EXPERIMENTAL VALIDATION OF SURVEILLANCE CAMERA AS A VIBRATION SENSOR

YUECHENG SHAO

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

BOSTON

UNIVERSITY

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Thesis

EXPERIMENTAL VALIDATION OF SURVEILLANCE CAMERA AS A VIBRATION SENSOR

by

YUECHENG SHAO

B.S., Fudan University, 2009

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science
2011

Approved by

First Reader

Janusz Konrad, Ph.D.
Professor of Electrical and Computer Engineering

Second Reader

Prakash Ishwar, Ph.D.
Assistant Professor of Electrical and Computer Engineering

Acknowledgments

First and foremost I would like to thank my advisor Professor Janusz Konrad for his guidance and encouragement through this research. He always answers my questions very quickly and points out what should I improve during the progress of the research. Since this is my first thesis in English, he has spent a lot of time revising it although he is extremely busy. I have learnt how to be a good researcher from him.

In addition, I would like to thank Meng Wang, a PhD student and also a very good friend, for his suggestions and great ideas. I would also like to extend my thanks to all my friends in Boston University.

Finally, I would like to thank my committee Professor Janusz Konrad and Professor Prakash Ishwar for reviewing this thesis and providing valuable comments. It is my great pleasure to work with you.

EXPERIMENTAL VALIDATION OF SURVEILLANCE CAMERA AS A VIBRATION SENSOR

YUECHENG SHAO

ABSTRACT

In the last decade, surveillance video cameras have become ubiquitous in city centers, at transportation hubs, and along highways. Today, there are well over 30 million surveillance cameras in the US, all predominantly focused on security-related tasks. This high urban density of visual sensors, that are only expected to increase in the future, can potentially be leveraged for goals other than those related to security-related goals. A video camera, especially at a long zoom setting, is an excellent vibration sensor; as the camera shakes, the captured video undergoes jitter. If camera vibrations can be accurately characterized based on the captured jitter, applications such as earthquake early warning, building "tomography", or bridge stability testing can be envisaged in the future. This thesis attempts to validate the feasibility of using a surveillance camera as a vibration sensor. More specifically, the goal is to estimate small 3D translations and rotations of a camera and to compare them with data captured by an accelerometer attached to the camera. This comparison is conducted in a laboratory setting using planar, easy-to-track targets (chessboard patterns) under controlled lighting conditions. The homographies between consecutive video frames are estimated using feature-point correspondences, and then decomposed into 3D translations and rotations using a method proposed by Faugeras and Lustman. First, the algorithm is validated on synthetic data (ground truth) and tested for its robustness to uncertainty in localization of feature points. Then, the algorithm is applied to camera-captured data. The free 2-D movements of the camera on a level table are compared with the measurements of an accelerometer. The resulting normalized correlation factors between the camera and the accelerometer translation waveforms suggest that in a laboratory setting a camera can serve as a surrogate for an accelerometer.

Contents

1 Prior work **1**

2 Background material **6**

2.1 Pinhole camera model in Euclidean space 6

2.2 Homogeneous coordinates 7

2.3 Pinhole camera model in projective space 10

2.4 Homography 13

2.4.1 Definition 13

2.4.2 Estimation of homography using the Direct Linear Transform 15

3 Phase correlation **18**

4 Camera calibration method of Zhang **25**

5 Homography decomposition method of Faugeras and Lustman **38**

5.1 Case #1: $\lambda_1 \neq \lambda_2 \neq \lambda_3$ ($\lambda_1 > \lambda_2 > \lambda_3$) 43

5.2 Case #2: $\lambda_1 = \lambda_2 \neq \lambda_3$ ($\lambda_1 = \lambda_2 > \lambda_3$) or $\lambda_1 \neq \lambda_2 = \lambda_3$ ($\lambda_1 > \lambda_2 = \lambda_3$) 45

5.3 Case #3: $\lambda_1 = \lambda_2 = \lambda_3$ 46

5.4 Summary of solutions 46

5.5 Implementation of the method of Faugeras and Lustman 49

6 Experimental results **52**

6.1 Synthetic data experiments 52

6.2 “Ground-truth” experiments 73

6.2.1 Table-top experiments 73

    6.2.2 Platform-based experiments 89
  6.3 Accelerometer-based experiments 99
    6.3.1 Table-top experiments 102
    6.3.2 Platform-based experiments 102

7 Conclusions **107**
  7.1 Discussion of results 107
  7.2 Future work 108

References **110**

Curriculum Vitae **111**

List of Figures

1·1 Examples of augmented reality in (a) gaming and (b) broadcasting. . 2

2·1 Pinhole camera geometry in (a) 3D view and (b) side-view. The image plane is unrealistically located between the optical center and the object in order to forgo the ”-” sign in equations (2.1). 7

2·2 Example of projective geometry. Railroad track gets narrower and the rails meet at horizon. 8

2·3 Finding the direction of the cross product by the right-hand rule . . . 9

2·4 Image (x, y) and camera $(X_{cam}, Y_{cam}, Z_{cam})$ coordinate systems (Z_{cam} axis is not shown here and it's perpendicular to the image plane). . . 11

2·5 Transformation $\mathbf{R}$, $\vec{t}$ between the world coordinate system and the camera coordinate system. Three coordinate systems are shown: the camera coordinate system $(X_{cam}, Y_{cam}, Z_{cam})$, the world coordinate system $(X_{world}, Y_{world}, Z_{world})$, and the image coordinate system (x, y). . 11

2·6 A scene captured from five different orientations of a camera. 14

2·7 Alignment of five images from Figure 2·6 using homographies. 15

3·1 Two images corrupted by iid Gaussian noise. The right image is translated by (30, 33) pixels relative to the left image. 19

3·2 The result of phase correlation applied to two images from Figure 3·1 19

3·3 (a) A synthetic image and (b) its shifted version (by (6,15)). 21

3·4 Correlation surface $r(x, y)$ computed using equations (3.1) and (3.2) between images from Figure 3·3. 21

3·5 Two video frames captured by a vibrating camera. 24
3·6 Error between images in Figure 3·5: (a) without and (b) with motion compensation. 24
3·7 (a) Horizontal (x direction) and (b) vertical (y direction) displacement estimated from video shown in Figure 3·5 using phase correlation. . 24

4·1 Consecutive frames of a video captured by camera moving parallel to the target (chessboard pattern). 29
4·2 Consecutive frames of a video captured by camera (vertically flipped) moving perpendicular to the target (chessboard pattern). 29
4·3 Geometric interpretation of experimental results for camera motion parallel to the target (camera-centered view). 32
4·4 Geometric interpretation of experimental results for camera motion parallel to the target (world-centered view). 33
4·5 Geometric interpretation of experimental results for camera motion perpendicular to the target (camera-centered view). 34
4·6 Geometric interpretation of experimental results for camera motion perpendicular to the target (world-centered view). 35
4·7 (a) Frontal view and (b) non-frontal view of feature points on the chessboard pattern 36

5·1 The relationship between object plane (chessboard pattern), the first camera coordinate system $(X_{cam1}, Y_{cam1}, Z_{cam1})$ and the second camera coordinate system $(X_{cam2}, Y_{cam2}, Z_{cam2})$. 39

6·1 Synthetic lattice with the camera coordinate system aligned to the world coordinate system. 53

6·2 Translation of the synthetic lattice along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (1000k, 0, 0)^T$. 54

6·3 Translation of the synthetic lattice along y axis of the world coordinate system by $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 1000k, 0)^T$. 55

6·4 Translation of the synthetic lattice along z axis of the world coordinate system by $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 0, 1000k)^T$. 55

6·5 Rotation of the synthetic lattice around x axis of the world coordinate system by $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (k-5, 0, 0)^T$. 56

6·6 Rotation of the synthetic lattice around y axis of the world coordinate system by $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (0, k-5, 0)^T$. 56

6·7 Rotation of the synthetic lattice around z axis of the world coordinate system by $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (0, 0, k-5)^T$. 57

6·8 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (1000k, 0, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, 0)^T$ in case 1. 58

6·9 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 1000k, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, 0)^T$ in case 2. 59

6·10 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 0, 1000k)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, 0)^T$ in case 3. 60

6·11 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 0, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (k-5, 0, 0)^T$ in case 4. 61

6·12 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 0, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, k-5, 0)^T$ in case 5. 62

6·13 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 0, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, k-5)^T$ in case 6. 63

6·14 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ for arbitrary camera motion in case 7. 64

6·15 Root-mean-squared error of translation in x direction. 67

6·16 Root-mean-squared error of translation in y direction. 68

6·17 Root-mean-squared error of translation in z direction. 68

6·18 Root-mean-squared error of rotation around x axis. 69

6·19 Root-mean-squared error of rotation around y axis. 69

6·20 Root-mean-squared error of rotation around z axis. 70

6·21 Root-mean-squared error of translation in x direction as a function of assumed focal length f (500 is the ground truth). 70

6·22 Root-mean-squared error of translation in y direction as a function of assumed focal length f (500 is the ground truth). 71

6·23 Root-mean-squared error of translation in z direction as a function of assumed focal length f (500 is the ground truth). 71

6·24 Root-mean-squared error of rotation around x axis as a function of assumed focal length f (500 is the ground truth). 72

6·25 Root-mean-squared error of rotation around y axis as a function of assumed focal length f (500 is the ground truth). 72

6·26 Root-mean-squared error of rotation around z axis as a function of assumed focal length f (500 is the ground truth). 73

6·27 Camera setup for real-data experiments 74

6·28 Translation of the camera along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$ in case 1. 76

6·29 Translation of the camera along z axis of the world coordinate system by $\vec{t}_g^{(k)} = (0, 0, -k)^T$ in case 2. 76

6·30 Rotation around x-axis by angle β_1 followed by translation of the camera along z axis of the world coordinate system by $\vec{t}_g^{(k)} = (0, 0, -k)^T$ in case 3. 77

6·31 Rotation around y-axis by angle β_2 followed by translation of the camera along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$ case 4. 77

6·32 Rotation around z-axis by angle β_3 followed by translation of the camera along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$ in case 5. 78

6·33 Camera movements in case 6. 78

6·34 Sample frames from a video captured in case 1. 79

6·35 Sample frames from a video captured in case 2. 79

6·36 Sample frames from a video captured in case 3. 80

6·37 Sample frames from a video captured in case 4. 80

6·38 Sample frames from a video captured in case 5. 81

6·39 Sample frames from a video captured in case 6. 81

6·40 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 1. 82

6·41 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 2. 83

6·42 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 3. 84

6·43 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 4. 85

6·44 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 5. 86

6·45 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 6. 87

6·46 Mechanical platform developed to allow precise camera translations and rotations. 90

6·47 Sample frames from a video captured while moving the camera along x-axis. 91

6·48 Sample frames from a video captured while moving the camera along z-axis. 91

6·49 Sample frames from a video captured while rotating the camera around x-axis. 92

6·50 Sample frames from a video captured while rotating the camera around y-axis. 92

6·51 Sample frames from a video captured while rotating the camera around z-axis. 93

6·52 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera moved along x-axis 94

6·53 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera moved along z-axis 95

6·54 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera rotated around its x-axis 96

6·55 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera rotated around its y-axis 97

6·56 Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera rotated around its z-axis 98

6·57 (a) Acceleration measured by accelerometer. (b) Velocity and (c) translation computed by integration and double integration, respectively, of acceleration from (a). Note the drift in velocity and translation caused by acceleration bias (see text for discussion). (e) Velocity and (e) translation obtained by integration after compensating the bias in acceleration. Cleraly, the drift has been significantly reduced. 101

6·58 Velocities along the x-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method). Normalized cross-correlation: 0.4773. . . 103

6·59 Velocities along the z-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method). Normalized cross-correlation: 0.9138. . . 103

6·60 Velocities along the x-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (phase correlation). Normalized cross-correlation: 0.3001. 104

6·61 Velocities along the x-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method tested on calibrated platform). Normalized cross-correlation: 0.9212. 105

6·62 Velocities along the z-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method tested on calibrated platform). Normalized cross-correlation: 0.8921. 106

Chapter 1

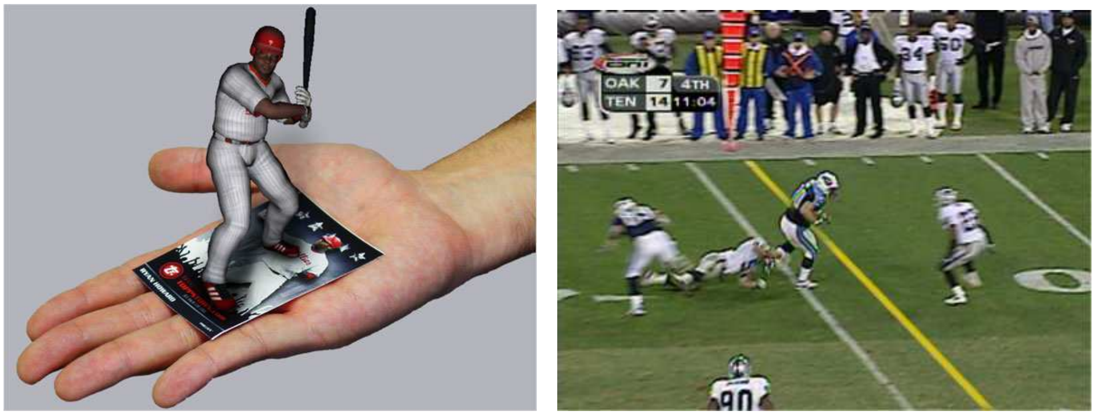
Prior work

3D motion estimation has been studied extensively in video coding, camera jitter compensation, autonomous navigation, robotic control, augmented reality and 3D scene reconstruction. Since motion of an object is interchangeable with motion of a camera, for example moving an object to the right is equivalent to moving the camera to the left, as long as we know the motion of one of them and their relative position, we can compute the motion of the other one. Therefore, we consider both methods targeting 3D camera motion and those that focus on 3D object motion.

The application of 3D motion estimation mentioned above can be categorized into two major groups. In the first group are methods that are concerned with the impact of 3D motion on images or videos, but are not concerned with the 3D motion by itself. Imagine recording a video using a hand-held camera. It is very likely that the video will be jittery due to the fact that operator's hand is shaking or he/she is walking during the recording. In order to cancel this jittering effect, first we have to estimate the jitter, or global motion, from frame to frame. By knowing how frames evolve over time, we can compensate those motions to stabilize the video. This kind of motion estimation is typically done based on a global affine model, which is also used in video coding. In video coding, rather than transmitting every frame, motion information (local or global), i.e., the direction of maximum data correlation, is transmitted.. By doing so, one is able to dramatically reduce bit rates.

In the second group are methods that are explicitly concerned with the analysis of

3D motion of objects in the real world. Typically, 3D models of objects are computed before estimating 3D motion. For example, in camera-based robotic control, in order to give robot a correct command, we have to know where the robot is in the real world and where we are going to move the robot. This requires understanding of the scene, i.e., the location and shape of objects in the world or camera coordinate system. Then, using properties of projection we are able to estimate 3D motion from 2D images. Augmented reality (Figure 1·1) is another recent application of 3D reconstruction, especially in sports telecasting. The yellow “first down” line seen in television broadcasts of American football games shows the line the offensive team must cross to receive a first down (Figure 1·1). The real-world elements are the football field and players, and the virtual element is the yellow line, which augments the image in real time. In order to make sure this yellow line is always shown where it should be during the camera panning, tilting and zooming, the 3D model of the field must be computed prior to the game.



(a) (b)

Figure 1·1: Examples of augmented reality in (a) gaming and (b) broadcasting.

First, we discuss motion estimation techniques in the first group. One technique to estimate global motion is to use a perspective (8-parameter) model for camera motion and minimize the sum of square differences (SSD) between the current frame and the motion-compensated previous frame (Dufaux and Konrad, 2000). A hierarchical algorithm is introduced which first computes a coarse estimate of the translation component between the frames by 3-step matching (Koga et al., 1981), and then iterates a gradient descent to minimize the sum of squared differences to obtain and refine 8 parameters of the perspective model. The perspective model is suitable when the scene can be approximated by a planar surface, or when the scene is static and the camera motion is a pure rotation around its optical center. This model accurately representates homography in Euclidean space, which transforms an image to another image in projective space. The 8 parameters of the perspective model or a homography matrix contain 3D motion information (translation and rotation) of a camera but they don't automatically tell us what the translation and rotation are. Thus, we have to find a way to extract 3D translation and rotation from them.

Another technique is based on the use of focus of expansion (Burger and Bhanu, 1994). It allows one to estimate 3D motion of a camera directly from 2D images without computing the 3D scene structure. By computing optical flow from video captured by a camera moving towards or away from an object, one can identify a point or small region (due to estimation errors), which is called the focus of expansion. From the location of the focus of expansion, one can estimate the 3D motion of the camera. Unfortunately, computing the precise location of the focus of expansion is difficult in real-life scenarios due to the impact of camera rotations, noisy optical flow vectors, and spatial discretization errors.

Another technique in the first group is phase correlation that can estimate displacement between two similar images. The main idea is to rely on the shift property of the Fourier transform. First, one calculates the discrete 2D Fourier transform of both images to obtain the normalized cross-power spectrum. Then, one applies the inverse Fourier transform to obtain the so-called correlation surface. By finding a peak in the correlation surface, one can estimate translation between these two images. We will discuss this method in detail in Chapter 3. Phase correlation is a very simple and effective way to calculate displacement between two images, but its performance depends on the type of 3D motion observed. The method works well only in very restricted scenarios such as images undergoing small camera pan and tilt. It fails under camera rotation and zoom.

The second group of methods, we are primarily interested in, deals with 3D motion of objects in the real world. First, we consider a popular camera calibration method (Zhang, 98). It requires the camera to observe a planar pattern shown at a few different orientations. The method offers a closed-form solution, followed by a nonlinear refinement based on the maximum likelihood criterion. The details will be discussed in Chapter 4. We implemented this method and tested it in a laboratory environment. We obtained very good results using this method on a chessboard pattern, but we abandoned it because it requires a precise specification of the pattern used. In particular, the user needs to provide coordinates of feature points in the real-world coordinate system (3D), which is impractical in our scenario of thousands of surveillance cameras.

None of the papers discussed so far propose a suitable technique for our goal, namely to precisely estimate camera motion using single camera viewing a planar object, without prior knowledge of scene depth and with only a limited knowledge of intrinsic camera parameters. An ideal method would:

1. precisely estimate 3D translation and 3D rotation of a camera.
2. require no or minimal knowledge of the scene.
3. require no camera calibration.

In short, what we are looking for is a method that can estimate 3D translation and 3D rotation from two images taken by single camera targeting a static scene in different positions without pre-calibration of the camera.

Given these constraints we seleted a technique that decomposes a homography into 3D translations and 3D rotations (Faugeras and Lustman, 1988). The authors of the method prove that it is possible to recover 3D translations and 3D rotations from a provided homography. A homography can be obtained when the scene can be approximated by a planar surface, or when the scene is static and the camera motion is a pure rotation around its optical center. This is acceptable since, in practice, we can find planar objects such as floors, walls, ceilings, roads etc., in the field of view of surveillance cameras. If we cannot find a planar surface in the field of view, we argue that it is acceptable to assume the camera is only rotating around its optical center, if the vibrations are small enough that the effects of translation and rotation on the images are almost the same. Details of this method will be discussed in Chapter 5, where we will derive all equations needed.

Chapter 2

Background material

In order to pose the problem of 3D motion estimation and describe some of methods used to solve it, we need to introduce several definitions, such as the pinhole camera model, homogeneous coordinates and homography. We will first introduce the pinhole camera model in Euclidean space, then the concept of homogeneous coordinates, then turn to the pinhole camera model in projective space, and finally we will introduce homography.

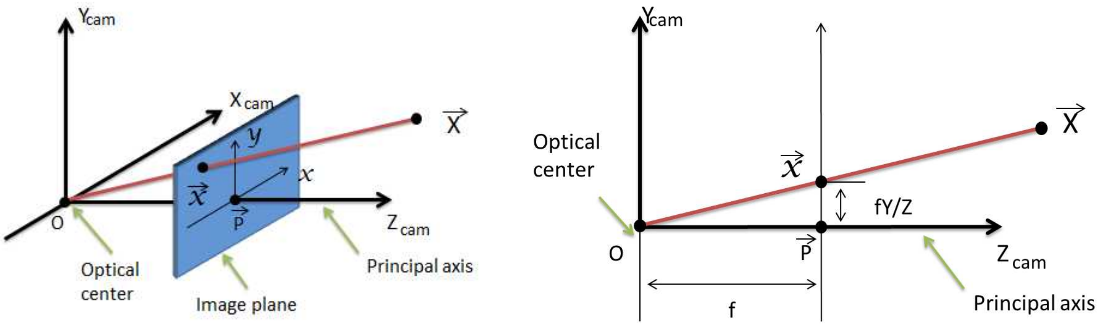
2.1 Pinhole camera model in Euclidean space

The pinhole camera model describes the relationship between a 3D scene and the corresponding 2D image. By applying the pinhole camera model, we project points from a 3D coordinate system onto a 2D coordinate system. If we denote a point in the 3D coordinate system as $\vec{X} = (X, Y, Z)^T$ and in the 2D coordinate system as $\vec{x} = (x, y)^T$, the relationship between them, under a pinhole camera model, is (Figure 2·1)

$$\begin{cases} \frac{x}{X} = \frac{f}{Z} & \Rightarrow \quad x = \frac{fX}{Z} \\ \\ \frac{y}{Y} = \frac{f}{Z} & \Rightarrow \quad y = \frac{fY}{Z} \end{cases} \tag{2.1}$$

where f is the focal length of the camera.

