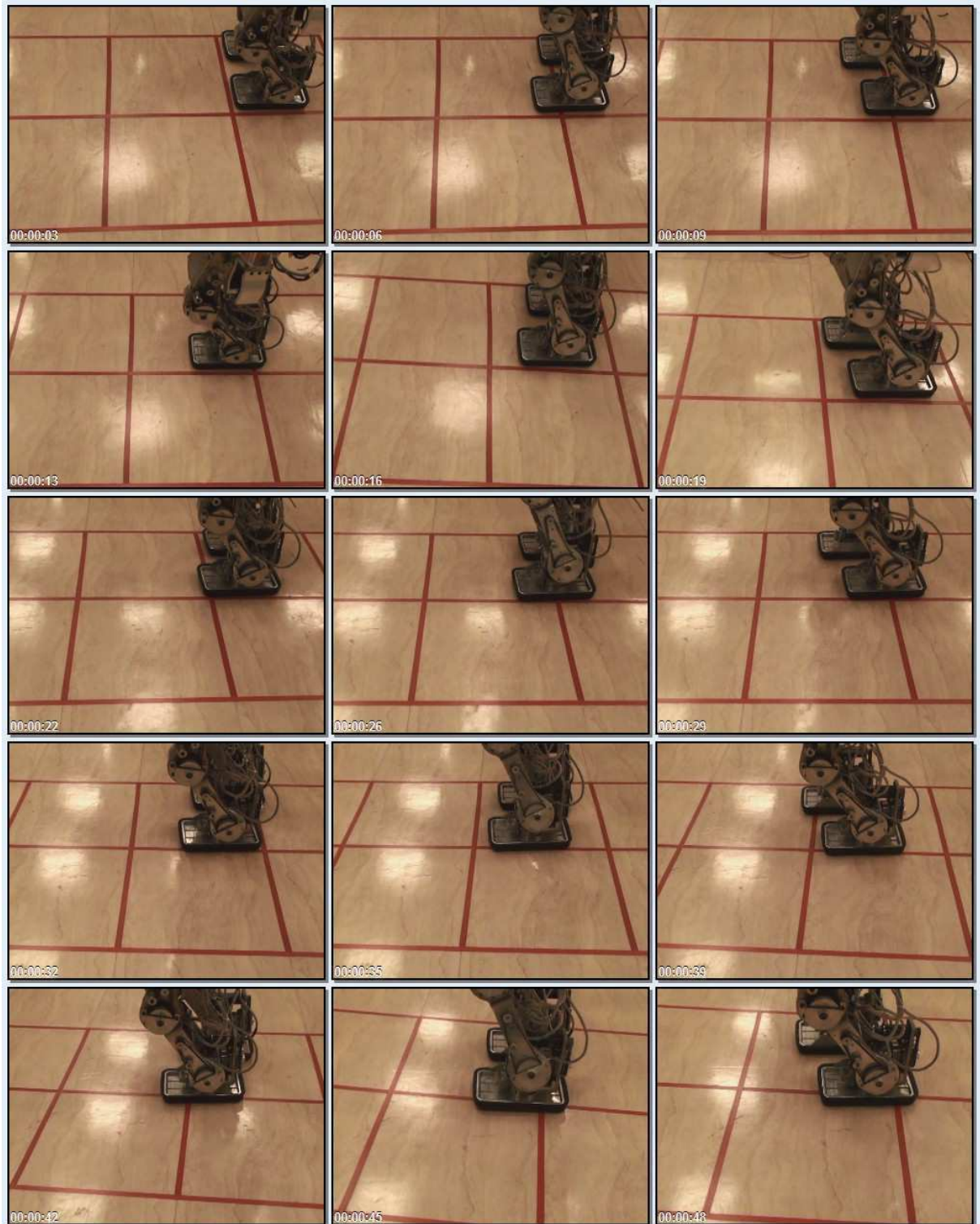
$e_{max}$  is the limit of error magnitude after which the controller performs the maximum allowed control action. The maximum action occurs when  $r_c = r_{min}$ , and  $r_{min}$  is chosen to be 0.5m which is the minimum radius that SURALP can turn without showing any sign of imbalance.  $r_{max}$  is the largest radius to be performed and is the equivalent of the action the robot must take right on the boundary of the dead zone.

Any magnitude of  $e_\gamma$  between  $e_{max}$  and  $e_{min}$  creates an interpolated arc walk radius for SURALP to follow. The visualization of the control method is shown in Figure 6.1.