As we can see from equation (2.1), coordinates of the projection of a point are proportional to the focal length and inversely proportional to the distance of the 3D point from the X_{cam} - Y_{cam} plane. Figure 2·1 illustrates the pinhole camera geometry.



(a) (b)

Figure 2·1: Pinhole camera geometry in (a) 3D view and (b) side-view. The image plane is unrealistically located between the optical center and the object in order to forgo the "-" sign in equations (2.1).

Note that point $\vec{P}$ is the intersection of the image plane with the principal axis, and is called the principal point. We will return to the pinhole camera model after introducing homogeneous coordinates in the next section. Then, will interpret this model using homogeneous coordinates.

2.2 Homogeneous coordinates

In Euclidean space, it is known that two parallel lines on the same plane cannot intercept. However, it is not true any more in a projective space. For example, railroad tracks become narrower as the distance increases (Figure 2·2). The two parallel rails meet at the horizon, forming a point at infinity. The Euclidean space describes 2D/3D geometry quite well, but cannot handle projective effects. Indeed, Euclidean geometry is a subset of projective geometry.

Points at infinity are very unique points that we want to distinguish from finite points. But it is difficult to distinguish projections of 3D infinite points from projections of 3D finite points in 2D Euclidean space. In addition, equations (2.1) form a nonlinear system that is difficult to solve for x and y. Therefore, the question is

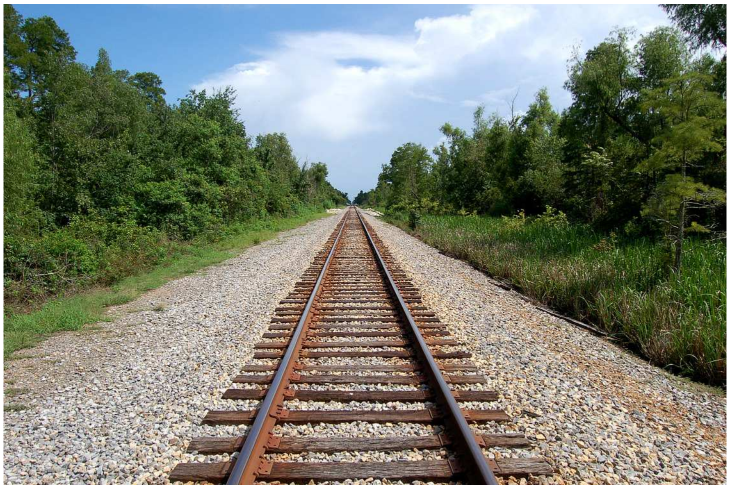


Figure 2·2: Example of projective geometry. Railroad track gets narrower and the rails meet at horizon.

whether we can find a linear representation of the pinhole camera model. Mathematicians have thought about these issues long time ago.

Homogeneous coordinates, introduced by August Ferdinand Möbius, make geometric calculations possible in projective space. At the highest level, one can thinks of homogeneous coordinates as representing N-dimensional coordinates with $N+1$ numbers.

A point in 2D Euclidean space is represented by two coordinates: $(x, y)^T$. We may define the same point by adding an extra coordinate to this pair, such as $(x, y, 1)^T$. One can multiply these three coordinates by any non-zero value k to obtain $(kx, ky, k)^T$ which represents the same point on the image plane due to the projection model our eye or camera we use. These are called the homogeneous coordinates of a point. A point with homogeneous coordinates $\vec{x} = (x_1, x_2, x_3)^T$ in 3D projective space corresponds to a point in 2D Euclidean space with coordinates $(x_1/x_3, x_2/x_3)^T$.

A line in 2D Euclidean space can also be represented in 3D projective space using homogeneous coordinates. Consider a line represented by an equation, such as

$ax+by+c=0$, in which a, b and c are line's three parameters. Thus, we can represent a line in 2D Euclidean space by a 3D vector $(a,b,c)^T$. Clearly lines $ax+by+c=0$ and $(ka)x+(kb)y+(kc)=0$ are indistinguishable for any non-zero k, and are represented by homogeneous coordinates $(a,b,c)^T$ and $k(a,b,c)^T$.

There are three results concerning the relationship between homogeneous points and lines that we will use later:

Result 1: A point $\vec{x}$ lies on the line $\vec{l}$ *if and only if* $\vec{x}^T\vec{l}=0$.

Result 2: The intersection of two lines $\vec{l}$ and $\vec{l'}$ is the point $\vec{x}=\vec{l}\times\vec{l'}$.

Result 3: The line through two points $\vec{x}$ and $\vec{x'}$ is $\vec{l}=\vec{x}\times\vec{x'}$.

The operation "×" between vectors is called the cross product and is defined by the formula: $\vec{a}\times\vec{b}=\|a\|\|b\|\vec{n}\sin\theta$, where $\vec{a}$ and $\vec{b}$ are vectors, $\|a\|$ and $\|b\|$ are norms of those vectors, $\vec{n}$ is a unit vector perpendicular to the plane containing $\vec{a}$ and $\vec{b}$ in the direction given by the right-hand rule as illustrated in Figure 2·3 and θ is the measure of the smaller angle between $\vec{a}$ and $\vec{b}$ $(0^\circ \le \theta \le 180^\circ)$.

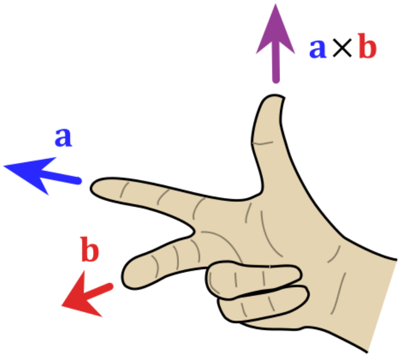


Figure 2·3: Finding the direction of the cross product by the right-hand rule

The homogeneous representation of points and lines in 3D Euclidean space is a straightforward generalization of those quantities in 2D Euclidean space, and we will not discuss them in detail here.

2.3 Pinhole camera model in projective space

If the world and image points are represented by homogeneous vectors, then the pinhole camera model can be simply expressed as a linear mapping between their homogeneous coordinates. In particular, equations (2.1) may be written in terms of matrix multiplication as follows:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.2}$$

The matrix in this expression may be written as $diag(f, f, 1)[\mathbf{I}|\vec{0}]$ where $diag(f, f, 1)$ is a diagonal matrix and $[\mathbf{I}|\vec{0}]$ represents a matrix divided up into a 3×3 block (the identity matrix) plus a column vector, here the zero vector.

The expression (2.2) assumes the origin of coordinates of the image plane is at the principal point $\vec{P}$ (Figure 2·1). However, it may not be the case in practice. The origin of coordinates of the image plane is more likely to be at the left top or left bottom corner of an image as shown in Figure 2·4. Thus, the mapping $(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T$ becomes $(X, Y, Z)^T \mapsto (fX/Z + u_0, fY/Z + v_0)^T$, where (u_0, v_0) are the coordinates of the principal point in the image coordinate system.

If we use matrix notation, (2.2) becomes

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zu_0 \\ fY + Zv_0 \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

We would like to emphasize that the point (X, Y, Z) is expressed in the camera coordinate system rather than the world coordinate system. So what if an object is described in the world coordinate system which is different from the camera coordinate system and we want to apply the pinhole camera model to it?

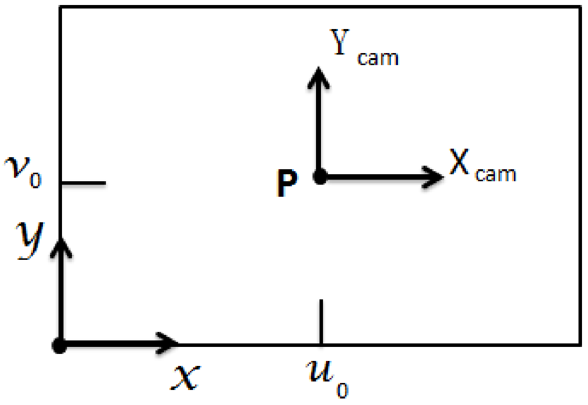


Figure 2·4: Image (x, y) and camera $(X_{cam}, Y_{cam}, Z_{cam})$ coordinate systems (Z_{cam} axis is not shown here and it's perpendicular to the image plane).

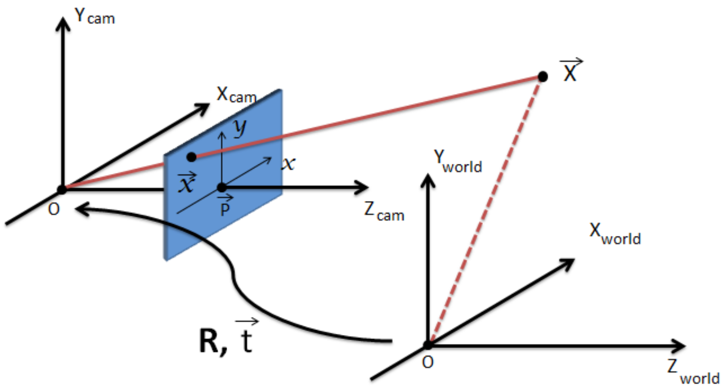


Figure 2·5: Transformation $\mathbf{R}$, $\vec{t}$ between the world coordinate system and the camera coordinate system. Three coordinate systems are shown: the camera coordinate system $(X_{cam}, Y_{cam}, Z_{cam})$, the world coordinate system $(X_{world}, Y_{world}, Z_{world})$, and the image coordinate system (x, y).

We have to transform the world coordinate system into the camera coordinate system; they are related by rotation and translation (Figure 2·5). Let's consider a point $(\hat{X}, \hat{Y}, \hat{Z})$ expressed in the world coordinate system. We need to apply the rotation and translation first, which in homogeneous coordinates can be expressed as follows:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & \vec{t} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{pmatrix} \tag{2.3}$$

where $\mathbf{R}$ is a 3×3 3D rotation matrix and $\vec{t}$ is a 3D translation vector. Then, we can project (X, Y, Z), which is expressed in the camera coordinate system, onto the image plane.

For convenience, let's change our previous notation to the conventional notation of the central projection:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} f_x & \gamma & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \lambda \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} \,|\, \vec{t} \end{bmatrix} \begin{pmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{pmatrix} \tag{2.4}$$

where $(x, y, 1)^T$ is the projection point on the image, $(\hat{X}, \hat{Y}, \hat{Z}, 1)$ is the object point expressed in the world coordinate system, λ is an arbitrary scale factor, (f_x, f_y) are focal lengths in x and y directions expressed in pixels, γ is a skew factor that accounts for pixel aspect ratio, (u_0, v_0) are coordinates of the principal point in the image coordinate system, $\mathbf{R}$ is a rotation matrix between the world coordinate system and the camera coordinate system and $\vec{t}$ is a translation vector between those two coordinate systems.

2.4 Homography

Homography is a very important concept in computer vision especially when considering multiple cameras. Projections of a scene onto an image plane are related by homography in two situations. In one situation, several cameras are viewing a planar object; the projections of that planar object onto those cameras are related by homographies. In the second situation, a camera viewing a scene undergoes a rotation around its optical center; the different projections of the scene are again related by homographies. Below, we define homography and provide some examples.

2.4.1 Definition

Consider a point $\vec{x} = (x, y, w)^T$ in one image and its corresponding point $\vec{x}' = (x', y', w')^T$ in another image, where w and w' are non-zero scale factors. Corresponding points are projections of the same 3D feature observed in world coordinates, e.g., tip of a person's nose. These points are related by a 3×3 matrix called the *homography matrix* **H** (Malis and Vargas, 2007):

$$\vec{x}' = \mathbf{H}\vec{x}, \qquad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{2.5}$$

if and only if, the scene can be approximated by a planar surface, or when the scene is static and the camera motion is a pure rotation around its optical center. By applying a homography matrix to every pixel in one image, one can find the corresponding points in the other image.

Figure 2·6 shows five images taken from different orientations of a camera and Figure 2·7 shows the result of mosaicing those images together into a large image using homograpies. Since the camera undergoes rotation around its optical center, the homography is quite accurate in this case.

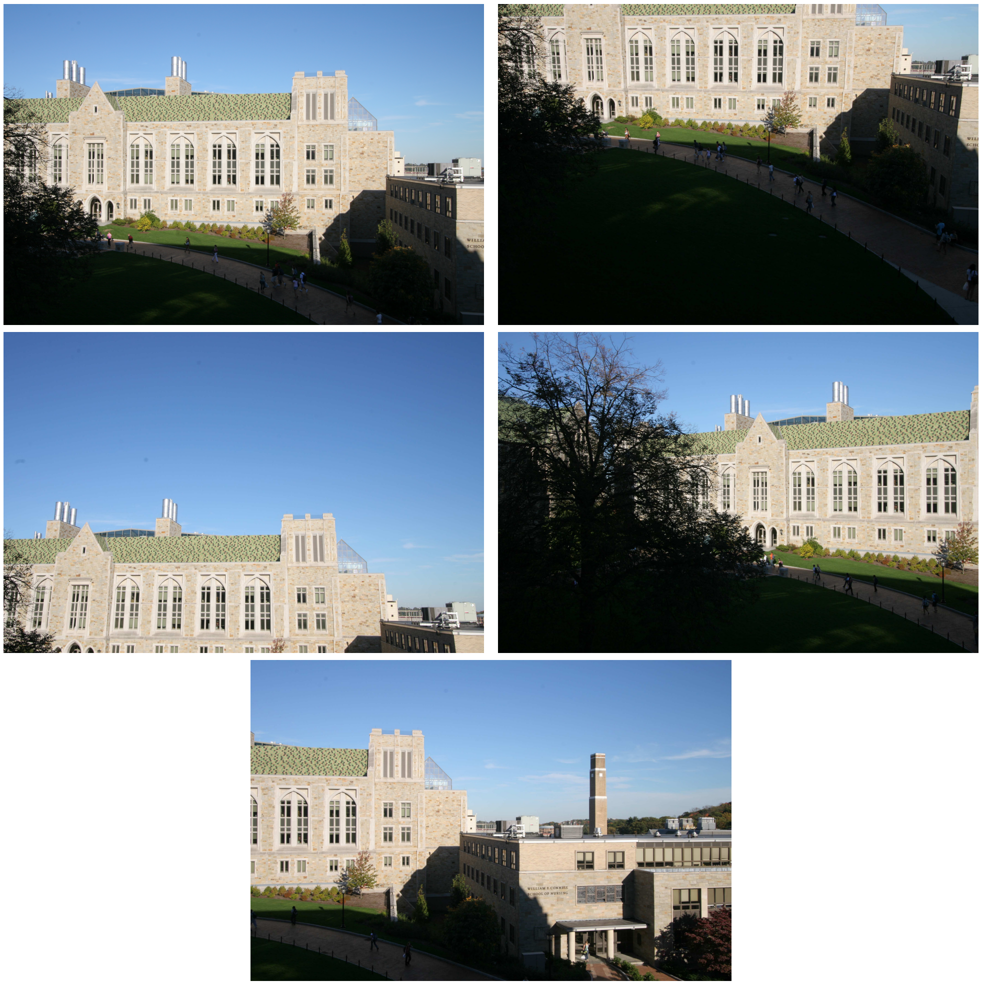


Figure 2·6: A scene captured from five different orientations of a camera.

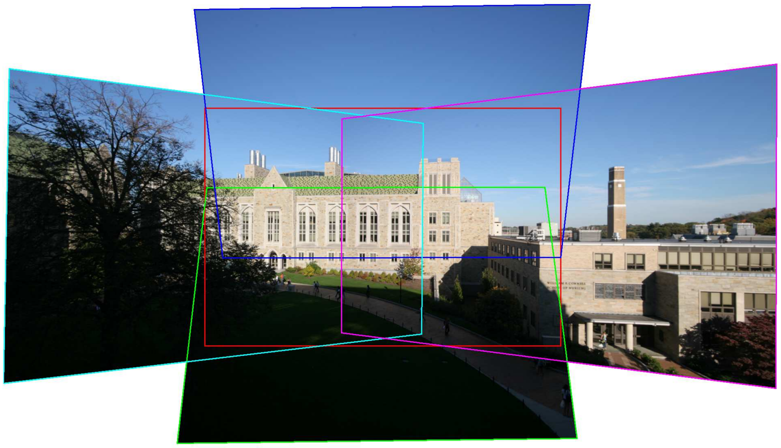


Figure 2·7: Alignment of five images from Figure 2·6 using homographies.

Assuming we already know the corresponding points in two images, how can we compute a homography given those corresponding points? We introduce a method called Direct Linear Transform (Hartley and Zisserman, 2004) next.

2.4.2 Estimation of homography using the Direct Linear Transform

The relationship (2.5) involves homogeneous vectors, so $\vec{x}'$ and $\mathbf{H}\vec{x}$ are not really equal to each other. They are equal to each other up to a non-zero scale factor s. Then, the correct relationship should be written as $\vec{x}' = s\mathbf{H}\vec{x}$, where s is a non-zero scale factor. If we express this equation in the notation of cross product, it becomes $\vec{x}' \times \mathbf{H}\vec{x} = \vec{0}$.

Let's denote the i^{th} pair of corresponding points in two images as $\vec{x_i}$ and $\vec{x_i}'$, and the j^{th} row of the matrix $\mathbf{H}$ by $\vec{h^j}^T$. Then, we have

$$\mathbf{H}\vec{x_i} = \begin{pmatrix} \vec{h^1}^T \vec{x_i} \\ \vec{h^2}^T \vec{x_i} \\ \vec{h^3}^T \vec{x_i} \end{pmatrix}$$

and, the cross product becomes

$$\begin{aligned}\vec{x_i}' \times \mathbf{H}\vec{x_i} &= \begin{pmatrix} y_i'\vec{h^3}^T\vec{x_i} - w_i'\vec{h^2}^T\vec{x_i} \\ w_i'\vec{h^1}^T\vec{x_i} - x_i'\vec{h^3}^T\vec{x_i} \\ x_i'\vec{h^2}^T\vec{x_i} - y_i'\vec{h^1}^T\vec{x_i} \end{pmatrix} \\ &= \begin{bmatrix} \vec{0}^T & -w_i'\vec{x_i}^T & y_i'\vec{x_i}^T \\ w_i'\vec{x_i}^T & \vec{0}^T & -x_i'\vec{x_i}^T \\ -y_i'\vec{x_i}^T & x_i'\vec{x_i}^T & \vec{0}^T \end{bmatrix} \begin{pmatrix} \vec{h^1} \\ \vec{h^2} \\ \vec{h^3} \end{pmatrix} \\ &= \vec{0}. \end{aligned} \tag{2.6}$$

The equation (2.6) has the form $\mathbf{A}_i\vec{h} = 0$, where $\mathbf{A}_i$ is a 3×9 matrix and $\vec{h}$ is a 9 dimensional vector made up of the entries of matrix $\mathbf{H}$ as follows:

$$\vec{h} = \begin{bmatrix} \vec{h^1} \\ \vec{h^2} \\ \vec{h^3} \end{bmatrix}.$$

There are three equations in (2.6), but only two of them are linearly independent. In consequence, each two points in correspondence give us two equations. If we only keep the first two equations, then (2.6) becomes

$$\mathbf{A}_i\vec{h} = \begin{bmatrix} \vec{0}^T & -w_i'\vec{x_i}^T & y_i'\vec{x_i}^T \\ w_i'\vec{x_i}^T & \vec{0}^T & -x_i'\vec{x_i}^T \end{bmatrix} \begin{pmatrix} \vec{h^1} \\ \vec{h^2} \\ \vec{h^3} \end{pmatrix} = 0 \tag{2.7}$$

where $\mathbf{A}_i$ is now a 2×9 matrix. Therefore, if we have n correspondences, we can stack up $\mathbf{A}_i$ matrices to form a $2n \times 9$ matrix $\mathbf{A}$.

Now our task is to solve $\mathbf{A}\vec{h} = 0$. Due to uncertainty in feature detection (locations $\vec{x}$ and $\vec{x}'$ are noisy), equation (2.7) may not have an exact solution. Instead, we would like to find an approximate solution $\vec{h}$ such that it minimizes the norm $\|\mathbf{A}\vec{h}\|$, for example under the condition that $\|\vec{h}\| = 1$.

The solution to this problem is known as the unit eigenvector corresponding to the smallest singular value of $\mathbf{A}$. We first apply SVD to $\mathbf{A}$ so that $\mathbf{A} = \mathbf{UDV^T}$. Then, the task is to minimize $\|\mathbf{UDV^T}\vec{h}\|$. Due to the fact that $\|\mathbf{U}\| = \|\mathbf{V}\| = \|\mathbf{V^T}\| = 1$, we have $\|\mathbf{UDV^T}\vec{h}\| = \|\mathbf{DV^T}\vec{h}\|$. Let's denote $\vec{y} = \mathbf{V^T}\vec{h}$. Finally, we need to minimize $\|\mathbf{D}\vec{y}\|$ subject to $\|\vec{y}\| = 1$. Since $\mathbf{D}$ is a matrix with entries on the diagonal in decreasing order, we want to make $\vec{y} = (0, 0, ..., 0, 1)^T$, and $\vec{h} = \mathbf{V}\vec{y}$ is then simply the last column of $\mathbf{V}$. Thus, we have solved the minimization problem.

In order to make geometric error of DLT algorithm invariant to similarity transforms of the images, we usually normalize coordinates of correspondences before applying the DLT algorithm. Here is the description of the normalized DLT algorithm:

(1) **Normalization of $\vec{x}$**: Compute a similarity transformation $\mathbf{T}$, consisting of a translation and scaling, that takes points $\vec{x}_i$ to a new set of points $\vec{\check{x}}_i$ such that the centroid of the points $\vec{\check{x}}_i$ is the coordinate origin $(0, 0)^T$, and their average distance from the origin is $\sqrt{2}$.