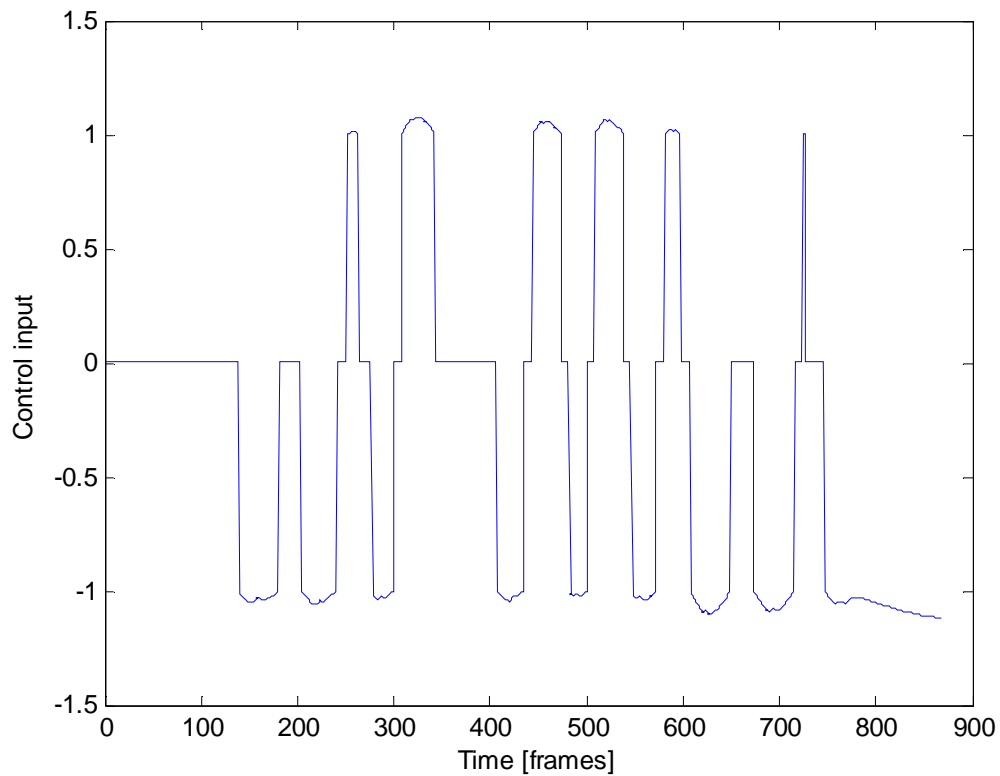
**Figure 6.1.** The arc walk curvature radius decision visualization

The Figure 6.2 shows a series of snapshots from a walking experiment consisting of 10 steps with the yaw orientation controller turned off. SURALP starts with its feet parallel to the lines marked on the floor and it is commanded to walk straight, but cannot keep its direction and walks onto the floor markers.



**Figure 6.2.** SURALP walking experiment with the yaw controller turned off.

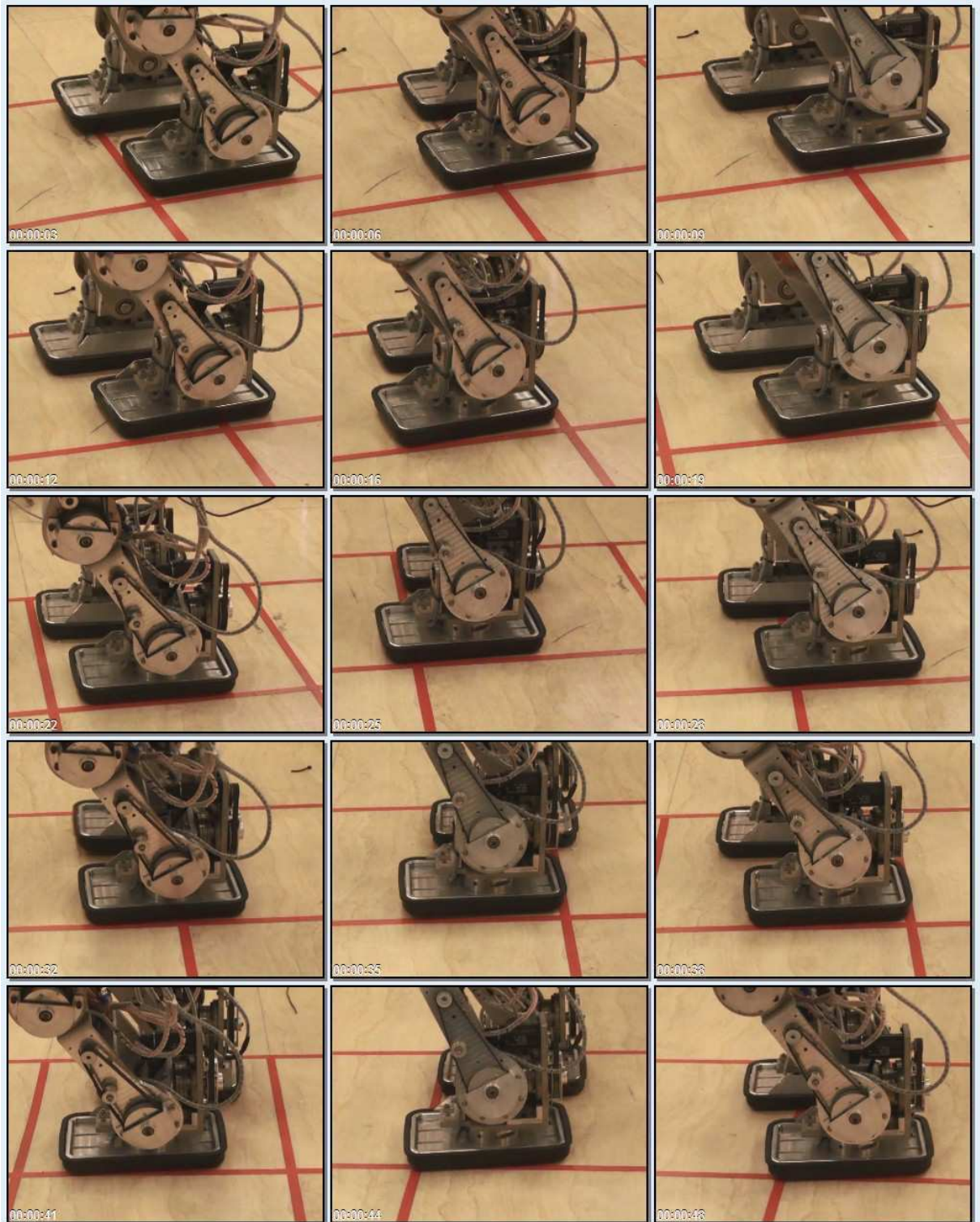
The same experiment is carried out with the yaw orientation controller turned on. The control action decided by the designed controller is shown in Figure 6.3. Because the control action is inversely proportional to the decided arc walk radius  $r_c$ ,  $1/r_c$  is plotted for visualization purposes.



**Figure 6.3.** Inverse of the arc walk radius decided by the controller

Figure 6.4 shows snapshots from a walking experiment with the yaw orientation controller turned on. The improvement is clearly visible from the floor markers.





**Figure 6.4.** SURALP walking experiment with the yaw controller turned on.

## Chapter 7

### 7. CONCLUSIONS

This thesis concentrates on the walking reference correction with visual feedback for humanoid robots. The task for the humanoid robot is to simply walk on a straight path looking at its environment. This task requires the robot to localize itself using visual data gathered from a stereo camera pair.

A visual odometry algorithm is developed using a set of real images grabbed by a wheeled robot and comparing the estimations with the ground truth trajectory provided. The proposed algorithm finds distinct corners in the environment and reconstructs their 3D positions using the stereo camera pair. Then these 3D points are tracked in time to continuously localize the robot with respect to their observed positions in the camera coordinate frame. For the camera pose estimation problem, two different methods from the literature are tried and a novel approach is proposed. Finally, a simple yaw orientation controller is designed to correct the robots walking path direction.

Walking experiments with and without the yaw orientation controller are carried out with the humanoid robot SURALP for comparison. The results show that the robot was able to correct its trajectory with the generated visual feedback.

The developed visual odometry algorithm is deemed to be a good local estimator for position and orientation. But the inevitable drift in estimation inhibits its use as a stand-alone solution for long term self-localization. The promising results suggest its potential to be used in a more sophisticated Simultaneous Localization and Mapping method utilizing multiple sensors.

The 3D reconstruction algorithm implemented as a part of this thesis' work is the first stereo vision application that has been employed for SURALP. It created a framework for other 3D vision applications which may allow the robot to perform higher level tasks in the future.