(2) **Normalization of $\vec{x}'$**: Compute a similar transformation $\mathbf{T}'$ for the points in the second image, transforming points $\vec{x}'_i$ to $\vec{\check{x}}'_i$.

(3) **DLT**: Apply the DLT algorithm to the correspondences $\vec{\check{x}}_i \leftrightarrow \vec{\check{x}}'_i$ to obtain a homography $\check{\mathbf{H}}$.

(iv) **Denormalization**: Set $\mathbf{H} = \mathbf{T}'^{-1}\check{\mathbf{H}}\mathbf{T}$.

Chapter 3

Phase correlation

An image plane is a 2D plane, so motion observed in the image plane cannot fully describe the 3D motion in the world coordinate system. For example, one will see no change in an image of a ball rotating around its center (ignoring the pattern on the ball, light conditions and other factors). However, 3D motion is somehow related to 2D motion and, in some circumstances, 2D motion is a good approximation of 3D motion. For example, when viewing a very far away object, both small horizontal camera translation and rotation around y axis (pam) will cause almost the same effect, namely a displacement in the image. Thus, under certain assumptions, such as large distance from the camera, we expect 2D motion estimated using phase correlation to be a good approximation of 3D motion we are interested in. We first show an example of phase correlation and then we discuss its mathematical details.

Phase correlation is an important method to estimate 2D displacements between two similar images. Phase correlation is computationally efficient, as it relies on FFT, and resilient to noise, occlusions, and other defects. For example, consider two images corrupted by iid Gaussian noise (Figure 3·1), of which one is translated by (30, 33) pixels. The white spot at (30, 33) in Figure 3·2 is the result of phase correlation applied to these images.

Let's now consider mathematical details of phase correlation. Let $I_a(x, y)$ be an $M \times N$ image and let $I_b(x, y) = I_a((x - \triangle x) \mathrm{mod} M, (y - \triangle y) \mathrm{mod} N)$ be its circularly-shifted version, by $\triangle x$ along x direction and by $\triangle y$ along y direction. Let $F_a(k, l) =$

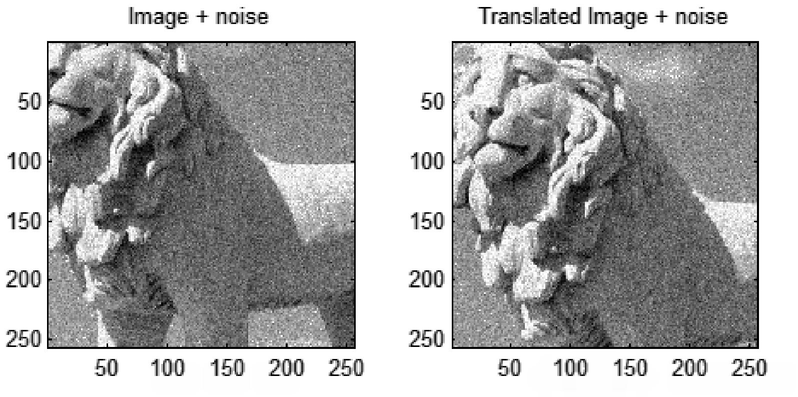


Figure 3·1: Two images corrupted by iid Gaussian noise. The right image is translated by (30, 33) pixels relative to the left image.

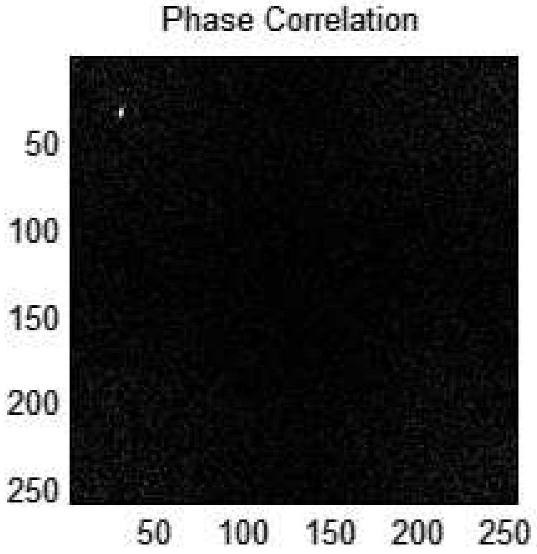


Figure 3·2: The result of phase correlation applied to two images from Figure 3·1

$\mathcal{F}\{I_a(x,y)\}$ and $F_b(k,l) = \mathcal{F}\{I_b(x,y)\}$, where $\mathcal{F}$ denotes the discrete-space Fourier transform. Then:

$$F_b(k,l) = F_a(k,l)e^{-2\pi i(\frac{k\triangle x}{M}+\frac{l\triangle y}{N})}.$$

The normalized cross-power spectrum of those two images is:

$$\begin{aligned} R(k,l) &= \frac{F_a(k,l)F_b^*(k,l)}{|F_a(k,l)F_b^*(k,l)|} \\ &= \frac{F_a(k,l)F_a^*(k,l)e^{2\pi i(\frac{k\triangle x}{M}+\frac{l\triangle y}{N})}}{|F_a(k,l)F_a^*(k,l)e^{2\pi i(\frac{k\triangle x}{M}+\frac{l\triangle y}{N})}|} \\ &= \frac{F_a(k,l)F_a^*(k,l)e^{2\pi i(\frac{k\triangle x}{M}+\frac{l\triangle y}{N})}}{|F_a(k,l)F_a^*(k,l)|} \\ &= e^{2\pi i(\frac{k\triangle x}{M}+\frac{l\triangle y}{N})} \end{aligned} \tag{3.1}$$

since $F_a(k,l)F_a^*(k,l) = |F_a(k,l)|^2 = |F_a(k,l)F_a^*(k,l)|$. Taking the inverse Fourier transform of $R(k,l)$, we obtain:

$$r(x,y) = \delta(x+\triangle x, y+\triangle y). \tag{3.2}$$

Thus, by finding the location of an impulse on surface $r(x,y)$, one can identify the displacement, or shift $(\triangle x, \triangle y)$ between images. In practice, the motion is often a combination of translation and rotation, rather than a pure translation. In that case, the correlation surface $r(x,y)$ will not be a Kronecker delta, but will be somewhat spread out or multiple peaks may be present. A suitable search for the dominant peak must be performed. However, the performance of phase correlation is not satisfactory when dealing with a combination of motions, especially if camera rotation or zoom are present.

First, we tested this algorithm on synthetic data: an original image and its shifted version (Figure 3·3). As we can see from Figure 3·4, there is a peak at (6, 15) which indicates the exact amount of shift in the synthetic data.

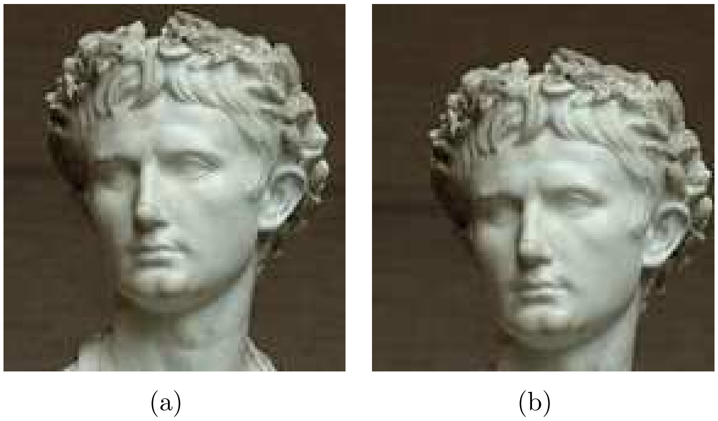

(a) (b)

Figure 3·3: (a) A synthetic image and (b) its shifted version (by (6,15)).

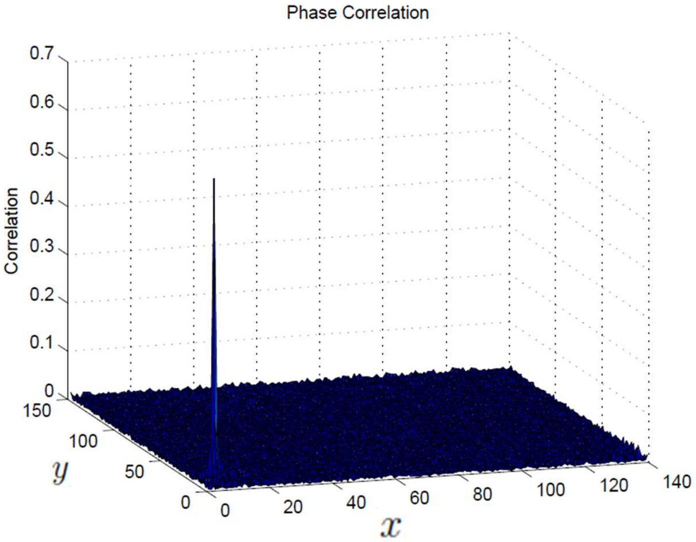


Figure 3·4: Correlation surface $r(x, y)$ computed using equations (3.1) and (3.2) between images from Figure 3·3.

We also applied phase correlation to real data. We captured video from a camera undergoing vibrations. Figure 3·5 shows two frames from that video. We estimated displacements in x and y directions over time by comparing the k^{th} frame denoted as $I(x, y, t = k)$ with the initial frame denoted as $I(x, y, t = 0)$. Since we do not have ground truth to compare this result with, we performed a different test. We motion compensated the video using vectors estimated by phase correlation as follows:

$$\hat{I}(x, y, t = k) = I(x + \triangle x^{(k)}, y + \triangle y^{(k)}, t = k)$$

where $\hat{I}(x, y, t = k)$ is the motion compensated k^{th} frame and $(\triangle x^{(k)}, \triangle y^{(k)})$ is displacement vector between the k^{th} frame and the initial frame. In order to show phase correlation is effective, we first calculated an error image $I_{error}(x, y, t = k)$ with no motion compensation

$$I_{error}(x, y, t = k) = I(x, y, t = k) - I(x, y, t = 0)$$

and then an error image $\hat{I}_{error}(x, y, t = k)$ after motion compensation

$$\hat{I}_{error}(x, y, t = k) = \hat{I}(x, y, t = k) - I(x, y, t = 0).$$

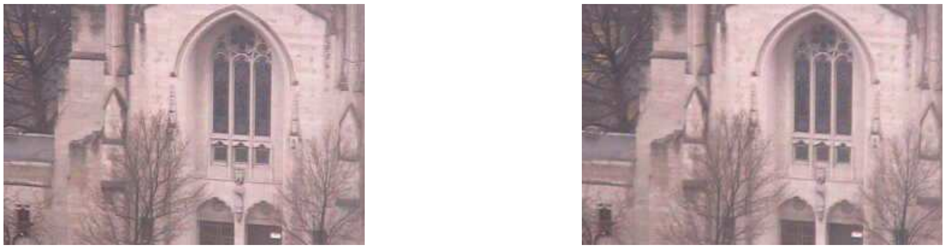
In addition, we calculated the mean-squared error before (ε) and after $(\hat{\varepsilon})$ motion compensation as follows:

$$\begin{cases} \varepsilon = \frac{1}{MNT} \sum\limits_{x=1}^{M} \sum\limits_{y=1}^{N} \sum\limits_{t=1}^{300} (I_{error}(x, y, t))^2 \\ \hat{\varepsilon} = \frac{1}{MNT} \sum\limits_{x=1}^{M} \sum\limits_{y=1}^{N} \sum\limits_{t=1}^{300} (\hat{I}_{error}(x, y, t))^2 \end{cases}$$

where T is the number of video frames used in the test.

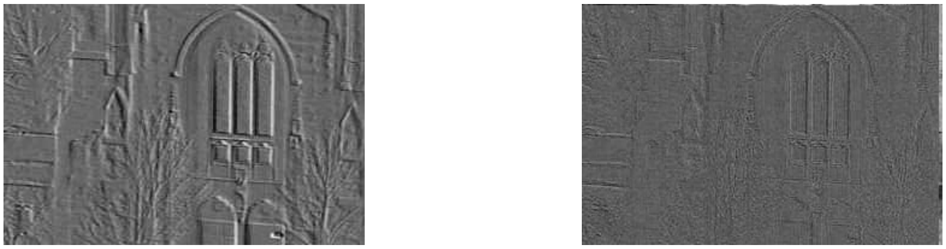
For video from Figure 3·5 we obtained $\hat{\varepsilon} = 13.46$ after compensation and $\varepsilon = 147.13$ before compensation, indicating that phase correlation estimates the displacements effectively. In Figure 3·6 we show the error images before ($I_{error}(x, y, t = k)$) and after ($\hat{I}_{error}(x, y, t = k)$) compensation for $k = 14$. We mapped the dynamic range of the error images to the range [0,255] for display. As we can see from Figure 3·6, the error is smaller after compensation. Figure 3·7 shows the estimated displacements in x and y directions over time. Clearly, the estimated displacements have integer values and many zeros. This is due to the fact that we applied phase correlation with full pixel precision and some displacements may be too small to be accurately captured by phase correlation. Also local motion of trees may have biased the estimation results.

Although the phase correlation algorithm is computationally efficient and works relatively well, it has severe limitations. It can only estimate spatially-invariant 2D displacements along x and y directions. It cannot compensate 2D rotations, 3D rotations and zooming effects. In consequence, it may not be accurate enough for our application and we have to consider alternatives.



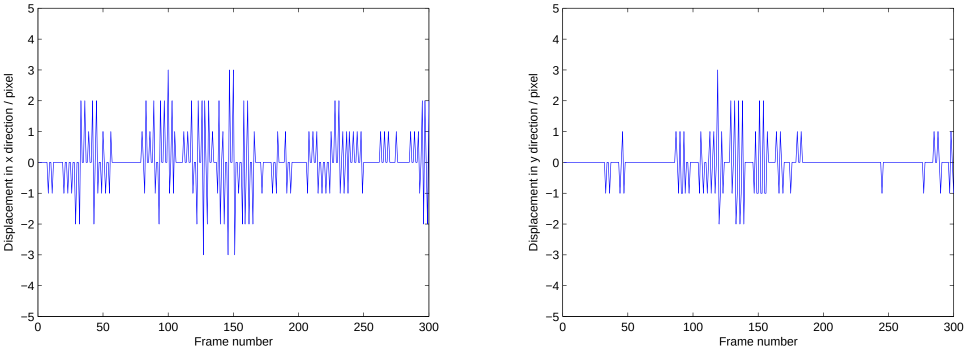
(a) $I(x, y, t = 0)$ (b) $I(x, y, t = 14)$

Figure 3·5: Two video frames captured by a vibrating camera.



(a) $I_{error}(x, y, t = 14)$ (b) $\hat{I}_{error}(x, y, t = 14)$

Figure 3·6: Error between images in Figure 3·5: (a) without and (b) with motion compensation.



(a) (b)

Figure 3·7: (a) Horizontal (x direction) and (b) vertical (y direction) displacement estimated from video shown in Figure 3·5 using phase correlation.

Chapter 4

Camera calibration method of Zhang

In order to control a robot by cameras or reconstruct a face from images taken by multiple cameras, it is important to know the relationship between coordinate systems of any two cameras or between a camera coordinate system and the world coordinate system. This procedure is called calibration. By calibrating a camera with the world, we can recover the position and orientation of the object and of the camera in the world coordinate system. After calibration, we can use 2D images to recover 3D motion either caused by camera motion or object motion. Thus, it is of interest to us to investigate calibration methods in order to recover 3D camera motion from 2D images.

There exits a popular camera calibration method that is effective not only in a laboratory environment but also in real world use (Zhang, 98) (Hartley and Zisserman, 2004). It only requires the camera to observe a planar pattern shown at a few different orientations. Without knowing the motion, it can calibrate the camera coordinate system with the world coordinate system. We discuss the details of this method below.

Let a 2D point be denoted by $\vec{x} = (x, y, 1)^T$, and let a 3D point be denoted by $\vec{X} = [X, Y, Z, 1]^T$. We can express a 3D point and its projection using the a pinhole camera model (equation (2.4)) as follows:

$$\vec{x} = \lambda \mathbf{A} [\, \mathbf{R} \quad \vec{t}\,] \vec{X} = \lambda \mathbf{A} [\, \vec{r_1} \quad \vec{r_2} \quad \vec{r_3} \quad \vec{t}\,] \vec{X}' \tag{4.1}$$

where λ is an arbitrary scale factor, $\mathbf{R}$ is a rotation matrix and $\vec{t}$ is translation vector which relates the world coordinate system to the camera coordinate system. $\vec{r_i}$ is i^{th} column of the rotation matrix $\mathbf{R}$. $\mathbf{A}$, called the intrinsic matrix, is given by equation (2.4)

$$\mathbf{A} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

If we assume $Z = 0$, which implies that target is a planar object at $Z = 0$ of the world coordinate system, then (4.1) becomes

$$\begin{aligned} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} &= \lambda \mathbf{A} [\, \vec{r_1} \quad \vec{r_2} \quad \vec{r_3} \quad \vec{t}\,] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \\ &= \lambda \mathbf{A} [\, \vec{r_1} \quad \vec{r_2} \quad \vec{t}\,] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}. \end{aligned} \tag{4.2}$$

We denote our new $\vec{X}$ as $(X, Y, 1)^T$, so that

$$\vec{x} = \mathbf{H}\vec{X} \qquad \text{with} \qquad \mathbf{H} = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{bmatrix} = [\, \vec{h}_1 \quad \vec{h}_2 \quad \vec{h}_3 \,] = \lambda \mathbf{A} [\, \vec{r_1} \quad \vec{r_2} \quad \vec{t}\,] \tag{4.3}$$

where $\vec{h_i} = (h_{i1}, h_{i2}, h_{i3})^T$ is i^{th} column of $\mathbf{H}$. Using the constraint that $\vec{r}_1$ and $\vec{r}_2$ are orthonormal, we get

$$\left\{ \begin{array}{l} \vec{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \vec{h}_2 = 0 \\ \vec{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \vec{h}_1 = \vec{h}_2^T \mathbf{A}^{-T} \mathbf{A}^{-1} \vec{h}_2 \end{array} \right. \tag{4.4}$$

We introduce the following notations in order to solve equation (4.4) for $\mathbf{A}$. Let

$$\begin{aligned}\mathbf{B} &= \mathbf{A}^{-T}\mathbf{A}^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{f_x^2} & -\frac{\gamma}{f_x^2 f_y} & \frac{v_0\gamma - u_0 f_y}{f_x^2 f_y} \\ -\frac{\gamma}{f_x^2 f_y} & \frac{\gamma^2}{f_x^2 f_y^2} + \frac{1}{f_y^2} & -\frac{\gamma(v_0\gamma - u_0 f_y)}{f_x^2 f_y^2} - \frac{v_0}{f_y^2} \\ \frac{v_0\gamma - u_0 f_y}{f_x^2 f_y} & -\frac{\gamma(v_0\gamma - u_0 f_y)}{f_x^2 f_y^2} - \frac{v_0}{f_y^2} & \frac{(v_0\gamma - u_0 f_y)^2}{f_x^2 f_y^2} + \frac{v_0^2}{f_y^2} + 1 \end{bmatrix}. \qquad (4.5)\end{aligned}$$

Clearly, $\mathbf{B}$ is a symmetric matrix. By using matrix $\mathbf{B}$, equation (4.4) becomes:

$$\begin{cases} \vec{h}_1^T \mathbf{B} \vec{h}_2 = 0 \\ \vec{h}_1^T \mathbf{B} \vec{h}_1 = \vec{h}_2^T \mathbf{B} \vec{h}_2 \end{cases} . \qquad (4.6)$$

Let's define $\vec{b}$ as

$$\vec{b} \equiv (\ B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}\ )^T$$

and $\vec{v}_{ij}$ as

$$\vec{v}_{ij} \equiv (\ h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}\ )^T.$$

Because $\vec{h}_i^T \mathbf{B} \vec{h}_j$ and $\vec{v}_{ij}^T \vec{b}$ are equal to each other, we can rewrite equation (4.6) using $\vec{b}$ and $\vec{v}_{ij}$ as follows:

$$\begin{cases} \vec{v}_{12}^T \vec{b} = 0 \\ \vec{v}_{11}^T \vec{b} = \vec{v}_{22}^T \vec{b} \end{cases} . \qquad (4.7)$$

By rewriting equation (4.7) into matrix representation, we obtain:

$$\begin{bmatrix} \vec{v}_{12}^T \\ (\vec{v}_{11} - \vec{v}_{22})^T \end{bmatrix} \vec{b} = 0. \qquad (4.8)$$

If we have many images, we stack up those equations similarly to (4.8) and obtain:

$$\mathbf{V}\vec{b} = 0 \qquad (4.9)$$

The solution to (4.9) is the eigenvector of $\mathbf{V}^T\mathbf{V}$ associated with the smallest eigenvalue mentioned in Section 2.4.2. After $\vec{b}$ is obtained, we can uniquely extract the intrinsic parameters from matrix $\mathbf{B}$ using equation (4.5) as follows:

$$\begin{cases} f_x = \sqrt{\frac{1}{B_{11}}} \\ \\ f_y = \sqrt{\frac{B_{11}}{B_{11}B_{22}-B_{12}^2}} \\ \\ \gamma = -\sqrt{\frac{B_{12}}{B_{11}^2-B_{11}B_{12}}} \\ \\ u_0 = \frac{B_{12}B_{23}-B_{22}B_{13}}{B_{11}B_{12}-B_{12}^2} \\ \\ v_0 = \frac{B_{12}B_{13}-B_{11}B_{23}}{B_{11}B_{22}-B_{12}^2} \end{cases} .$$

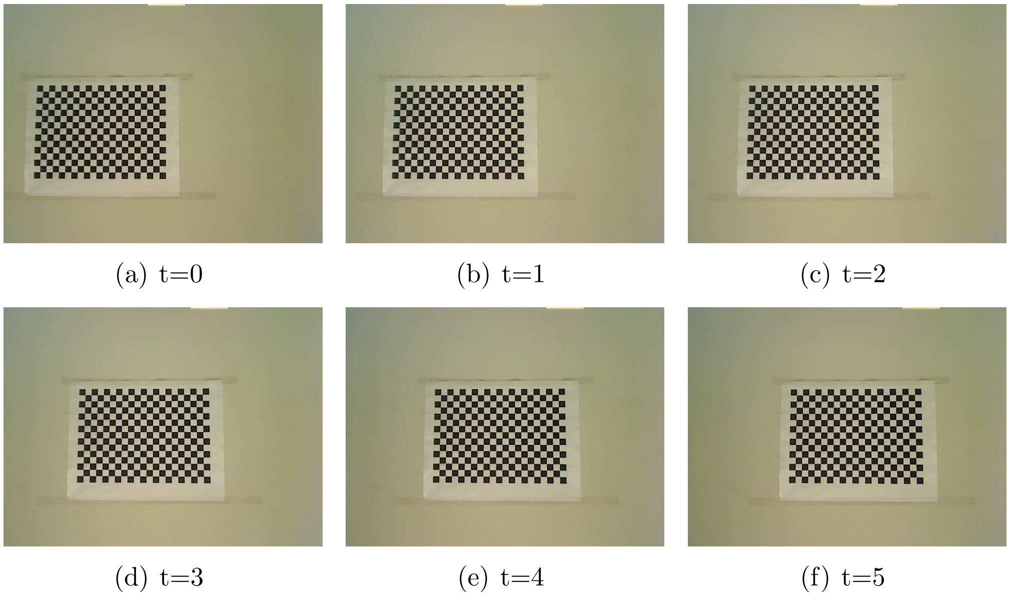
After $\mathbf{A}$ is calculated, we finally obtain:

$$\begin{cases} \vec{r_1} = \lambda \mathbf{A}^{-1}\vec{h}_1 \\ \vec{r_2} = \lambda \mathbf{A}^{-1}\vec{h}_2 \\ \vec{r_3} = \vec{r_1} \times \vec{r_2} \\ \vec{t} = \lambda \mathbf{A}^{-1}\vec{h}_3 \end{cases} \tag{4.10}$$

with $\lambda = 1/\|\mathbf{A}^{-1}\vec{h}_1\| = 1/\|\mathbf{A}^{-1}\vec{h}_2\|$ to assure unit length of $\vec{r_1}$, $\vec{r_2}$ and $\vec{r_3}$.