## REFERENCES

- [1] A. R. Bruss and B. K. Horn. "Passive navigation". *Computer Graphics and Image Processing*, 21:3–20, 1983.
- [2] J. H. Rieger and D.T. Lawton. "Processing differential image motion". *Journal of Optical Society America A*, 2(2):354–359, 1985.
- [3] K. Prazdny. "On the information in optical flows". *Computer Graphics and Image Processing*, 22:239–259, 1983.
- [4] D. J. Heeger and A. D. Jepson. "Subspace methods for recovering rigid motion. i. algorithm and implementation". *International Journal of Computer Vision*, 7(2):95–117, 1992.
- [5] E. C. Hildreth. "Recovering heading for visually-guided navigation". *Vision Research*, 32(6):1177–1192, 1992.
- [6] A. D. Jepson and D. J. Heeger. "A fast subspace algorithm for recovering rigid motion". In *Proceedings of IEEE Workshop on Visual Motion*, pages 124–131, Princeton, NJ, 1991.
- [7] A. D. Jepson and D. J. Heeger. "Linear subspace methods for recovering translation direction". *Spatial Vision in Humans and Robots*, pages 39–62. Cambridge University Press, New York, 1993.
- [8] C. Tomasi and J. Shi. "Direction of heading from image deformations". In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 422–427, New York, 1993.
- [9] T. Y. Yian, C. Tomasi, D. J. Heeger. "Comparison of Approaches to Egomotion Computation", In *Proceedings of CVPR'96*, 1996.
- [10] O. D. Faugeras, F. Lustman, and G. Toscani. "Motion and structure from motion from point and line matches". In *Proceedings of the First International Conference on Computer Vision*, pages 25–34, London, 1987.
- [11] H. C. Longuet-Higgins. "A computer algorithm for reconstructing a scene from two projections". *Nature*, 293(10):133–135, 1981.
- [12] J. Weng, T. S. Huang, and N. Ahuja. "Motion and structure from two perspective views: Algorithms, error analysis, and error estimation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–467, 1989.

- [13] J. Weng, N. Ahuja, and T.S. Huang. “Optimal motion and structure estimation”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):864–884, 1993.
- [14] B. Triggs; P. McLauchlan and R. Hartley and A. Fitzgibbon (1999). “Bundle Adjustment — A Modern Synthesis”. *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*. Springer-Verlag. pp. 298–372.
- [15] M. Intel, Ameller, M. Quan, and L. Triggs. “Camera pose revisited: New linear algorithms”, 2002.
- [16] R. I. Hartley and A. Zisserman. “Multiple View Geometry in Computer Vision”. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [17] F. Moreno-Noguer, V. Lepetit, and P. Fua. “Accurate Non-Iterative O(n) Solution To the PnP Problem”. In *ICCV*, Rio de Janeiro, Brazil, October 2007.
- [18] H. Alismail, B. Browning, and M. B. Dias. ”Evaluating Pose Estimation Method for Stereo Visual Odometry on Robots”. In *Proceedings of the 11th International Conference on Intelligent Autonomous Systems (IAS-11)*, 2010.
- [19] S. Umeyama. “Least-squares estimation of transformation parameters between two point patterns”. *PAMI*, 13(4):376–380, 1991.
- [20] G. Bleser, H. Wuest, D. Stricker. “Online camera pose estimation in partially known and dynamic scenes”. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 56-65. 2006.
- [21] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, “A robust visual odometry and precipice detection system using consumer-grade monocular vision”, in *IEEE International Conference on Robotics and Automation*, 2005.
- [22] M. Zaman. “High Precision Relative Localization Using a Single Camera”, 2007 *IEEE International Conference on Robotics and Automation*, pp.3908-3914, 10-14 April 2007
- [23] Sunglok Choi; Ji Hoon Joung; Wonpil Yu; Jae-Il Cho; , “What does ground tell us? Monocular visual odometry under planar motion constraint”, *2011 11th International Conference on Control, Automation and Systems (ICCAS)*., pp.1480-1485, 26-29 Oct. 2011
- [24] Nourani-Vatani, N.; Roberts, J.; Srinivasan, M.V.; , “Practical visual odometry for car-like vehicles”, *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on* , pp.3551-3557, 12-17 May 2009

- [25] N. Nourani-Vatani, J. Roberts, and M. V. Srinivasan, "IMU aided 3D visual odometry for car-like vehicles" in Australasian Conference on Robotics and Automation, 2008.
- [26] Gross, H.-M.; Koenig, A.; Boehme, H.-J.; Schroeter, Ch.; , "Vision-based Monte Carlo self-localization for a mobile service robot acting as shopping assistant in a home store" 2002. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol.1, no., pp. 256- 262 vol.1, 2002
- [27] Rofer, T.; Jungel, M.; , "Vision-based fast and reactive Monte-Carlo localization" 2003. *Proceedings. ICRA '03. IEEE International Conference on Robotics and Automation*, vol.1, no., pp. 856- 861 vol.1, 14-19 Sept. 2003
- [28] Saeedi, P.; Lawrence, P.; Lowe, D.; , "3D motion tracking of a mobile robot in a natural environment". *Proceedings. ICRA '00. IEEE International Conference on Robotics and Automation*, vol.2, pp.1682-1687, 2000
- [29] Nister, D.; Naroditsky, O.; Bergen, J.; , "Visual odometry". *CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004* , vol.1, pp. I-652- I-659, 27 June-2 July 2004
- [30] Achtelik, Markus; Bachrach, Abraham; He, Ruijie; Prentice, Samuel; Roy, Nicholas. "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments." Unmanned Systems Technology XI. Ed. Grant R. Gerhart, Douglas W. Gage, & Charles M. Shoemaker. Orlando, FL, USA: SPIE, 2009. 733219-10.
- [31] M.I.A. Lourakis and A.A. Argyros. "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm". Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from <http://www.ics.forth.gr/~lourakis/sba>.
- [32] Sunderhauf, N., Konolige, K., Lacroix, S. & Protzel, P. (2005). "Visual Odometry using Sparse Bundle Adjustment on an Autonomous Outdoor Vehicle". *Tagungsband Autonome Mobile Systeme 2005*, Reihe Informatik aktuell, Springer Verlag, pp. 157-163.
- [33] Olson, C.F.; , "Probabilistic self-localization for mobile robots", *IEEE Transactions on Robotics and Automation*, vol.16, no.1, pp.55-66, Feb 2000

- [34] Clark F. Olson, Larry H. Matthies, Marcel Schoppers, Mark W. Maimone, “Rover navigation using stereo ego-motion”, *Robotics and Autonomous Systems*, Volume 43, Issue 4, Pages 215-229, 30 June 2003.
- [35] Yang Cheng; Mark Maimone; Larry Matthies; , “Visual odometry on the Mars Exploration Rovers”, *IEEE International Conference on Systems, Man and Cybernetics, 2005*, vol.1, no., pp. 903- 910 Vol. 1, 10-12 Oct. 2005
- [36] Matthies, L.; Shafer, S.; , “Error modeling in stereo navigation” , *IEEE Journal of Robotics and Automation*, vol.3, no.3, pp.239-248, June 1987
- [37] Trebi-Ollennu, A.; Huntsberger, T.; Yang Cheng; Baumgartner, E.T.; Kennedy, B.; Schenker, P.; , “Design and analysis of a sun sensor for planetary rover absolute heading detection” , *IEEE Transactions on Robotics and Automation* , vol.17, no.6, pp.939-947, Dec 2001
- [38] R. Smith, M. Self, P. Cheeseman. “Estimating uncertain spatial relationships in robotics”. I.J. Cox, G.T. Wilfong (Eds.), *Autonomous Robot Vehncles*, Springer, Berlin (1990), pp. 167–193
- [39] R. Smith, P. Cheeseman. “On the Representation and Estimation of Spatial Uncertainty”. *The International Journal of Robotics Research* December 1986 vol. 5 no. 4 56-68
- [40] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. P. Gerkey, “Outdoor mapping and navigation using stereo vision,” in *Proc. of the Intl. Symp. on Experimental Robotics (ISER)*, July 2006.
- [41] R. Sim, P. Elinas, J. J. Little. “A Study of the Rao-Blackwellised Particle Filter for Efficient and Accurate Vision-Based SLAM”. *International Journal of Computer Vision*, vol. 74 , issue 3, pages 303-318, September 2007
- [42] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy,”Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments,” in *Robotics: Science and Systems Conference*, June 2008.
- [43] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, “Visual odometry and mapping for autonomous flight using an RGB-D camera,” in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, (Flagstaff, USA), August 2011.
- [44] Ryan M. Eustice, *Large-Area Visually Augmented Navigation for Autonomous Underwater Vehicles*, Massachusetts Institute of Technology, PhD Thesis, June, 2005.