We have tested this algorithm on real data using a chessboard pattern as target. Figures 4·1 and 4·2 show two different kinds of motion: in one case the camera is moving to the left parallel to the target and in the other case it is moving away from the target perpendicularly. In both cases, we move the camera inch by inch. The images in Figure 4·2 may seem a bit odd since the table is at the top of the images. It is so because we vertically flipped the camera in order to avoid the table occupying too much space in the images (without flipping camera's center was too close to the table).

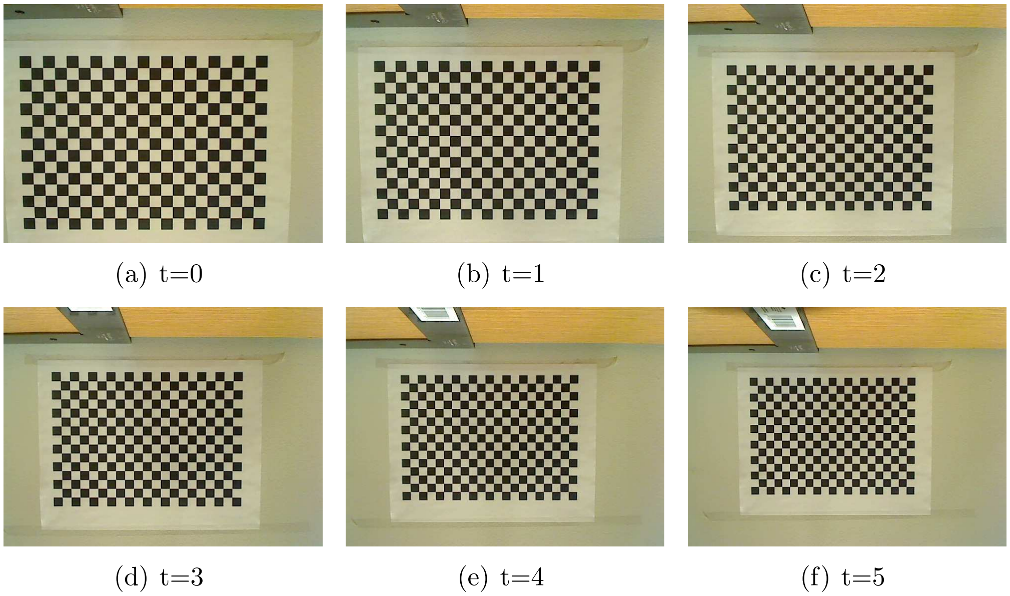
We can show the experimental results geometrically in two different ways. In one setting, that we call camera-centered view, we fix the position of the camera and show different positions of the pattern. In a world-centered view, we fix the position



(a) t=0 (b) t=1 (c) t=2

(d) t=3 (e) t=4 (f) t=5

Figure 4·1: Consecutive frames of a video captured by camera moving parallel to the target (chessboard pattern).



(a) t=0 (b) t=1 (c) t=2

(d) t=3 (e) t=4 (f) t=5

Figure 4·2: Consecutive frames of a video captured by camera (vertically flipped) moving perpendicular to the target (chessboard pattern).

of the pattern and show different positions of the camera. The second setting is more intuitive as it corresponds to the way we set up the problem.

Figures 4·3 and 4·4 show the estimation results for the first case, i.e., camera motion parallel to the target, in both camera-centered view and world-centered view. Figures 4·5 and 4·6 show the estimation results for the second case. As shown in Figure 4·3, the chessboard pattern is moving with constant velocity to the right, i.e., dispalcement from frame to frame is fixed, and as shown in Figure 4·4, the camera is moving with constant velocity to the left, both consistent with the fact that we are moving the camera to the left parallel to the chessboard pattern. Figures 4·5 and 4·6 also support the fact that we are moving the camera away from the chessboard pattern perpendicularly.

Tables 4.1 and 4.2 show the above experimental results numerically. We denote the estimated translation and rotation of the k^{th} frame relative to the first frame ($k = 0$) as $\vec{t_k} = (t_x^{(k)}, t_y^{(k)}, t_y^{(k)})^T$ (the unit is 1 millimeter) and $\vec{\theta_k} = (\theta_x^{(k)}, \theta_y^{(k)}, \theta_y^{(k)})^T$ (the unit is 1 degree), and denote the increments of translation and rotation of the k^{th} frame relative to the $(k-1)^{th}$ frame as $\triangle\vec{t_k} = (\triangle t_x^{(k)}, \triangle t_y^{(k)}, \triangle t_y^{(k)})^T$ and $\triangle\vec{\theta_k} = (\triangle\theta_x^{(k)}, \triangle\theta_y^{(k)}, \triangle\theta_y^{(k)})^T$. As we can see from Tables 4.1 and 4.2 the increments $\triangle t_x^{(k)}$ in the first case and $\triangle t_z^{(k)}$ in the second case are much larger numbers compared to other increments, because we are moving along the x axis in the first case and along the z axis in the second case. One can observe that $\triangle t_y^{(k)}$ and $\triangle t_z^{(k)}$ in the first case and $\triangle t_x^{(k)}$ and $\triangle t_y^{(k)}$ in the second case are not close to zero. It is because the table used in testing was not flat and level. In addition, since we did not provide the size of the chessboard pattern and the pixel size, the estimation is only proportional to the ground truth. We calculated the mean and standard deviation of translation increments along x-axis in the first case and along z-axis in the second case. Respectively, they were -513 and 4.7969 in the first case and -465.8 and 13.4663 in the second case.

Since the standard deviations are small in both cases compared to the means, we can conclude that this method estimates the motion of the camera accurately.

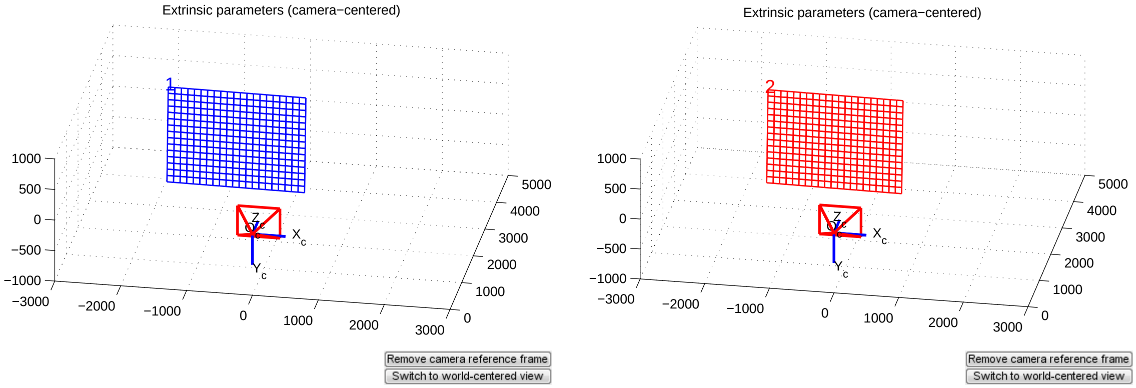
Table 4.1: Translation and rotation increments for a camera moving parallel to the target.

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$\triangle t_x^{(k)}$	-514.1	-516.3	-509.7	-518.3	-506.6
$\triangle t_y^{(k)}$	-16.2	-10.3	-13.6	-11.5	-16.0
$\triangle t_z^{(k)}$	-2.6	-3.8	-6.4	-7.0	-8.6
$\triangle \theta_x^{(k)}$	-0.02	0.04	0.16	-0.03	-0.18
$\triangle \theta_y^{(k)}$	0.03	0.02	-0.13	-0.15	-0.08
$\triangle \theta_z^{(k)}$	-0.10	-0.09	0.03	0.05	-0.02

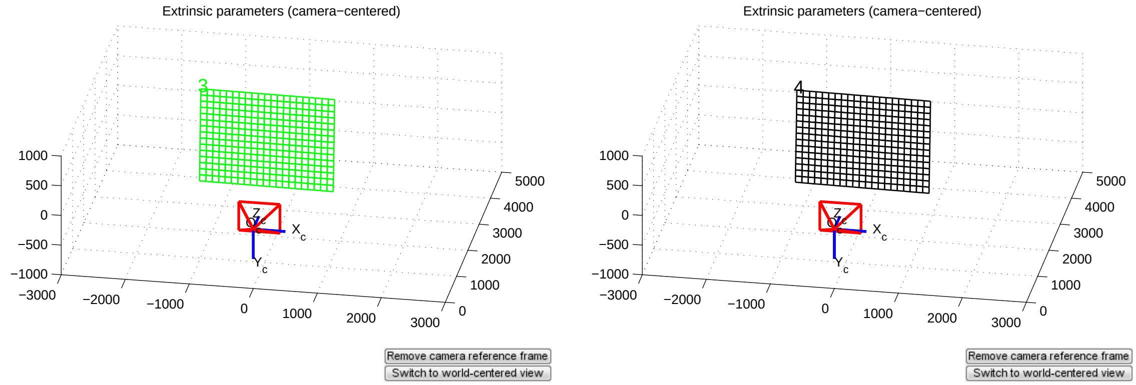
In conclusion, Zhang's method calibrates camera's coordinate system with the world coordinate system. It can even estimate camera motion in the units of pixel size. As long as we know the pixel size and provide coordinates of target points in absolute units, we will recover the camera motion in absolute units.

Thus, why don't we use this method to estimate camera motion? The critical fact is that it requires the frontal view of the pattern as implied by equation (4.2), in which it is assumed that the target pattern is on the plane $Z = 0$ of the world coordinate system. In other words, if we use a chessboard pattern as the target, we should provide the coordinates of feature points such as the left image in Figure 4·7 rather than the right one.

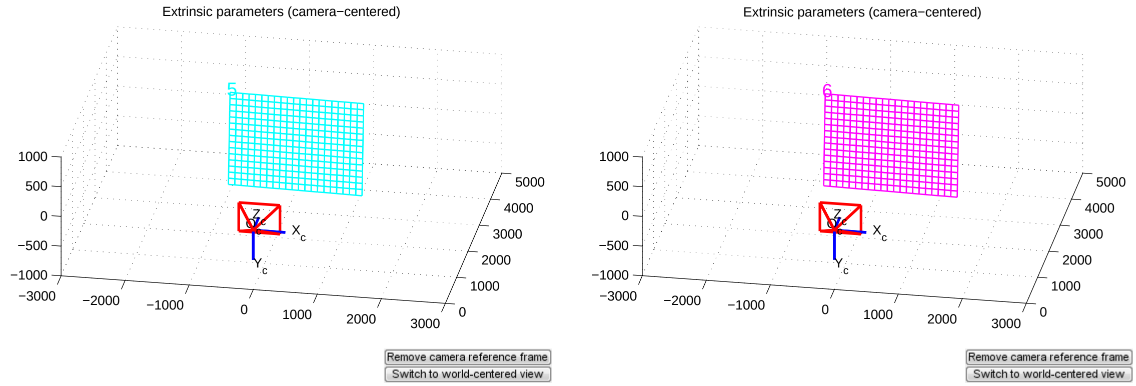
As we mentioned before, we expect to use thousands of surveillance cameras and thus it is impossible to provide the structure of targets except perhaps the fact that they are planar. Furthermore, we don't really care about the units of camera motion,



(a) t=0 (b) t=1

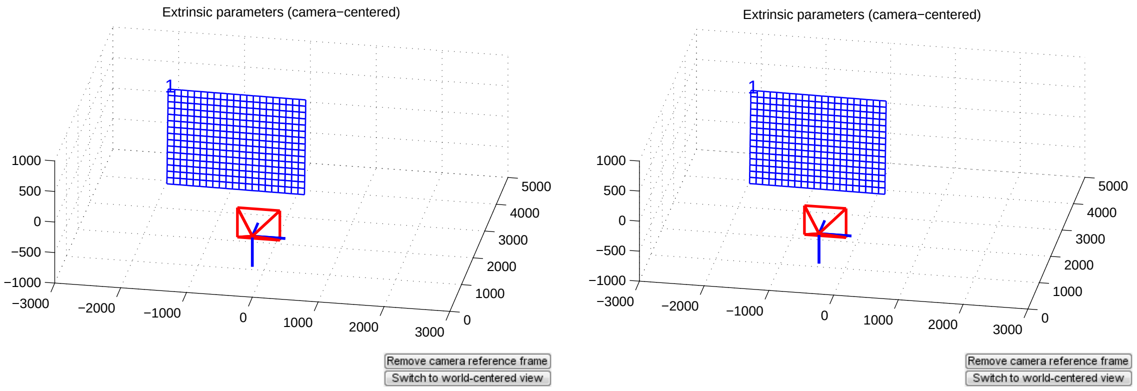


(c) t=2 (d) t=3

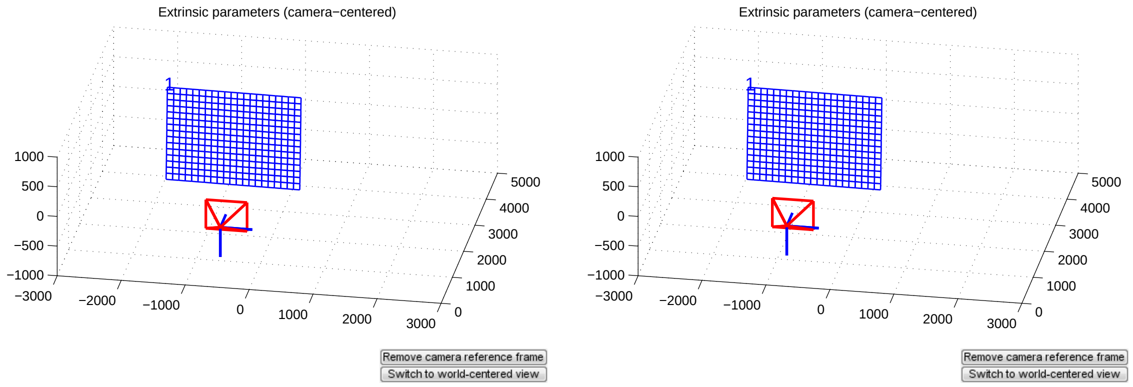


(e) t=4 (f) t=5

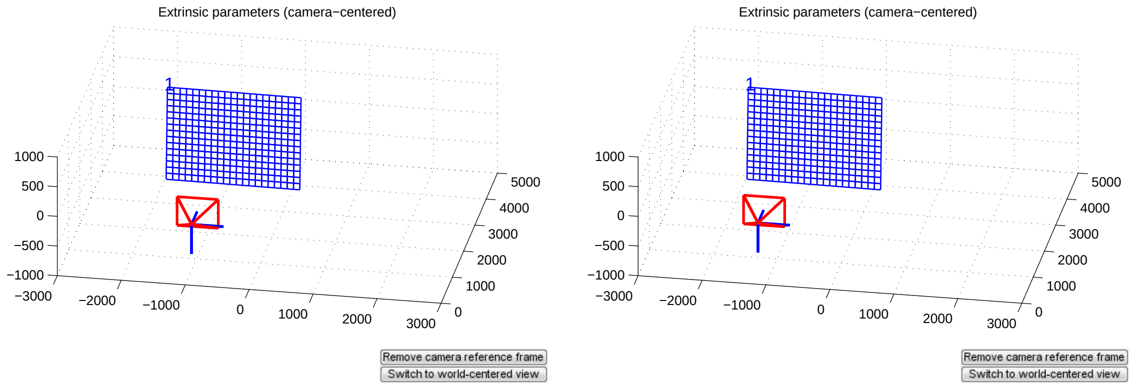
Figure 4·3: Geometric interpretation of experimental results for camera motion parallel to the target (camera-centered view).


Extrinsic parameters (camera−centered)
1
1000
500
0
−500
−1000
−3000
−2000
−1000
0
1000
2000
3000
0
1000
2000
3000
4000
5000
Remove camera reference frame
Switch to world-centered view
Extrinsic parameters (camera−centered)
1
Remove camera reference frame
Switch to world-centered view

(a) t=0 (b) t=1

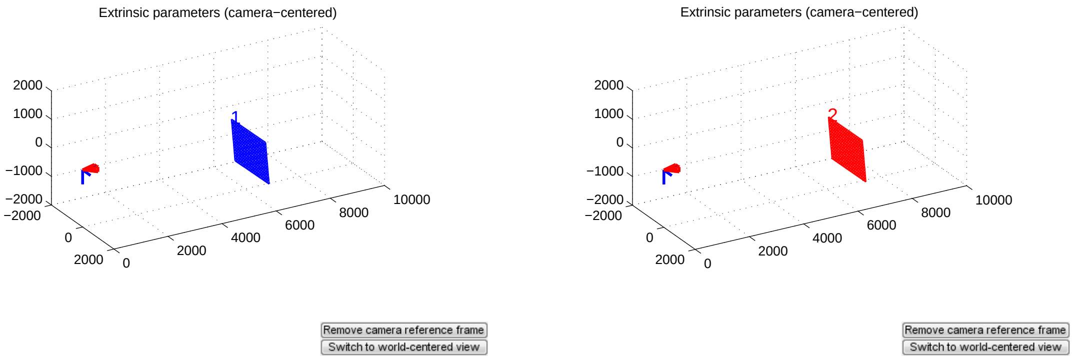

Extrinsic parameters (camera−centered)
1
Remove camera reference frame
Switch to world-centered view
Extrinsic parameters (camera−centered)
1
Remove camera reference frame
Switch to world-centered view

(c) t=2 (d) t=3

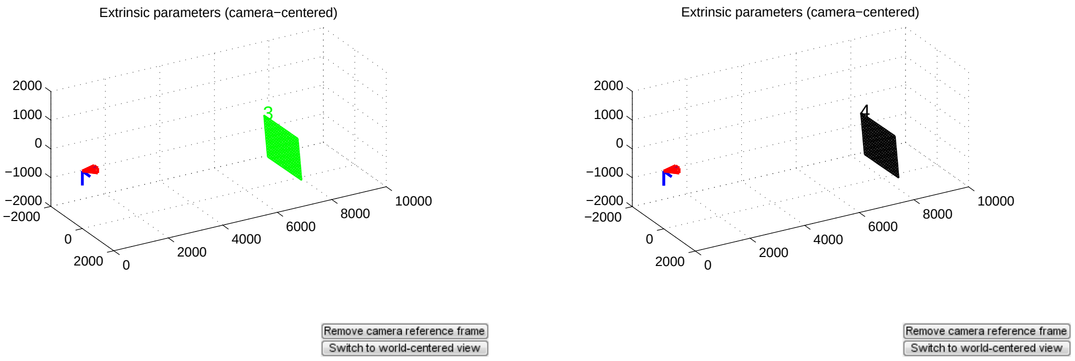

Extrinsic parameters (camera−centered)
1
Remove camera reference frame
Switch to world-centered view
Extrinsic parameters (camera−centered)
1
Remove camera reference frame
Switch to world-centered view

(e) t=4 (f) t=5

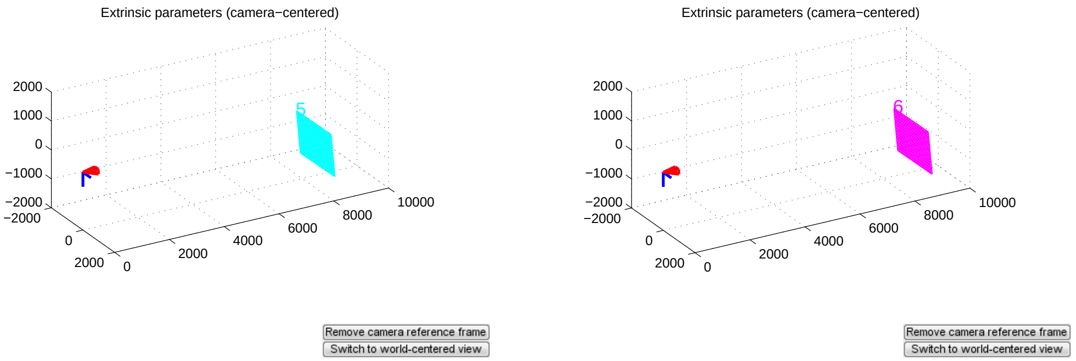
Figure 4·4: Geometric interpretation of experimental results for camera motion parallel to the target (world-centered view).



(a) t=0 (b) t=1

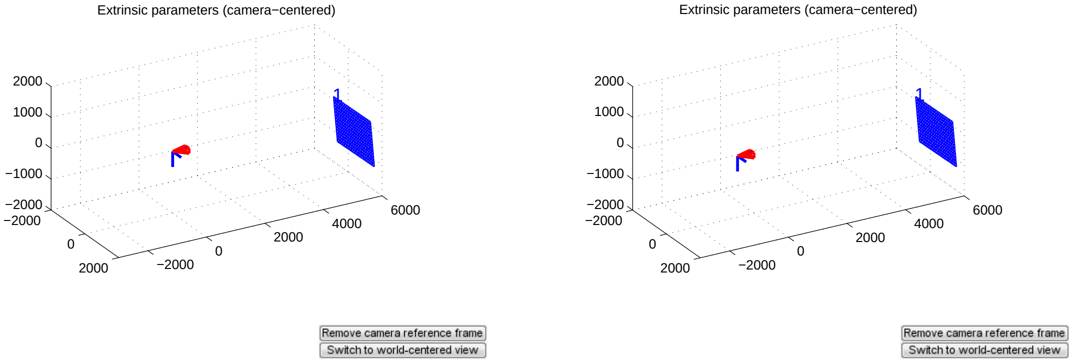


(c) t=2 (d) t=3

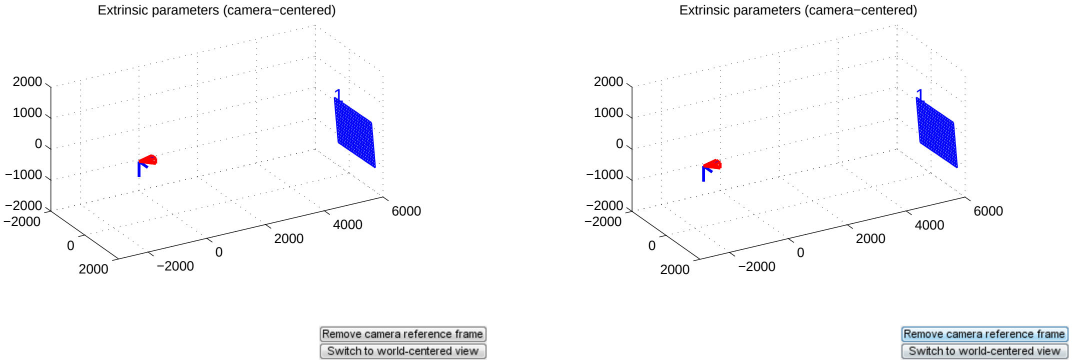


(e) t=4 (f) t=5

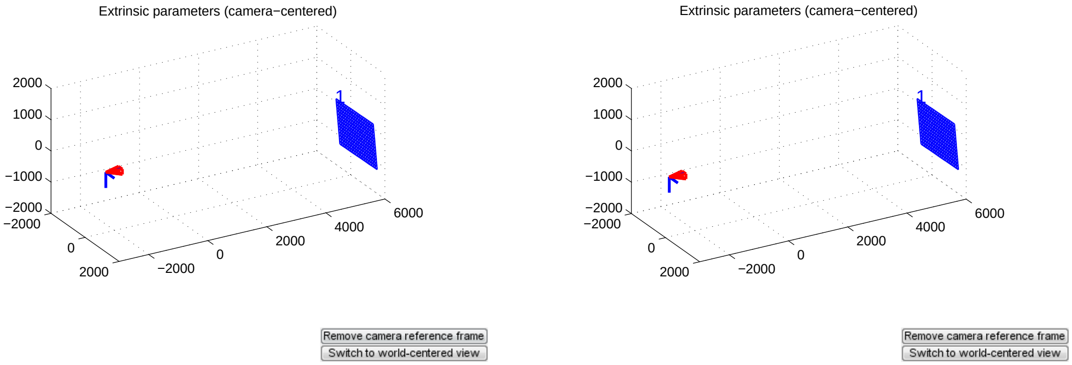
Figure 4·5: Geometric interpretation of experimental results for camera motion perpendicular to the target (camera-centered view).



(a) t=0 (b) t=1



(c) t=2 (d) t=3

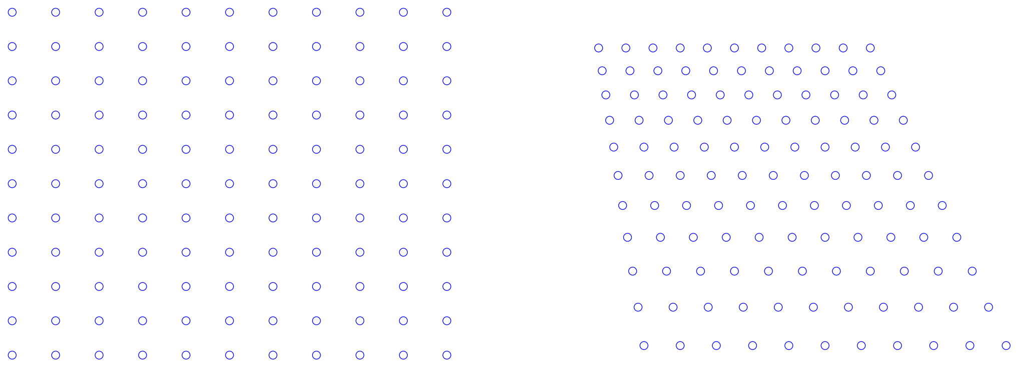


(e) t=4 (f) t=5

Figure 4·6: Geometric interpretation of experimental results for camera motion perpendicular to the target (world-centered view).

Table 4.2: Translation and rotation increments for a camera moving perpendicular to the target.

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
$\triangle t_x^{(k)}$	-19.4	-18.6	-28.2	-30.6	-9.2
$\triangle t_y^{(k)}$	-22.8	-27.6	-29.0	-33.8	-36.6
$\triangle t_z^{(k)}$	-451.4	-467.8	-459.6	-462.8	-487.4
$\triangle \theta_x^{(k)}$	0.03	0.08	-0.03	0.13	-0.04
$\triangle \theta_y^{(k)}$	0.01	-0.05	-0.08	-0.03	0.08
$\triangle \theta_z^{(k)}$	-0.07	0.10	0.03	-0.06	-0.08



(a) (b)

Figure 4·7: (a) Frontal view and (b) non-frontal view of feature points on the chessboard pattern so it is acceptable to estimate camera motion in relative units.

Chapter 5

Homography decomposition method of Faugeras and Lustman

As discussed in Chapter 2, a homography contains 3D translation and rotation between two camera positions. In addition, homography can be calculated directly from two images without using the 3D coordinate system, which is exactly what we want because we are not interested in the structure of the 3D scene. What we want to compute is the relative position of the camera, and a homography contains this information. However, 3D translation and rotation are not immediately appeared in a homography. We have to find a way to extract the information about 3D translation and rotation from a homography.

The method we are going to describe uses the fundamental property that projections of planar objects are related by homographies (Faugeras and Lustman, 1988). The authors show that given a homography, the position and orientation of the second camera and the target plane can be recovered with respect to the first camera. There exists an ambiguity when solving the problem, but it can be removed by including another plane or by using a third image of the same plane.

The position and orientation of the second camera with respect to the first one are defined by a translation vector $\vec{t}$ and a rotation matrix $\mathbf{R}$, respectively. The object plane is defined by its normal direction $\vec{n}$ and its distance d to the origin of the first camera coordinate system is shown in Figure 5·1. We will show it is possible to decompose a homography into physical parameters:

1. the normal direction $\vec{n}$ to the reference plane,
2. the rotation matrix $\mathbf{R}$,
3. the ratio $\vec{t}/d$ of the translation and the distance of the plane to the origin.

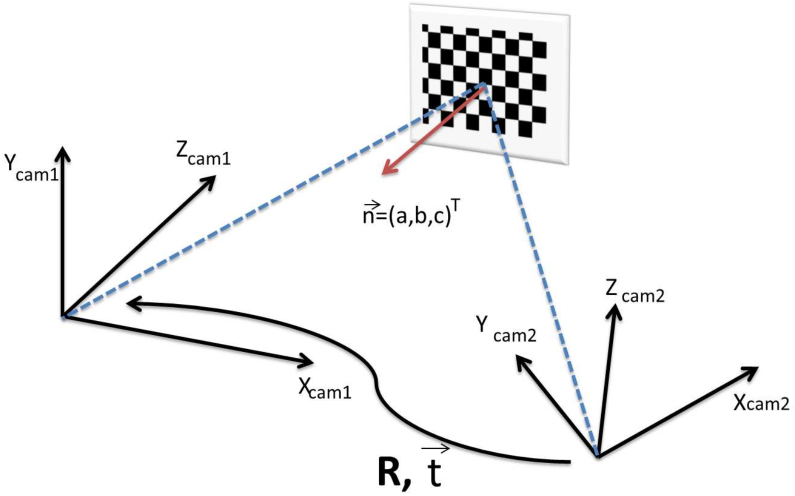


Figure 5·1: The relationship between object plane (chessboard pattern), the first camera coordinate system $(X_{cam1}, Y_{cam1}, Z_{cam1})$ and the second camera coordinate system $(X_{cam2}, Y_{cam2}, Z_{cam2})$.

Consider a planar object denoted by $\vec{\pi} = (\vec{n}^T, d)^T = (a, b, c, d)^T$ where $\vec{n}$ is the normal of this object in the first camera coordinate system. A 3D point on this planar object is denoted by $\vec{X} = (X, Y, Z, W)^T$. Recall the three results from Section 2.2 (page 9). We can generalize result 1 to 3D space: a point $\vec{X}$ lies on the plane $\vec{\pi}$ if and only if $\vec{\pi}^T\vec{X} = 0$. If we assume two camera coordinate sysmtes are related via $\mathbf{R}$ and $-\vec{t}$, we can express their 2D projections as follows

$$\begin{cases} \vec{x}_1 = \mathbf{A}[\,\mathbf{I}\,|\,0\,]\vec{X} \\ \vec{x}_2 = \mathbf{A}[\,\mathbf{R}\,|\,-\vec{t}\,]\vec{X} \end{cases} . \tag{5.1}$$

From the first equation in (5.1), we have $\vec{X} = (\mathbf{A}^{-1}\vec{x}_1, W)^T$. Since the 3D point $\vec{X}$ is on the planar object, expressed as $\vec{\pi}^T\vec{X} = 0$, solving for W we have $W = -\frac{\vec{n}^T\mathbf{A}^{-1}\vec{x}_1}{d}$.

Substituting $\vec{X}$ in the second equation in (5.1), we have

$$\begin{aligned}
\vec{x}_2 &= \mathbf{A}[\,\mathbf{R}\,|\,-\vec{t}\,]\vec{X} \\
&= \mathbf{A}[\,\mathbf{R}\,|\,-\vec{t}\,]\begin{pmatrix} \mathbf{A}^{-1}\vec{x}_1 \\ -\frac{\vec{n}^T\mathbf{A}^{-1}\vec{x}_1}{d} \end{pmatrix} \\
&= \mathbf{A}(\mathbf{R}\mathbf{A}^{-1}\vec{x}_1 + \vec{t}\frac{\vec{n}^T\mathbf{A}^{-1}\vec{x}_1}{d}) \\
&= \mathbf{A}(\mathbf{R} + \frac{\vec{t}\vec{n}^T}{d})\mathbf{A}^{-1}\vec{x}_1
\end{aligned} \tag{5.2}$$

Recall that, two projections of a planar object are related by a homography:

$$\vec{x}_2 = s\mathbf{H}\vec{x}_1.$$

Combined with equation (5.2), a homography matrix $\mathbf{H}$, up to a scale factor, can be described as follows:

$$\mathbf{H} = \mathbf{A}(d\mathbf{R} + \vec{t}\vec{n}^T)\mathbf{A}^{-1}. \tag{5.3}$$

We denote normalized homography as

$$\hat{\mathbf{H}} = \mathbf{A}^{-1}\mathbf{H}\mathbf{A} = d\mathbf{R} + \vec{t}\vec{n}^T \tag{5.4}$$

The goal is to solve equation (5.4), i.e., based on a normalized homography matrix $\hat{\mathbf{H}}$, find translation vector $\vec{t}$ and rotation matrix $\mathbf{R}$.

Using singular value decomposition (SVD), we can decompose $\hat{\mathbf{H}}$ into $\mathbf{U\Lambda V^T}$, where $\mathbf{\Lambda}$ is a diagonal matrix with square roots of eigenvalues λ_i of $\hat{\mathbf{H}}\hat{\mathbf{H}}^T$ sorted in decreasing order: $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$, and $\mathbf{U}$ and $\mathbf{V}$ being orthonormal matrices.

There is no restriction on homography which means that any 3×3 matrix can be a homography matrix. Thus, we can assume $\mathbf{\Lambda}$ is a homography matrix,

$$\mathbf{\Lambda} = d^{'}\mathbf{R}^{'} + \vec{t}^{'}\vec{n}^{'T} \tag{5.5}$$

where $\mathbf{R}$, d, $\vec{t}$ and $\vec{n}$ are related to $\mathbf{R}'$, d', $\vec{t'}$ and $\vec{n}'$ as follows:

$$\begin{cases} \mathbf{R} = \mathbf{U}\mathbf{R}^{'}\mathbf{V^T} \\ \vec{t} = \mathbf{U}\vec{t^{'}} \\ d = d^{'} \\ \vec{n} = \mathbf{V}\vec{n}' \end{cases} \tag{5.6}$$

with

$$\vec{t'} = (t_1', t_2', t_3')^T \qquad \vec{n}' = (a_1, a_2, a_3)^T.$$

We assume that $\vec{n}'$ is a unit-length vector. If we expand equation (5.5), we will get

$$\begin{aligned} \mathbf{\Lambda} &= \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \\ &= d^{'}\mathbf{R}^{'} + \begin{pmatrix} t_1^{'} \\ t_2^{'} \\ t_3^{'} \end{pmatrix} \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix} \\ &= d^{'}\mathbf{R}^{'} + \begin{bmatrix} t_1^{'}a_1 & t_1^{'}a_2 & t_1^{'}a_3 \\ t_2^{'}a_1 & t_2^{'}a_2 & t_2^{'}a_3 \\ t_3^{'}a_1 & t_3^{'}a_2 & t_3^{'}a_3 \end{bmatrix}. \end{aligned} \tag{5.7}$$

If we use an orthonormal basis: $\vec{e_1} = (1,0,0)^T, \vec{e_2} = (0,1,0)^T, \vec{e_3} = (0,0,1)^T$, we can express equation (5.7) as follows:

$$\mathbf{\Lambda}\vec{e_i} = \lambda_i\vec{e_i} = d^{'}\mathbf{R}^{'}\vec{e_i} + \vec{t}^{'}a_i \quad \text{for} \quad i = 1,2,3$$

which implies

$$\begin{cases} \lambda_1\vec{e_1} = d^{'}\mathbf{R}^{'}\vec{e_1} + \vec{t}^{'}a_1 \\ \lambda_2\vec{e_2} = d^{'}\mathbf{R}^{'}\vec{e_2} + \vec{t}^{'}a_2 \\ \lambda_3\vec{e_3} = d^{'}\mathbf{R}^{'}\vec{e_3} + \vec{t}^{'}a_3 \end{cases} \tag{5.8}$$

or

$$\lambda_i\vec{e_i} = d^{'}\mathbf{R}^{'}\vec{e_i} + \vec{t}^{'}a_i \quad \text{for} \quad i = 1,2,3. \tag{5.9}$$

If we eliminate $\vec{t'}$, we have

$$\lambda_i\vec{e_i}a_j - \lambda_j\vec{e_j}a_i = d^{'}\mathbf{R}^{'}(a_j\vec{e_i} - a_i\vec{e_j}) \quad \text{for} \quad i \neq j. \tag{5.10}$$

As $\mathbf{R}'$ is a rotation matrix, it preserves the vector norm :

$$\|\lambda_i \vec{e_i} a_j - \lambda_j \vec{e_j} a_i\| = \|d'(a_j \vec{e_i} - a_i \vec{e_j})\| \quad \text{for} \quad i \neq j$$

thus leading to:

$$\left\{ \begin{array}{l} (d'^2 - \lambda_1^2)a_3^2 + (d'^2 - \lambda_3^2)a_1^2 = 0 \\ (d'^2 - \lambda_2^2)a_1^2 + (d'^2 - \lambda_1^2)a_2^2 = 0 \\ (d'^2 - \lambda_3^2)a_2^2 + (d'^2 - \lambda_2^2)a_3^2 = 0 \end{array} \right. . \tag{5.11}$$

These are linear equations with unknowns a_1^2, a_2^2, a_3^2. Equations (5.11) have the form $\mathbf{C}\vec{x} = 0$, where

$$\mathbf{C} = \begin{bmatrix} (d'^2 - \lambda_3^2) & 0 & (d'^2 - \lambda_1^2) \\ (d'^2 - \lambda_2^2) & (d'^2 - \lambda_1^2) & 0 \\ 0 & (d'^2 - \lambda_3^2) & (d'^2 - \lambda_2^2) \end{bmatrix} \quad \text{and} \quad \vec{x} = \begin{pmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{pmatrix}.$$

In order to obtain a non-zero solution, $\mathbf{A}$ must be singular which means its determinant must be zero. Otherwise, we will get a solution all equal to zero which violates the assumption that $\vec{n}'$ has a unit norm. Thus, we have

$$(d'^2 - \lambda_1^2)(d'^2 - \lambda_2^2)(d'^2 - \lambda_3^2) = 0. \tag{5.12}$$

Equation (5.12) implies that $d' = \pm\lambda_1$, $\pm\lambda_2$ or $\pm\lambda_3$. Now we are facing three different cases, according to the order of multiplicity of the singular values λ_i's of $\mathbf{\Lambda}$:

1. $\lambda_1 \neq \lambda_2 \neq \lambda_3$ $(\lambda_1 > \lambda_2 > \lambda_3)$
2. $\lambda_1 = \lambda_2 \neq \lambda_3$ $(\lambda_1 = \lambda_2 > \lambda_3)$ or $\lambda_1 \neq \lambda_2 = \lambda_3$ $(\lambda_1 > \lambda_2 = \lambda_3)$
3. $\lambda_1 = \lambda_2 = \lambda_3$.

We will consider these three cases separately in order to solve equation (5.4) for translation vector $\vec{t}$ and rotation matrix $\mathbf{R}$.

5.1 Case #1: $\lambda_1 \neq \lambda_2 \neq \lambda_3$ $(\lambda_1 > \lambda_2 > \lambda_3)$

If we assume $d^{'} = \pm\lambda_1$, the equation (5.11) will yield:

$$\left\{\begin{array}{l} (\lambda_1^2 - \lambda_3^2)a_1^2 = 0 \\ (\lambda_1^2 - \lambda_2^2)a_1^2 = 0 \\ (\lambda_1^2 - \lambda_3^2)a_2^2 + (\lambda_1^2 - \lambda_2^2)a_3^2 = 0 \end{array}\right. \tag{5.13}$$

The first two equations in (5.13) imply $a_1 = 0$. In addition, as $\lambda_1 > \lambda_2 > \lambda_3$, $(\lambda_1^2 - \lambda_3^2)$ and $(\lambda_1^2 - \lambda_2^2)$ will be both positive in the third equation in (5.13). Thus, we get $a_2 = a_3 = 0$, which is impossible for the assumption that $\vec{n}'$ has a unit norm.

If we assume $d^{'} = \pm\lambda_3$, the equation (5.11) will yield:

$$\left\{\begin{array}{l} (\lambda_3^2 - \lambda_1^2)a_3^2 = 0 \\ (\lambda_3^2 - \lambda_2^2)a_1^2 + (\lambda_3^2 - \lambda_1^2)a_2^2 = 0 \\ (\lambda_3^2 - \lambda_2^2)a_3^2 = 0 \end{array}\right. . \tag{5.14}$$

The first and the third equations in (5.14) imply $a_3 = 0$. In addition, as $\lambda_1 > \lambda_2 > \lambda_3$, $(\lambda_3^2 - \lambda_2^2)$ and $(\lambda_3^2 - \lambda_1^2)$ will be both negative in the second equation in (5.14). Thus, we get $a_1 = a_2 = 0$, which is again impossible for the assumption that $\vec{n}'$ has a unit norm.