- [45] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard, Visually Navigating the RMS Titanic with SLAM Information Filters, Proceedings of Robotics Science and Systems, June 2005.
- [46] O. Pizarro, R. Eustice, and H. Singh, Large Area 3D Reconstructions from Underwater Surveys, OCEANS 2004 MTS/IEEE Conference and Exhibition, vol 2, pp 678-687, Kobe, Japan, November 2004.
- [47] SB. Williams, I. Mahon. Simultaneous Localisation and Mapping on the Great Barrier Reef. IEEE International Conference on Robotics and Automation, April 26-May 1, 2004, New Orleans, USA.
- [48] Saez, J.M.; Hogue, A.; Escolano, F.; Jenkin, M.; , “Underwater 3D SLAM through entropy minimization”. Proceedings of IEEE International Conference on Robotics and Automation 2006. pp.3562-3567, 15-19 May 2006
- [49] Kolter, J. Zico; Youngjun Kim,; Ng, Andrew Y.; , “Stereo vision and terrain modeling for quadruped robots”. IEEE International Conference on Robotics and Automation 2009, pp.1557-1564, 12-17 May 2009
- [50] Adam Schmidt and Andrzej Kasinski. 2010. “The visual SLAM system for a hexapod robot”. In Proceedings of the 2010 International Conference on Computer Vision and Graphics: Part II (ICCVG'10), Leonard Bolc, Ryszard Tadeusiewicz, Leszek J. Chmielewski, and Konrad Wojciechowski (Eds.). Springer-Verlag, Berlin, Heidelberg, 260-267.
- [51] Durgin, F. H., Akagi, M., Gallistel, C. R., & Haiken, W. (2008). “The precision of locomotor odometry in humans”. *Experimental Brain Research*, vol. 193, pages 429–436.
- [52] T. Albright, “Cortical processing of visual motion,” in *Visual Motion and its Use in the Stabilization of Gaze*, J. Wallman and F. Miles, Eds. New York: Elsevier, 1993, ch. 9, pp. 177–201.
- [53] Pretto, A., Menegatti, E., Bennewitz, M., Burgard, W., Pagello, E. “A visual odometry framework robust to motion blur”. IEEE International Conference on Robotics and Automation 2009, pp.2250-2257, 12-17 May 2009
- [54] Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., Fujimura, K. “The intelligent ASIMO: system overview and integration”. IEEE/RSJ International Conference on Intelligent Robots and Systems 2002, vol.3, pp. 2478-2483, 2002

- [55] Tellez, R., Ferro, F., Garcia, S., Gomez, E., Jorge, E., Mora, D., Pinyol, D., Oliver, J., Torres, O., Velazquez, J., Faconti, D. "Reem-B: An autonomous lightweight human-size humanoid robot". 8th IEEE-RAS International Conference on Humanoid Robots 2008, pp.462-468, 1-3 Dec. 2008
- [56] Hirohisa Hirukawa, Fumio Kanehiro, Kenji Kaneko, Shuuji Kajita, Kiyoshi Fujiwara, Yoshihiro Kawai, Fumiaki Tomita, Shigeoki Hirai, Kazuo Tanie, Takakatsu Isozumi, Kazuhiko Akachi, Toshikazu Kawasaki, Shigehiko Ota, Kazuhiko Yokoyama, Hiroyuki Handa, Yutaro Fukase, Jun-ichiro Maeda, Yoshihiko Nakamura, Susumu Tachi, Hirochika Inoue, "Humanoid robotics platforms developed in HRP", *Robotics and Autonomous Systems*, Volume 48, Issue 4, 31 October 2004, Pages 165-175
- [57] Stasse, O., Verrelst, B., Davison, A., Mansard, N., Vanderborght, B., Esteves, C., Saidi, F., Yokoi, K. "Integrating Walking and Vision to Increase Humanoid Robot Autonomy" IEEE International Conference on Robotics and Automation 2007, pp.2772-2773, 10-14 April 2007
- [58] Olivier Stasse, Andrew J. Davison, Ramzi Sellaouti, Kazuhito Yokoi. "Real-time 3D SLAM for Humanoid Robot considering Pattern Generator Information," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp.348-355, Oct. 2006
- [59] Kwak, N., Stasse, O., Foissotte, T., Yokoi, K. "3D grid and particle based SLAM for a humanoid robot". 9th IEEE-RAS International Conference on Humanoid Robots, pp.62-67, 7-10 Dec. 2009
- [60] Nishiwaki, K., Kuffner, J., Kagami, S., Inaba, M., & Inoue, H. "The experimental humanoid robot H7: A research platform for autonomous behaviour". *Philosophical Transaction A: Mathematical Physical and Engineering Sciences*, vol. 365, pp. 79–107, 2007.
- [61] Thompson, S., Kagami, S. "Humanoid robot localisation using stereo vision" 5th IEEE-RAS International Conference on Humanoid Robots, pp.19-25, 2005
- [62] Ozawa, R., Takaoka, Y., Kida, Y., Nishiwaki, K., Chestnutt, J., Kuffner, J., Kagami, J., Mizoguch, H., Inoue, H. "Using visual odometry to create 3D maps for online footstep planning". 2005 IEEE International Conference on Systems, Man and Cybernetics, vol.3, pp. 2643- 2648, 10-12 Oct. 2005