We conclude that the only solution to equation (5.12) is $d' = \pm\lambda_2$. Under the condition that $\lambda_1 \neq \lambda_3$ and $a_1^2 + a_2^2 + a_3^2 = 1$ ($\vec{n}'$ has the unit norm), we can express a_1, a_2 and a_3 using equation (5.11) as follows:

$$\left\{\begin{array}{l} a_1 = \varepsilon_1\sqrt{\frac{\lambda_1^2-\lambda_2^2}{\lambda_1^2-\lambda_3^2}} \\ a_2 = 0 \\ a_3 = \varepsilon_3\sqrt{\frac{\lambda_2^2-\lambda_3^2}{\lambda_1^2-\lambda_3^2}} \end{array}\right. \qquad \varepsilon_1, \varepsilon_3 = \pm 1 \quad . \tag{5.15}$$

1. Case #1, sub-case: $d' = \lambda_2$.

Since $a_2 = 0$ and $d' = \lambda_2$, according to the second equation in (5.8) we obtain: $\vec{e_2} = \mathbf{R}^{'}\vec{e_2}$, which implies that $\mathbf{R}^{'}$ is a rotation matrix around axis $\vec{e_2}$ (recall that

$\vec{e}_2 = (0,1,0)^T$):

$$\begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}.$$

Substituting $a_2 = 0$, $d' = \lambda_2$ and $\mathbf{R}' = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}$ into equation (5.10) and solving for $\sin\theta$ and $\cos\theta$, we have:

$$\left\{ \begin{array}{l} \sin\theta = (\lambda_1 - \lambda_3)\frac{a_1 a_3}{\lambda_2} \\ \cos\theta = \frac{\lambda_1 a_3^2 + \lambda_3 a_1^2}{\lambda_2} \end{array} \right. . \tag{5.16}$$

Substituting equation (5.15) into equation (5.16), we have:

$$\left\{ \begin{array}{l} \sin\theta = \varepsilon_1 \varepsilon_3 \frac{\sqrt{(\lambda_1^2 - \lambda_2^2)(\lambda_2^2 - \lambda_3^2)}}{(\lambda_1 + \lambda_3)\lambda_2} \\ \cos\theta = \frac{\lambda_2^2 + \lambda_1 \lambda_3}{(\lambda_1 + \lambda_3)\lambda_2} \end{array} \right. .$$

Substituting equation (5.16) into equation (5.8) with the condition that $a_1^2 + a_2^2 + a_3^2 = 1$ ($a_2 = 0$), we get:

$$\vec{t'} = (\lambda_1 - \lambda_3) \begin{pmatrix} a_1 \\ 0 \\ -a_3 \end{pmatrix}.$$

2. Case #1, sub-case: $d' = -\lambda_2$.

Since $a_2 = 0$ and $d' = -\lambda_2$, according to the second equation in (5.8), we obtain: $-\vec{e_2} = \mathbf{R}'\vec{e_2}$, which implies that

$$\mathbf{R}' = \begin{pmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & -1 & 0 \\ \sin\varphi & 0 & -\cos\varphi \end{pmatrix}.$$

Following the same procedure as for the case of $d' = \lambda_2$ we solve for $\sin\varphi$ and $\cos\varphi$:

$$\left\{ \begin{array}{l} \sin\varphi = (\lambda_1 + \lambda_3)\frac{a_1 a_3}{\lambda_2} = \varepsilon_1 \varepsilon_3 \frac{\sqrt{(\lambda_1^2 - \lambda_2^2)(\lambda_2^2 - \lambda_3^2)}}{(\lambda_1 - \lambda_3)\lambda_2} \\ \cos\varphi = \frac{\lambda_3 a_1^2 + \lambda_1 a_3^2}{\lambda_2} = \frac{\lambda_1 \lambda_3 - \lambda_2^2}{(\lambda_1 - \lambda_3)\lambda_2} \end{array} \right. . \tag{5.17}$$

Substituting these values into equation (5.8), we get:

$$\vec{t'} = (\lambda_1 + \lambda_3) \begin{pmatrix} a_1 \\ 0 \\ a_3 \end{pmatrix}.$$

5.2 Case #2: $\lambda_1 = \lambda_2 \neq \lambda_3$ $(\lambda_1 = \lambda_2 > \lambda_3)$ or $\lambda_1 \neq \lambda_2 = \lambda_3$ $(\lambda_1 > \lambda_2 = \lambda_3)$

We consider the sub-case of $\lambda_1 = \lambda_2 \neq \lambda_3$ (the other sub-case will be equivalent).

If we assume $d' = \pm\lambda_3$, equations (5.11) will yield equations (5.14) and thus $a_1 = a_2 = a_3 = 0$, which is again impossible for the assumption that $\vec{n}'$ has a unit norm.

Thus, we have $d' = \pm\lambda_1 = \pm\lambda_2$. Let's use $d' = \pm\lambda_2$ for consistency with the previous derivation. Under the condition $\lambda_1 = \lambda_2 \neq \lambda_3$ and $a_1^2 + a_2^2 + a_3^2 = 1$ ($\vec{n}'$ has a unit norm), we can express a_1, a_2 and a_3 using equation (5.11) as follows:

$$\begin{cases} a_1 = 0 \\ a_2 = 0 \\ a_3 = \pm 1 \end{cases}. \tag{5.18}$$

1. Case #2, sub-case: $d' = \lambda_2$.

Since we have $a_2 = 0$ and $d' = \lambda_2$, we obtain: $\vec{e_2} = \mathbf{R}'\vec{e_2}$ as in case #1, and following the same procedure, we get:

$$\begin{cases} \sin\theta = 0 \\ \cos\theta = 1 \end{cases}.$$

Substituting these values into equation (5.8), we get:

$$\vec{t'} = (\lambda_3 - \lambda_1) \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix}.$$

2. Case #2, sub-case: $d' = -\lambda_2$.

Again, since we have $a_2 = 0$ and $d' = -\lambda_2$, we obtain: $-\vec{e_2} = \mathbf{R}'\vec{e_2}$ as in case #1, and following the same procedure, we get:

$$\left\{ \begin{array}{l} \sin\varphi = 0 \\ \cos\varphi = -1 \end{array} \right. .$$

Substituting these values into equation (5.8), we get:

$$\vec{t'} = (\lambda_3 + \lambda_1) \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix}.$$

5.3 Case #3: $\lambda_1 = \lambda_2 = \lambda_3$

1. Case #3, sub-case: $d' = \lambda_1 = \lambda_2 = \lambda_3$.

In this case, a_1, a_2 and a_3 are undefined according to equation (5.11), which means that under the condition that $d' = \lambda_1 = \lambda_2 = \lambda_3$, equations (5.10) and (5.8) must be satisfied for all values of a_1, a_2 and a_3. Thus, equation (5.10) leads to $\mathbf{R}' = \mathbf{I}$ and (5.8) leads to $\vec{t'} = \vec{0}$.

2. Case #3, sub-case: $d' = -\lambda_1 = -\lambda_2 = -\lambda_3$.

In this case, a_1, a_2 and a_3 are again undefined according to equation (5.11). Furthermore, under the condition that $d' = -\lambda_1 = -\lambda_2 = -\lambda_3$, equation (5.10) becomes:

$$\mathbf{R}'(a_j\vec{e}_i - a_i\vec{e}_j) = -(a_j\vec{e}_i - a_i\vec{e}_j)$$

which must be satisfied for all values of a_1, a_2 and a_3. Thus, $\mathbf{R}'$ is a rotation matrix in the object plane, i.e., around the axis that is orthogonal to the normal $\vec{n}$. Because that normal $\vec{n}$ is undefined here, the solution in this case is undetermined.

5.4 Summary of solutions

On the following pages, we list solutions to all three cases.

1. Case #1:

$$\left\{\begin{array}{ll}\mathbf{R}'=\begin{pmatrix}\frac{\lambda_2^2+\lambda_1\lambda_3}{(\lambda_1+\lambda_3)\lambda_2} & 0 & -\varepsilon_1\varepsilon_3\frac{\sqrt{(\lambda_1^2-\lambda_2^2)(\lambda_2^2-\lambda_3^2)}}{(\lambda_1+\lambda_3)\lambda_2}\\ 0 & 1 & 0\\ \varepsilon_1\varepsilon_3\frac{\sqrt{(\lambda_1^2-\lambda_2^2)(\lambda_2^2-\lambda_3^2)}}{(\lambda_1+\lambda_3)\lambda_2} & 0 & \frac{\lambda_2^2+\lambda_1\lambda_3}{(\lambda_1+\lambda_3)\lambda_2}\end{pmatrix} & \\ \vec{t}'=(\lambda_1-\lambda_3)\begin{pmatrix}\varepsilon_1\sqrt{\frac{\lambda_1^2-\lambda_2^2}{\lambda_1^2-\lambda_3^2}}\\ 0\\ -\varepsilon_3\sqrt{\frac{\lambda_2^2-\lambda_3^2}{\lambda_1^2-\lambda_3^2}}\end{pmatrix} & d'>0, \varepsilon_1, \varepsilon_3=\pm 1.\\ \vec{n}'=\begin{pmatrix}\varepsilon_1\sqrt{\frac{\lambda_1^2-\lambda_2^2}{\lambda_1^2-\lambda_3^2}}\\ 0\\ -\varepsilon_3\sqrt{\frac{\lambda_2^2-\lambda_3^2}{\lambda_1^2-\lambda_3^2}}\end{pmatrix} & \end{array}\right.$$

$$\left\{\begin{array}{ll}\mathbf{R}'=\begin{pmatrix}\frac{\lambda_1\lambda_3-\lambda_2^2}{(\lambda_1-\lambda_3)\lambda_2} & 0 & \varepsilon_1\varepsilon_3\frac{\sqrt{(\lambda_1^2-\lambda_2^2)(\lambda_2^2-\lambda_3^2)}}{(\lambda_1-\lambda_3)\lambda_2}\\ 0 & -1 & 0\\ \varepsilon_1\varepsilon_3\frac{\sqrt{(\lambda_1^2-\lambda_2^2)(\lambda_2^2-\lambda_3^2)}}{(\lambda_1-\lambda_3)\lambda_2} & 0 & -\frac{\lambda_1\lambda_3-\lambda_2^2}{(\lambda_1-\lambda_3)\lambda_2}\end{pmatrix} & \\ \vec{t}'=(\lambda_1+\lambda_3)\begin{pmatrix}\varepsilon_1\sqrt{\frac{\lambda_1^2-\lambda_2^2}{\lambda_1^2-\lambda_3^2}}\\ 0\\ \varepsilon_3\sqrt{\frac{\lambda_2^2-\lambda_3^2}{\lambda_1^2-\lambda_3^2}}\end{pmatrix} & d'<0, \varepsilon_1, \varepsilon_3=\pm 1.\\ \vec{n}'=\begin{pmatrix}\varepsilon_1\sqrt{\frac{\lambda_1^2-\lambda_2^2}{\lambda_1^2-\lambda_3^2}}\\ 0\\ -\varepsilon_3\sqrt{\frac{\lambda_2^2-\lambda_3^2}{\lambda_1^2-\lambda_3^2}}\end{pmatrix} & \end{array}\right.$$

2. Case #2:

$$\left\{\begin{array}{l} \mathbf{R}' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\[2ex] \vec{t}' = (\lambda_3 - \lambda_1) \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix} \qquad\qquad d' > 0. \\[2ex] \vec{n}' = \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix} \end{array}\right.$$

$$\left\{\begin{array}{l} \mathbf{R}' = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\[2ex] \vec{t}' = (\lambda_3 + \lambda_1) \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix} \qquad\qquad d' < 0. \\[2ex] \vec{n}' = \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix} \end{array}\right.$$

3. Case #3:

$$\begin{cases} \mathbf{R}' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \\ \vec{t}' = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & d' > 0. \\ \vec{n}' \quad \text{undetermined} & \end{cases} \tag{5.19}$$

$$\begin{cases} \mathbf{R}' \quad \text{undetermined} & \\ \vec{t}' \quad \text{undetermined} & d' < 0. \\ \vec{n}' \quad \text{undetermined} & \end{cases} \tag{5.20}$$

After obtaining $\mathbf{R}'$, $\vec{t}'$ and $\vec{n}'$, $\mathbf{R}$, $\vec{t}$ and $\vec{n}$ can be calculated from equation (5.6).

Clearly, we will get multiple solutions in case #1 and case #2. How can we determine which one is the right solution? We explain this in detail in the next section.

5.5 Implementation of the method of Faugeras and Lustman

There are several issues that need to be resolved when implementing the method derived above:

1. First, we calculate normalized homographies $\hat{\mathbf{H}}$ between a reference frame (typically the first frame) and the consecutive frames. Then, we apply the method of Faugeras and Lustman to decompose the homographies into 3D rotations and translations. Every estimation result is relative to the coordinate system of the reference camera position (typically the initial camera position).

2. Equations (5.20) indicate that we cannot solve for $\mathbf{R}$, $\vec{t}$ and $\vec{n}$, when $d' < 0$ and $\lambda_1 = \lambda_2 = \lambda_3$. This case corresponds to the situation when a transparent

plane is observed from two opposite sides and at the same distance, which will not happen under our assumption that the target is not transparent and we are always viewing the target from one side.

3. After determining the sign of d', we have to choose from possible solutions in cases #1 and #2. Since we know we are viewing the object from one side, and the object is oriented towards the camera, we know that c in the normal $\vec{n} = (a, b, c)^T$ to the object will always be negative. Thus, we choose solutions with negative c.

4. After choosing solutions with negative c, we will still have two solutions in case #1, and we have to determine one unique solution. By applying the algorithm between an arbitrary frame and the reference frame, one quantity will remain the same, namely the normal vector $\vec{n}_{ref}$ of the object in the reference camera coordinate system. By finding the normal vector $\vec{n}_{ref}$, we are able to choose the correct solution to the problem, i.e., the solution with the minimum norm $\|\vec{n} - \vec{n}_{ref}\|$. In order to find $\vec{n}_{ref}$, we use a third image of the same object from a different camera position. Let's denote the normal vector estimated between the reference frame and the second frame as $\vec{n}_1$ and $\vec{n}_2$, and the normal vector estimated between the reference frame and the third frame as $\vec{n}'_1$ and $\vec{n}'_2$. We will set $\vec{n}_{ref} = (\vec{n}_i - \vec{n}'_j)/2$ $(i, j = 1, 2)$, where $\vec{n}_i$ and $\vec{n}'_j$ are the normal vectors with minimum norm $\|\vec{n}_i - \vec{n}'_j\|$ $(i, j = 1, 2)$.

5. Due to estimation errors, even if camera hasn't moved, the eigenvalues λ_i of $\mathbf{\Lambda}$ in equation (5.7) might be slightly different. In order to tolerate small errors, we threshold absolute eigenvalue differences as follows: if $|\lambda_i - \lambda_j| > \epsilon$ $(i \neq j)$, λ_i and λ_j are considered different, otherwise they are considered identical. The threshold ϵ is an important parameter that needs to be carefully selected. This

will be discussed in Chapter 6.

Chapter 6

Experimental results

In this chapter, we evaluate the effectiveness of the method described in Chapter 5 in estimation of 3D camera motion. First, we present results for synthetic data simulating translation and rotation of a planar object. Then, we present results of a "ground truth" experiment in which the camera is translated and/or rotated with precise increments in front of a planar object (chessboard pattern). At last, we discuss results of a direct comparison of camera-derived 3-D translations with those derived from accelerometer measurements.

6.1 Synthetic data experiments

We first validate the algorithm on simulated data, i.e., data generated in software without ever using a real camera. In this simulation, we aligned the camera coordinate system with the world coordinate system so that the image plane is in the (X_{world}, Y_{world}) plane of the world coordinate system for convenience (Figure 6·1). We generated a 2D lattice to simulate feature points of the chessboard pattern. The lattice is located on a plane with normal $\vec{n} = (0, 0, -1)^T$ and at distance $d = 1000$ from the origin of the camera coordinate system. We simulated a camera with the following intrinsic matrix:

$$\mathbf{A} = \begin{bmatrix} 500 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.1}$$

In addition, we set threshold ϵ described in Section 5.5 to 10^{-10}. We projected the synthetic lattice onto the image plane to produce a set of feature points in a video frame. Then, we changed the position/orientation of the synthetic lattice and projected it again onto the image plane. After obtaining two frames of feature points projected from two positions/orientations of the synthetic lattice, we calculated the normalized homography between them and then applied the algorithm described in Chapter 5 to estimate 3D translation and 3D rotation. By specifying different positions and orientations of the synthetic lattice, we have generated ground truth to compare our estimation with. We denote our ground truth translation vector as $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T$, and rotation vector as $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T$, where both vectors apply between the k^{th} frame and the first frame. The translation vector has no units, while the rotation vector is expressed in degrees.

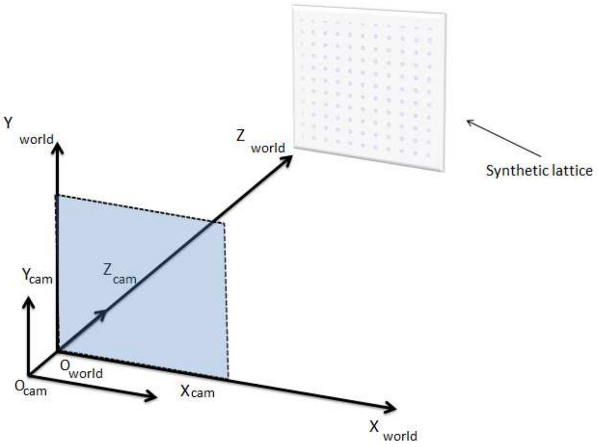


Figure 6·1: Synthetic lattice with the camera coordinate system aligned to the world coordinate system.

We have tested this method in the following seven cases of translation and rotation of the synthetic pattern:

- **Case 1:** Translation along x axis by vector $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (1000k, 0, 0)^T$ (Figure 6·2).

- **Case 2:** Translation along y axis by vector $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 1000k, 0)^T$ (Figure 6·3).
- **Case 3:** Translation along z axis by vector $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 0, 1000k)^T$ (Figure 6·4).
- **Case 4:** Rotation around x axis by vector $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (k-5, 0, 0)^T$ (Figure 6·5).
- **Case 5:** Rotation around y axis by vector $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (0, k-5, 0)^T$(Figure 6·6).
- **Case 6:** Rotation around z axis by vector $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (0, 0, k-5)^T$ (Figure 6·7).
- **Case 7:** Arbitrary translation and rotation.

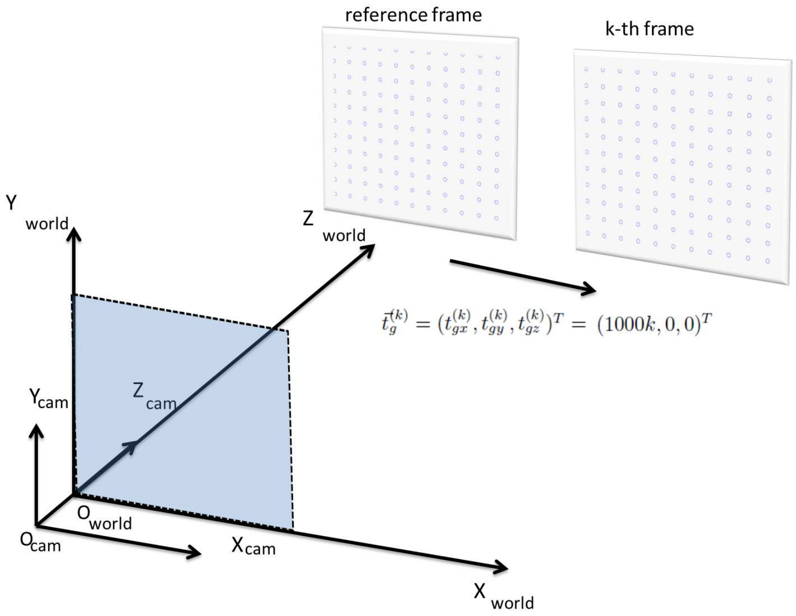

reference frame
k-th frame
$\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (1000k, 0, 0)^T$
Y world
Z world
Y cam
Z cam
O world
O cam
X cam
X world

Figure 6·2: Translation of the synthetic lattice along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (1000k, 0, 0)^T$.

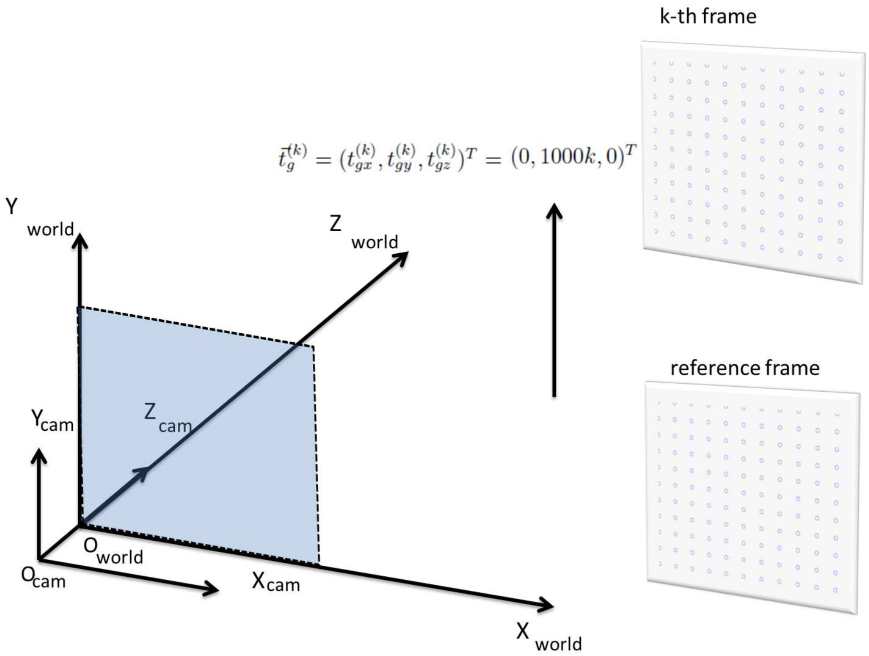

k-th frame
$\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 1000k, 0)^T$
reference frame
Y world
Z world
Y cam
Z cam
O world
O cam
X cam
X world

Figure 6·3: Translation of the synthetic lattice along y axis of the world coordinate system by $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 1000k, 0)^T$.