- [63] Takaoka, Y., Kida, Y., Kagami, S., Mizoguchi, H., Kanade, T. "3D map building for a humanoid robot by using visual odometry". IEEE International Conference on Systems, Man and Cybernetics 2004, vol.5, pp. 4444- 4449, 10-13 Oct. 2004
- [64] Jianbo Shi, Tomasi, C. "Good features to track". Proceedings of 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp.593-600, 21-23 Jun 1994
- [65] B. D. Lucas and T. Kanade. "An iterative image registration technique with an application to stereo vision". IJCAI, 1981.
- [66] Erbatur, K., U. Seven, E. Taşkıran, Ö. Koca, M. Yılmaz, G. Kızıldaş, M. Ünel, A. Sabanovic, A. Onat, "SURALP: A New Full-Body Humanoid Robot Platform", IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, St. Louis, MO, USA, October 2009.
- [67] Kajita, F. Kahehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, H. Hirukawa, " Biped walking pattern generation using preview control of the zero-moment-point", Proceedings of IEEE International Conference on Robotics and Automation, pp: 1620 - 1626, vol.2, Taipei, Taiwan, September 2003.
- [68] Y. Choi, B. J. You, and S. R. Oh, "On the stability of indirect ZMP controller for biped robot systems", Proceedings of International Conferenc on Intelligent Robots and Systems, pp: 1966 - 1971, vol.2, Sendai, Japan, June 2004.
- [69] Erbatur, K. and Kurt, O., "Natural ZMP Trajectories for Biped Robot Reference Generation", IEEE Transactions on Industrial Electronics, vol. 56, no. 3, pp. 835-845, March 2009.
- [70] Erbatur, K., O. Koca, E. Taskiran, M. Yılmaz and U. Seven, "ZMP Based Reference Generation for Biped Walking Robots," presented in International Conference on Intelligent Control, Robotics, and Automation, ICICRA 2009, Venice, Italy October 28-30, 2009, published in World Academy of Science Engineering and Technology, Vol. 58, pp. 546-553, October 2009.
- [71] Taskiran, E., M. Yılmaz, O. Koca, U. Seven and K. Erbatur, "Trajectory Generation with Natural ZMP References for the Biped Walking Robot SURALP," Proc. 2010 IEEE International Conference on Robotics and Automation, ICRA 2010, Alaska, USA.
- [72] Brown DC "Decentering distortion of lenses.". Photogrammetric Engineering, vol. 7, pp. 444-462, 1966.

- [73] hengyou Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations," IEEE International Conference on Computer Vision, vol. 1, p. 666 , 1999
- [74] J.Y.Bouguet. "MATLAB calibration tool". [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)
- [75] G. Bradski and A. Kaehler. "Learning OpenCV", Sep. 2008
- [76] Hartley, Richard I. "Theory and Practice of Projective Rectification". International Journal of Computer Vision, vol. 35, pp. 115-127, 1999.
- [77] C. Harris and M. Stephens. "A combined corner and edge detector", in Proc. Alvey Conf., pp. 147–151, 1988
- [78] Lowe, D.G. "Object recognition from local scale-invariant features". The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol.2, pp.1150-1157, 1999
- [79] H. Bay, T. Tuytelaars, L. Van Gool, "SURF: speeded up robust features", in ECCV, 2006.
- [80] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection". In Proc. 9th European Conference on Computer Vision (ECCV'06), Graz, May 2006.
- [81] Andrea Bonarini, Wolfram Burgard, Giulio Fontana, Matteo Matteucci, Domenico Giorgio Sorrenti and Juan Domingo Tardos. "RAWSEEDS: Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets". In proceedings of IROS'06 Workshop on Benchmarks in Robotics Research, 2006.
- [82] J. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm," OpenCV Document, Intel, Microprocessor Research Labs, 2000.
- [83] Jochen Schmidt, Heinrich Niemann: "Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization". VMV 2001, pp.399-406.

## APPENDIX A

### A. THE LEVENBERG-MARQUARDT METHOD

Levenberg-Marquardt method is an iterative, hybrid optimization method which seamlessly moves between the Newton's method and a form of Gradient Descent. This allows the solution to converge much faster than the Gradient Descent, and prevents it from diverging where the Hessian is not "well-behaving", as Newton's method would.