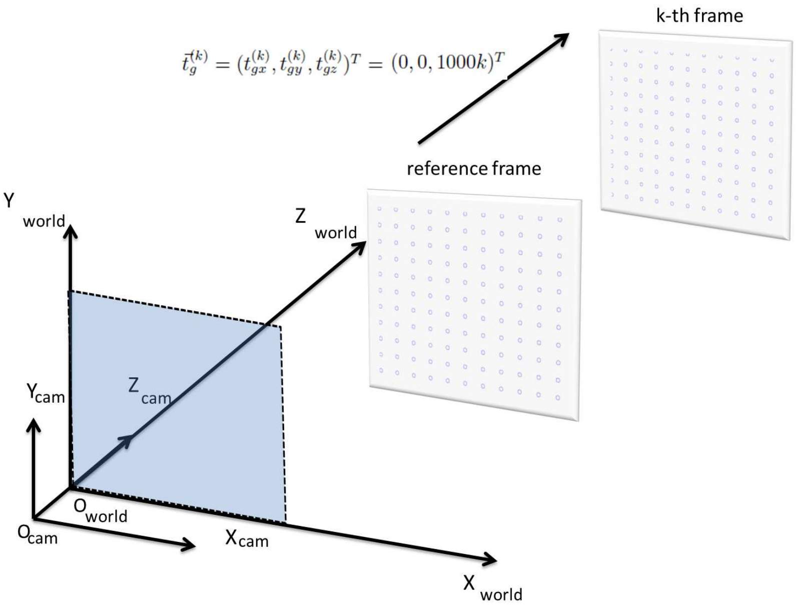

k-th frame
$\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 0, 1000k)^T$
reference frame
Y world
Z world
Y cam
Z cam
O world
O cam
X cam
X world

Figure 6·4: Translation of the synthetic lattice along z axis of the world coordinate system by $\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (0, 0, 1000k)^T$.

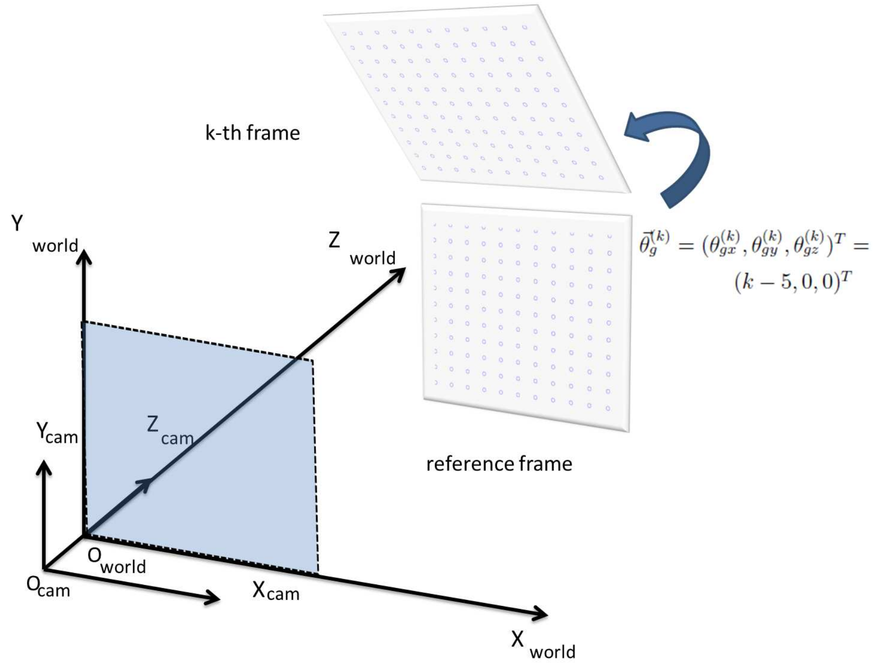

k-th frame
$\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T =$
$(k-5,0,0)^T$
Y world
Z world
reference frame
Y cam
Z cam
O world
O cam
X cam
X world

Figure 6·5: Rotation of the synthetic lattice around x axis of the world coordinate system by $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (k-5,0,0)^T$.

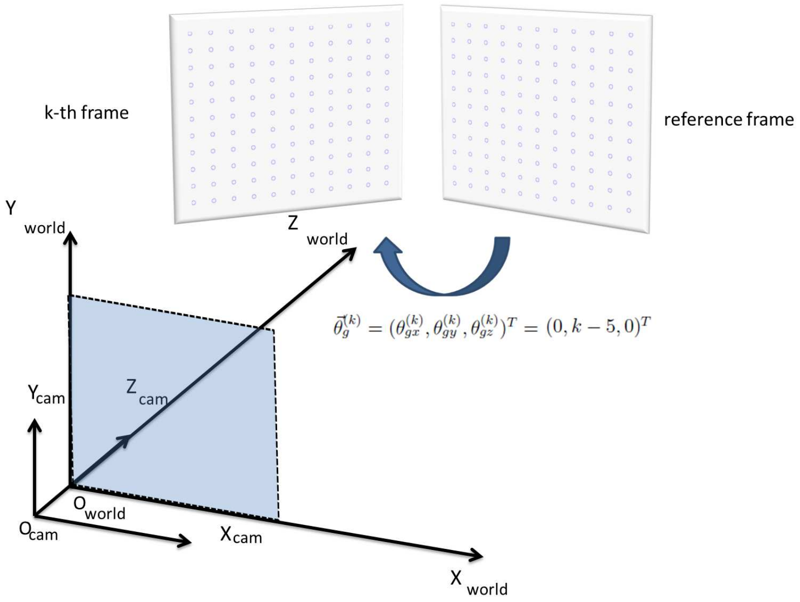

k-th frame
reference frame
Y world
Z world
$\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (0,k-5,0)^T$
Y cam
Z cam
O world
O cam
X cam
X world

Figure 6·6: Rotation of the synthetic lattice around y axis of the world coordinate system by $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (0,k-5,0)^T$.

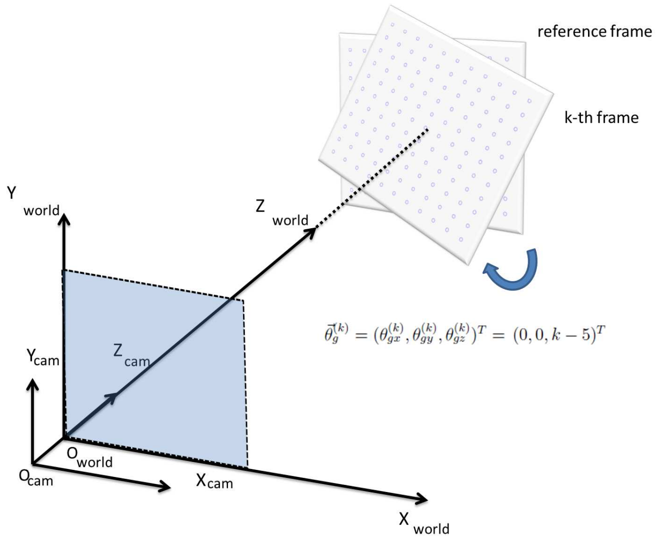
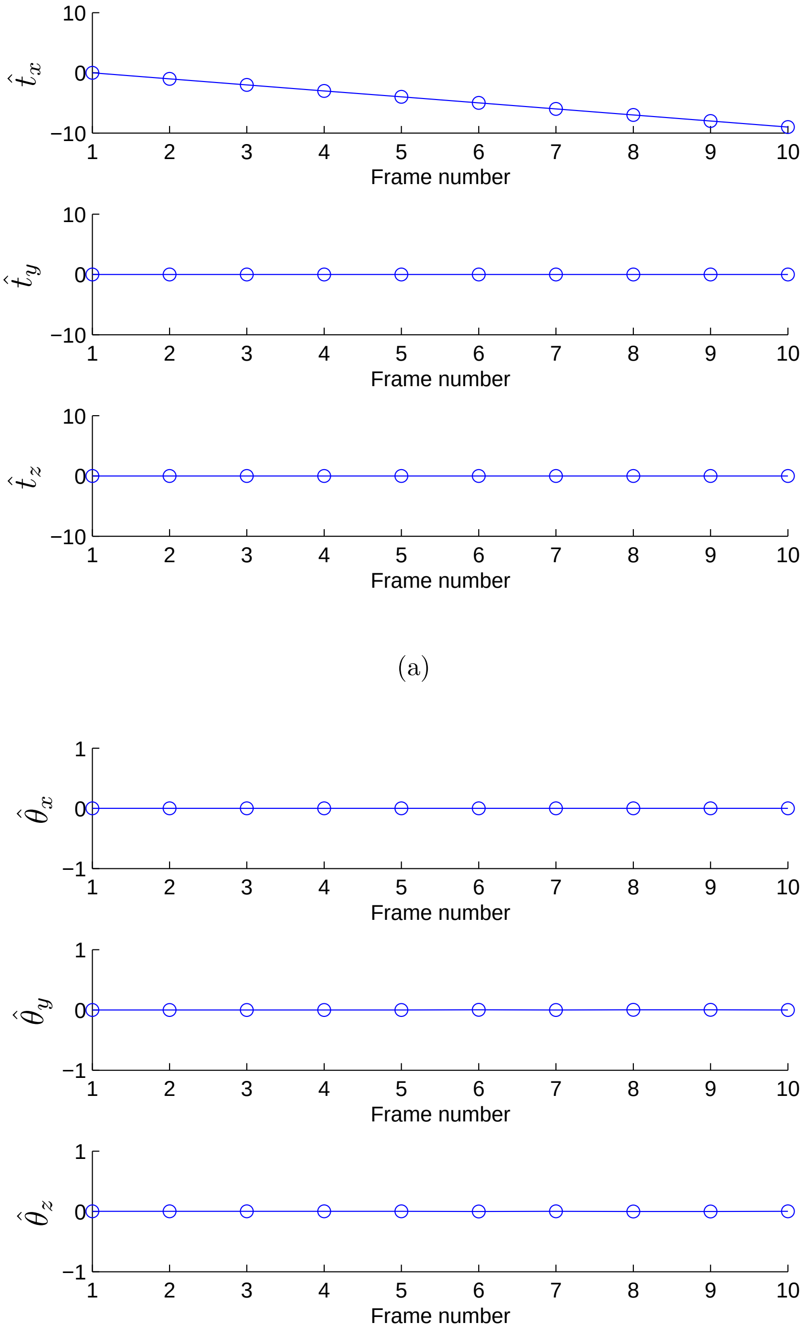


Figure 6·7: Rotation of the synthetic lattice around z axis of the world coordinate system by $\vec{\theta}_g^{(k)} = (\theta_{gx}^{(k)}, \theta_{gy}^{(k)}, \theta_{gz}^{(k)})^T = (0, 0, k-5)^T$.

Figures 6·8 to 6·14 illustrate the estimation results of translation and rotation in the above cases. Clearly, in each case the method of Faugeras and Lustman (Faugeras and Lustman, 1988) provides accurate estimations of translation and rotation on synthetic data.

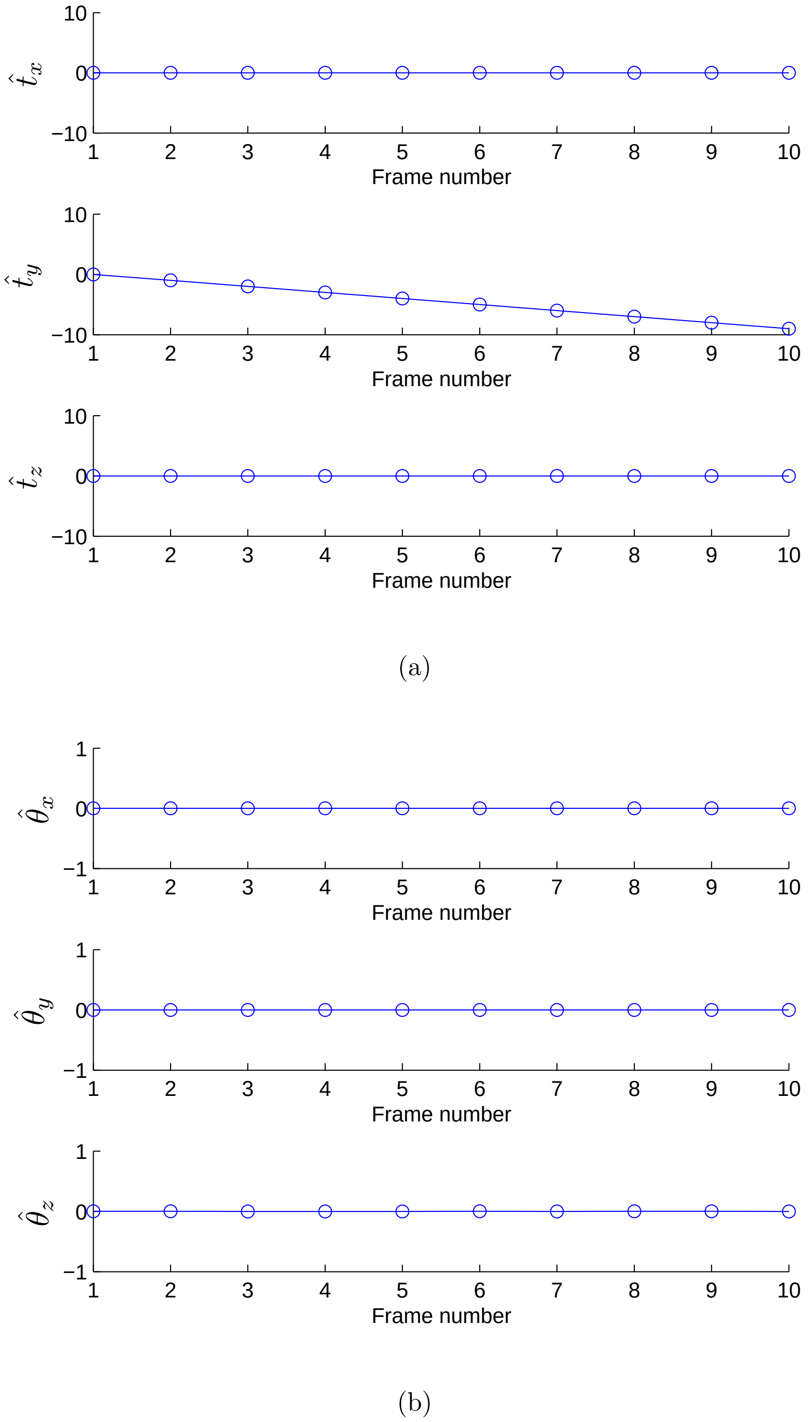
For quantitative assessment, we calculated average root-mean-squared error of



(a)

(b)

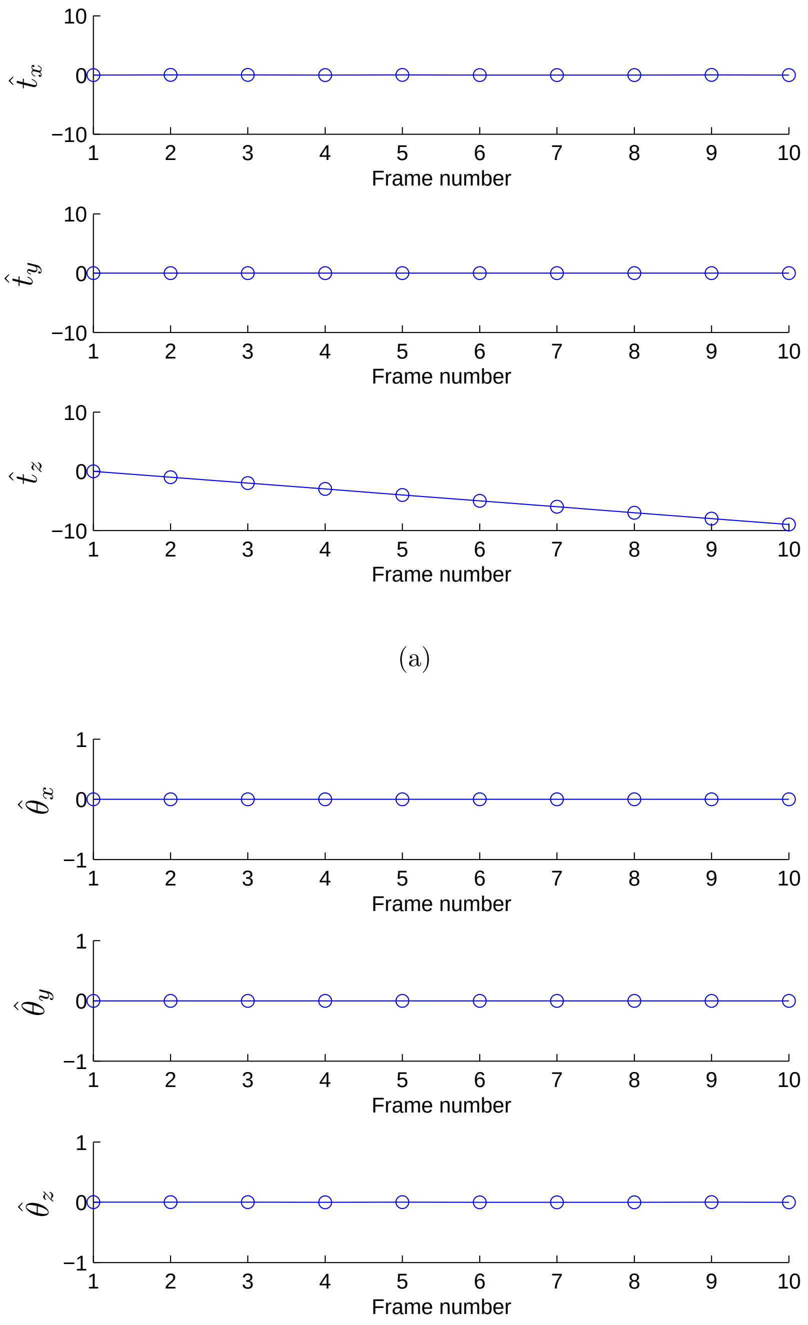
Figure 6·8: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (1000k, 0, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, 0)^T$ in case 1.



(a)

(b)

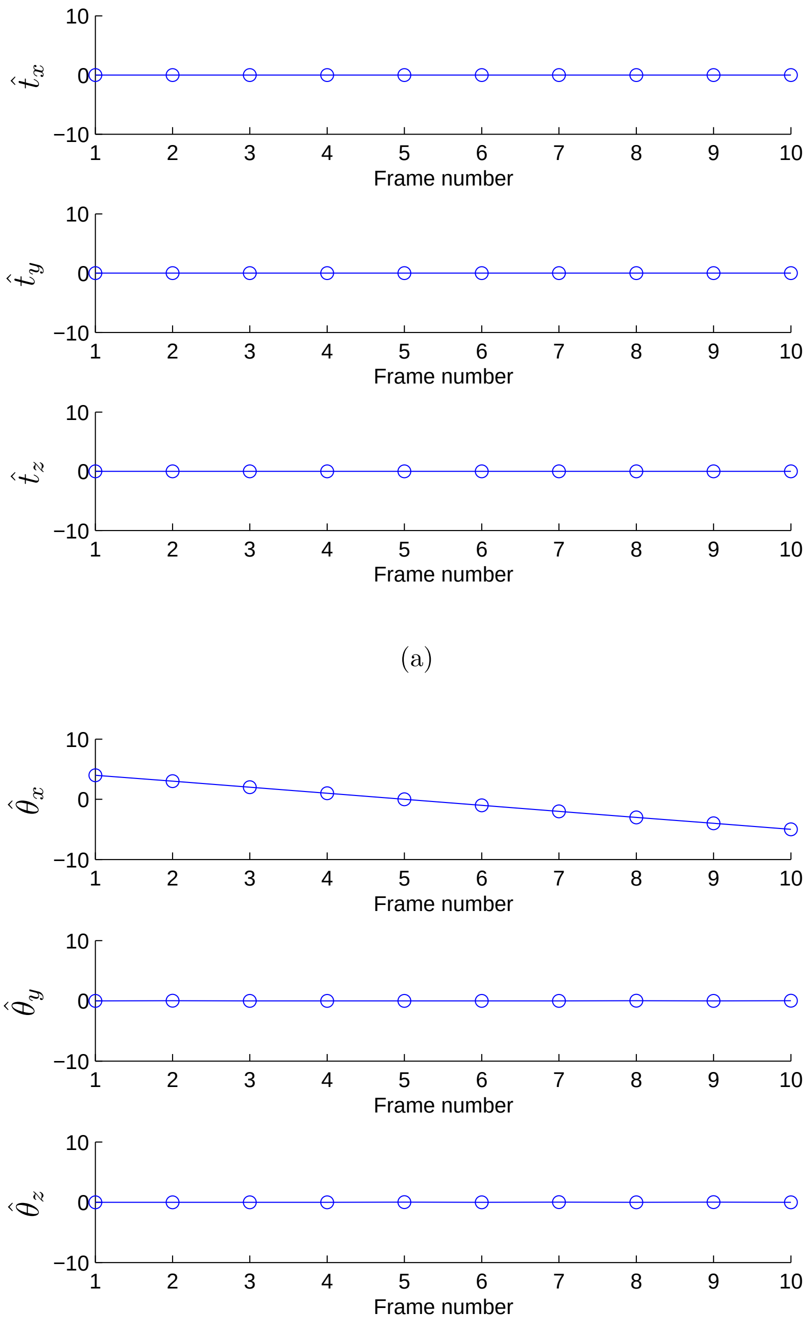
Figure 6·9: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 1000k, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, 0)^T$ in case 2.