Consider a generic nonlinear least squares problem given a set of observation pairs  $x_1 \dots x_N$  and  $y_1 \dots y_N$  and a nonlinear model curve  $f(x_i, \boldsymbol{\beta})$  relating them, the problem of finding the parameter vector  $\boldsymbol{\beta}$  minimizing the least squared error is stated as

$$\operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^N \|y_i - f(x_i, \boldsymbol{\beta})\|^2. \quad (\text{A.1})$$

At each iteration of the Levenberg-Marquardt method,  $\boldsymbol{\delta}$ , an update to the parameter vector  $\boldsymbol{\beta}$  is calculated. Defining a cost function  $S(\boldsymbol{\beta})$  as

$$S(\boldsymbol{\beta}) = \sum_{i=1}^N \|y_i - f(x_i, \boldsymbol{\beta})\|^2 \quad (\text{A.2})$$

or equivalently

$$S(\boldsymbol{\beta}) = \|\mathbf{Y} - \mathbf{F}\|^2 \quad (\text{A.3})$$

where

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad (\text{A.4})$$

$$\mathbf{F} = \begin{bmatrix} f(x_1, \boldsymbol{\beta}) \\ \vdots \\ f(x_N, \boldsymbol{\beta}) \end{bmatrix}. \quad (\text{A.5})$$

Then  $\boldsymbol{\delta}$  that would decrease  $S(\boldsymbol{\beta} + \boldsymbol{\delta})$  is calculated from the following equation

$$(\mathbf{J}^T \mathbf{J} + \xi \text{diag}(\mathbf{J}^T \mathbf{J})) \boldsymbol{\delta} = \mathbf{J}^T (\mathbf{Y} - \mathbf{F}) \quad (\text{A.6})$$

Where  $\mathbf{J}$  is the Jacobian matrix obtained from

$$\mathbf{J} = \frac{\partial \mathbf{F}}{\partial \boldsymbol{\beta}} \quad (\text{A.7})$$

This Jacobian matrix can be obtained by taking the derivative analytically, or performing discrete differentiation numerically. Although the analytic option would possibly perform faster as it is non-iterative, it may not always be possible or practical to get.

The pseudo-code for Levenberg-Marquardt algorithm is given in Table A.1.

**Table A.1.** Pseudo-code for Levenberg-Marquardt algorithm

---

```
Levenberg_Marquardt(x, y, beta, ksi_growth, ksi_max)
iter ← 0
converged ← false
failed ← false
error_old ← infinity
ksi ← small number (e.g. 0.00001)
Y ← stack y on top of each other
while converged = false
    iter ← iter + 1
    F ← compute f for all x using beta, stack them
    if iter = max_iter
        converged ← true
        failed ← true, could not converge in given iterations
    else
        error_decreased ← false
        J ← compute J from F and beta (analytic or numerical)
        while error_decreased = false
            delta ← perform equation A.6
            beta_new ← beta + delta
            error ← compute the cost function with beta_new
            if error < error_old
                error_decreased ← true
                beta ← beta_new
                ksi ← ksi/ksi_growth
            else
                ksi ← ksi*ksi_growth

            if ksi > ksi_max // error cannot be decreased
                converged ← true
                break error_decreased iteration
        return !failed
```

---

## APPENDIX B

### B. RANSAC

RANSAC is a robust estimation method that aims to fit a model to data possibly containing gross outliers. The pseudo-code for a generic RANSAC is given in Table B.1.

The parameters of the algorithm are the minimum number of data needed to fit a model ( $n$ ), number of iterations to be performed ( $\text{max\_iter}$ ), a threshold to decide whether model fits a datum ( $\text{epsilon}$ ), minimum number of inliers needed to establish a hypothesis model ( $\text{minN}$ ).

**Table B.1.** RANSAC pseudo-code

---

```
RANSAC(data, n, max_iter, epsilon, minN)
  iter ← 0
  best_model ← null
  best_error ← infinity

  while iter < max_iter
    consensus ← null
    random_data ← n randomly selected data points
    model ← fit a model to random_data
    for each point in data
      error ← calculate error with model
      if error < epsilon
        consensus ← add point to consensus
    if consensus size > minN
      consensus_model ← fit a model again to all the
```

---

---

```
points in the consensus
    consensus_error ← calculate the total error in the
consensus with the consensus_model
    if consensus_error < best_error
        best_model ← consensus_model
        best_error ← consensus_error
    iter ← iter + 1
return best_model
```

---