(a)

(b)

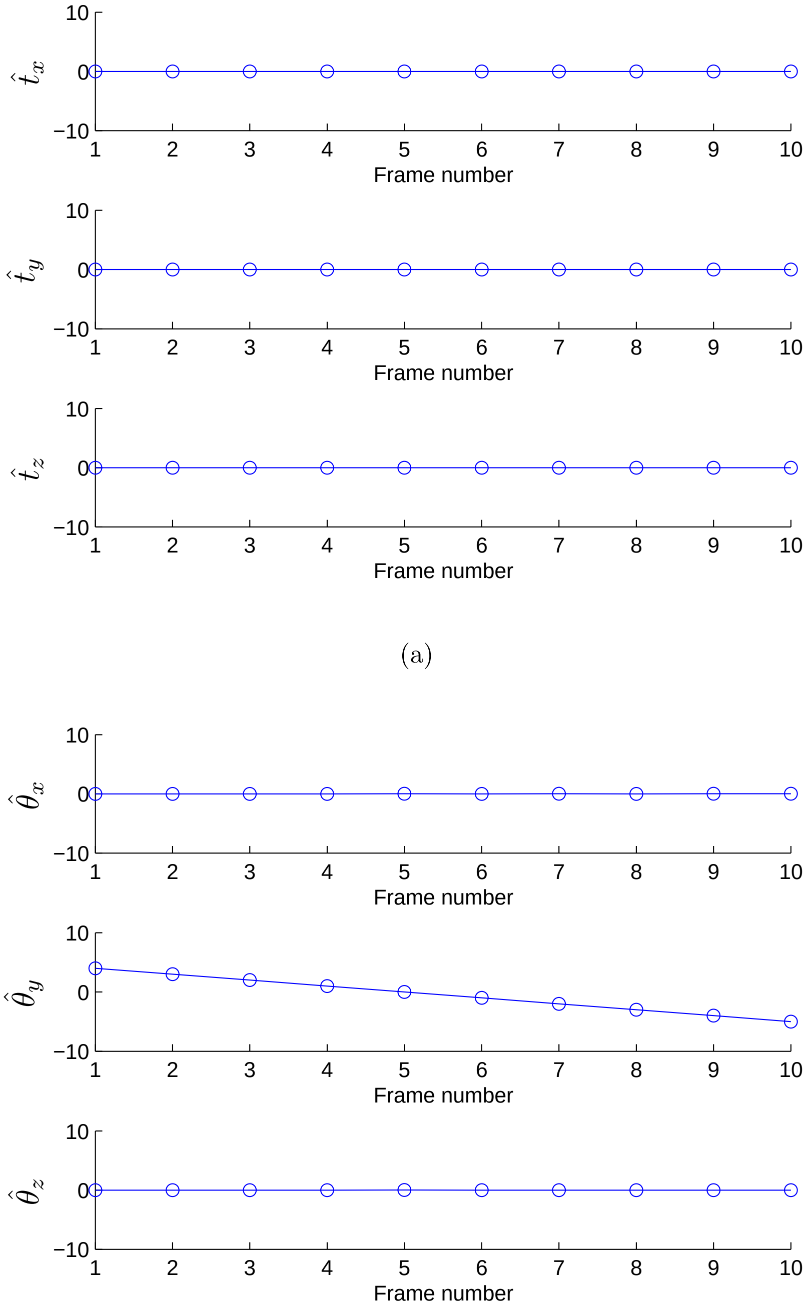
Figure 6·10: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 0, 1000k)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, 0)^T$ in case 3.



(a)

(b)

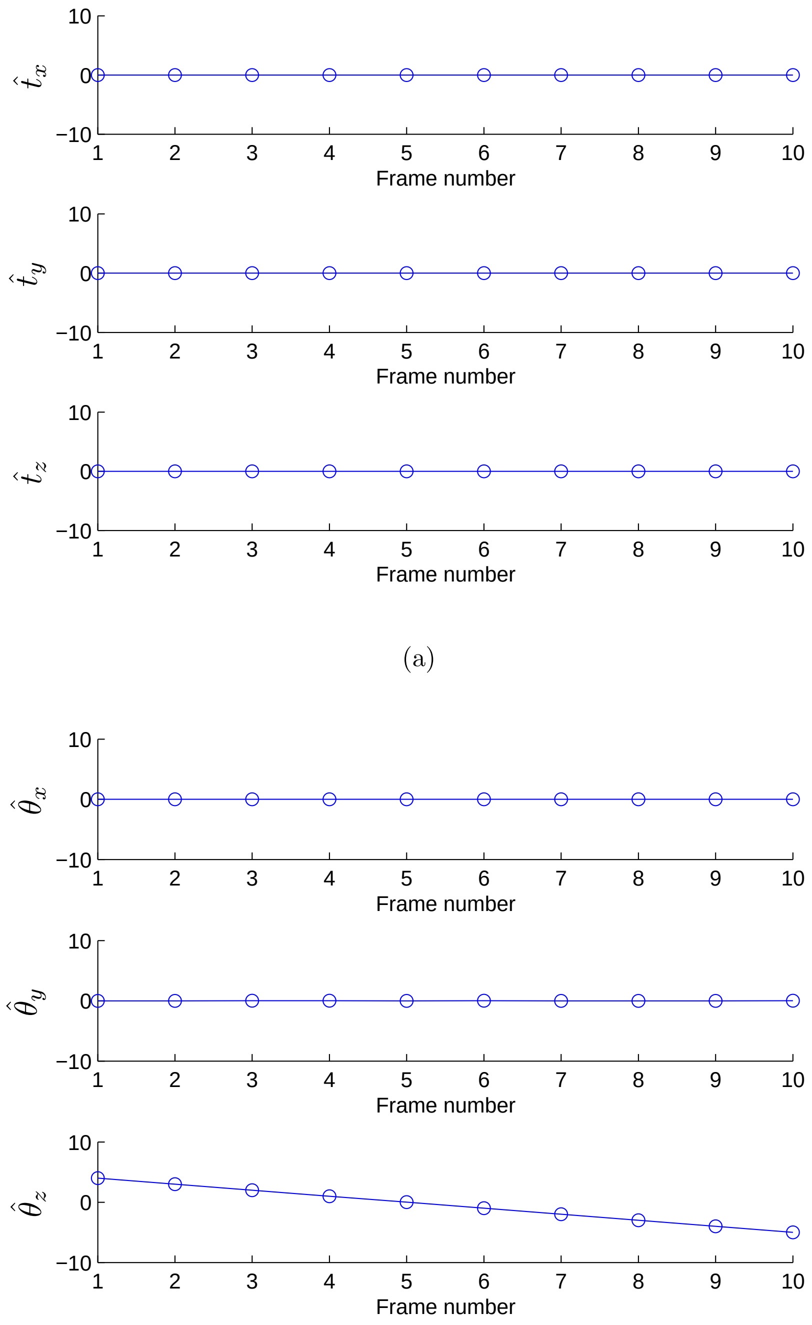
Figure 6·11: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0,0,0)^T$ and rotation $\vec{\theta}_g^{(k)} = (k-5,0,0)^T$ in case 4.



(a)

(b)

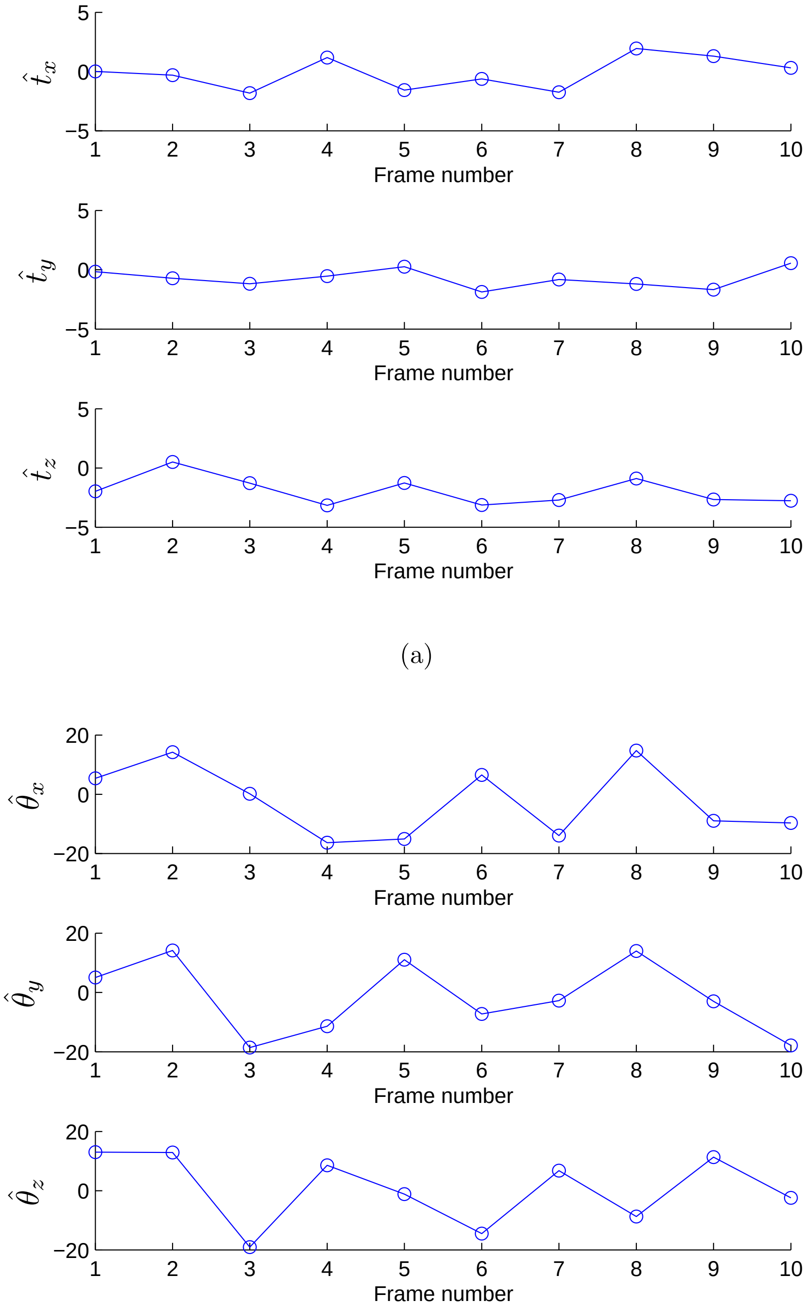
Figure 6·12: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 0, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, k-5, 0)^T$ in case 5.



(a)

(b)

Figure 6·13: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under ground truth translation $\vec{t}_g^{(k)} = (0, 0, 0)^T$ and rotation $\vec{\theta}_g^{(k)} = (0, 0, k - 5)^T$ in case 6.



(a)

(b)

Figure 6·14: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ for arbitrary camera motion in case 7.

translation and rotation in every case as follows:

$$\begin{cases} \bar{\varepsilon}_{tx} = \sqrt{\frac{1}{10}\sum\limits_{k=1}^{10}(\hat{t}_x^{(k)} - t_{gx}^{(k)})^2} \\ \\ \bar{\varepsilon}_{ty} = \sqrt{\frac{1}{10}\sum\limits_{k=1}^{10}(\hat{t}_y^{(k)} - t_{gy}^{(k)})^2} \\ \\ \bar{\varepsilon}_{tz} = \sqrt{\frac{1}{10}\sum\limits_{k=1}^{10}(\hat{t}_z^{(k)} - t_{gz}^{(k)})^2} \\ \\ \bar{\varepsilon}_{\theta x} = \sqrt{\frac{1}{10}\sum\limits_{k=1}^{10}(\hat{\theta}_x^{(k)} - \theta_{gx}^{(k)})^2} \\ \\ \bar{\varepsilon}_{\theta y} = \sqrt{\frac{1}{10}\sum\limits_{k=1}^{10}(\hat{\theta}_y^{(k)} - \theta_{gy}^{(k)})^2} \\ \\ \bar{\varepsilon}_{\theta z} = \sqrt{\frac{1}{10}\sum\limits_{k=1}^{10}(\hat{\theta}_z^{(k)} - \theta_{gz}^{(k)})^2} \end{cases}$$

where $(\hat{t_x}^{(k)}, \hat{t_y}^{(k)}, \hat{t_y}^{(k)})$ and $(\hat{\theta_x}^{(k)}, \hat{\theta_y}^{(k)}, \hat{\theta_z}^{(k)})$ are the estimated translation and rotation vectors, respectively. As can be seen in Table 6.1, these estimation errors are extremely small. This was to be expected since the error is only limited by the 64-bit precision of the computer we ran the experiments on. However, in cases 4, 5 and 6 the estimation errors for translation are exactly zero because in these three cases of pure rotation the three singular values of $\mathbf{\Lambda}$ are all equal and all the translation vectors are set to zero (equation (5.19)).

In order to test the robustness and accuracy of this method in presence of uncertainty, we added Gaussian-distributed noise to the positions of projected feature points in every video frame. There are two important parameters that affect the estimation results if noise is added: threshold ϵ and standard deviation σ of the noise.

Table 6.1: Root-mean-squared errors between ground truth and estimated translations and rotations for cases 1-7

	t_{gx} only	t_{gy} only	t_{gz} only	θ_{gx} only	θ_{gy} only	θ_{gz} only	free motion
$\bar{\varepsilon}_{tx}$	$2.1e^{-11}$	$4.7e^{-12}$	$3.1e^{-11}$	0	0	0	$1.1e^{-12}$
$\bar{\varepsilon}_{ty}$	$1.3e^{-11}$	$1.9e^{-11}$	$3.9e^{-12}$	0	0	0	$9.9e^{-13}$
$\bar{\varepsilon}_{tz}$	$8.0e^{-12}$	$8.0e^{-12}$	$3.6e^{-11}$	0	0	0	$7.0e^{-13}$
$\bar{\varepsilon}_{\theta x}$	$7.8e^{-13}$	$1.4e^{-11}$	$2.7e^{-13}$	$2.3e^{-15}$	$1.5e^{-15}$	$2.0e^{-15}$	$1.0e^{-14}$
$\bar{\varepsilon}_{\theta y}$	$1.3e^{-11}$	$3.4e^{-13}$	$6.4e^{-13}$	$1.4e^{-15}$	$3.4e^{-15}$	$2.5e^{-15}$	$9.6e^{-15}$
$\bar{\varepsilon}_{\theta z}$	$2.8e^{-12}$	$3.2e^{-13}$	$45.0e^{-14}$	$3.6e^{-15}$	$8.5e^{-16}$	$7.6e^{-16}$	$6.2e^{-15}$

We sampled ϵ in the range from 0 to 1 every 0.01 and σ from 0 to 10 every 0.1. For every pair of values, we applied the algorithm between projections of the reference frame with normal $n = (0, 0, -1)^T$ and $d = 1000$ and the second frame translated by $\vec{t_g} = (200, 200, -200)^T$ and rotated by $\vec{\theta_g} = (15, 15, 15)^T$. We added Gaussian noise with mean 0 and standard deviation σ to the positions of feature points on the image plane and we generated 1000 realizations. We computed root-mean-squared errors of translations and rotations for different values of ϵ and σ. As shown in Figures 6·15 to 6·20, the mean-square error increases as ϵ and σ increase. There is a discontinuity near $\epsilon = 0.14$ in rotation and translation errors because for larger ϵ we will classify the eigenvalues as belonging to case #2 (Section 5.2) rather than case #1 (Section 5.1). The same situation occurs near $\epsilon = 0.30$ where we classify the eigenvalues as belonging to case #3 (Section 5.3) rather than case #2 (Section 5.2). Since the estimated rotations are all set to be zero in cases #2 and #3, we will not see the second discontinuity in the plots of rotation errors. A saturation occurs when ϵ is over 0.3 since all results will be classified as case #3 and all the estimations will be the same. In conclusion, we should set ϵ small enough when we apply this algorithm to real data where noise will be introduced by feature extraction, low resolution of images, etc.

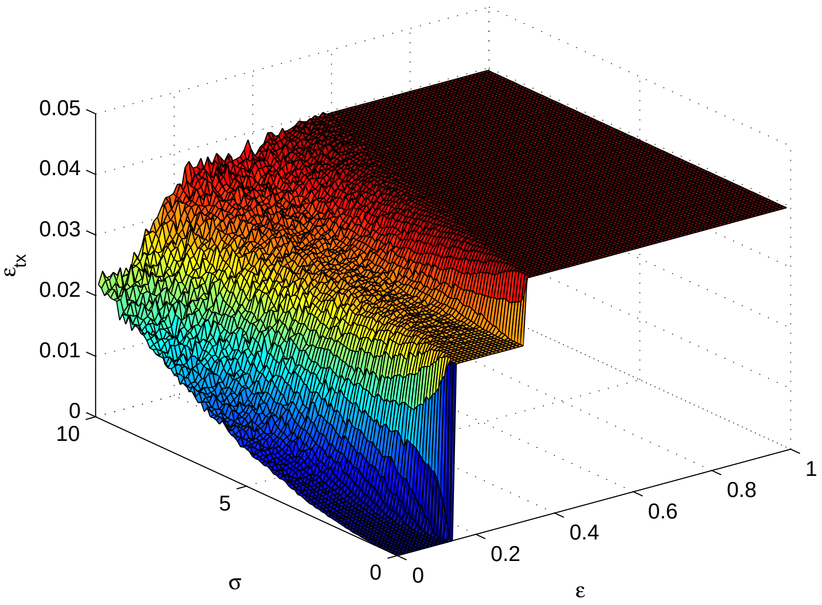


Figure 6·15: Root-mean-squared error of translation in x direction.

We also tested the impact of focal length error on the estimation error of translation and rotation. With the ground truth focal length at 500, we assumed varying focal lengths, from 400 to 600 with an increment of 1, in our estimation procedure. We used translation vector $\vec{t_g} = (100, 100, -100)^T$ and rotation vector $\vec{\theta_g} = (10, 15, 20)^T$ as ground truth. We added no noise to the positions of feature points and used $\epsilon = 10^{-10}$. As we see in Figures 6·21 to 6·26, the root-mean-squared errors of translation and rotation increase as focal length departs from the ground truth. However, overall the root-mean-squared errors are very small. Thus, although the focal length provided by a manufacturer may not be accurate, the estimation results won't be severely affected.

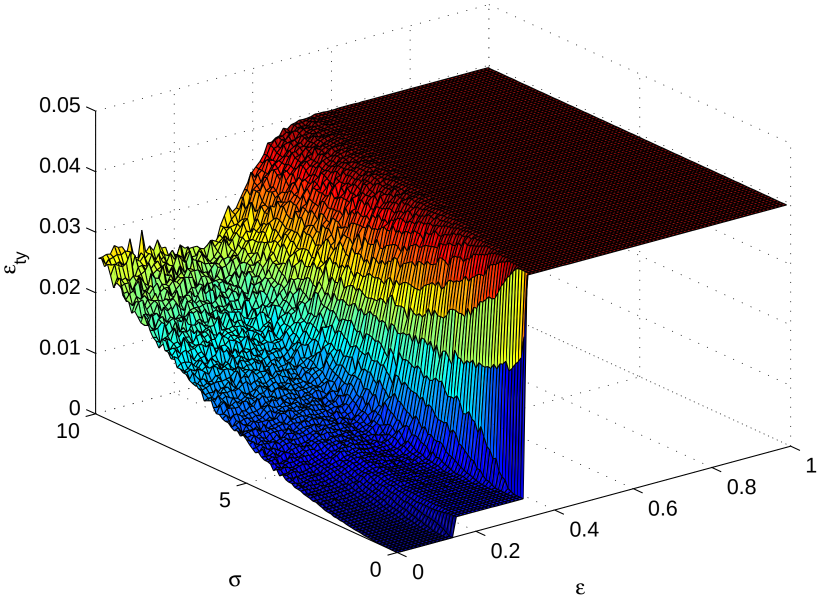


Figure 6·16: Root-mean-squared error of translation in y direction.

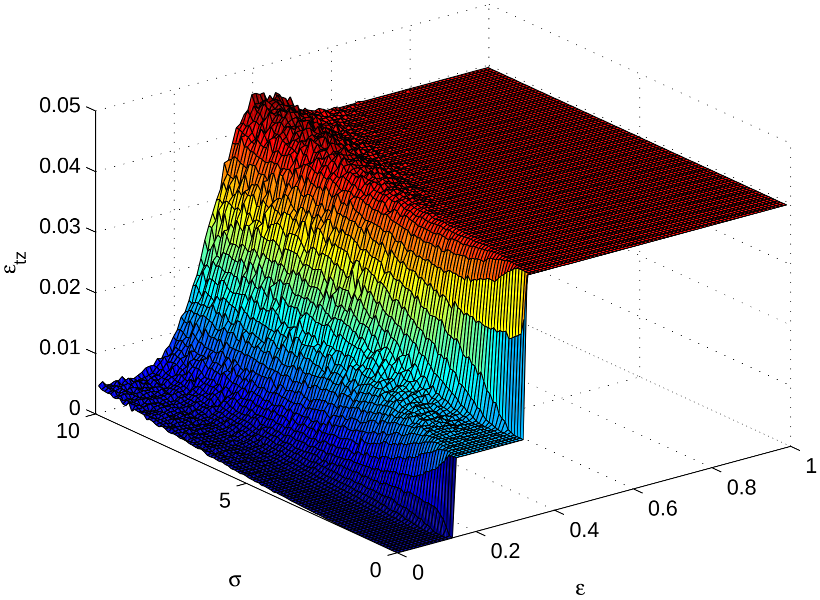


Figure 6·17: Root-mean-squared error of translation in z direction.

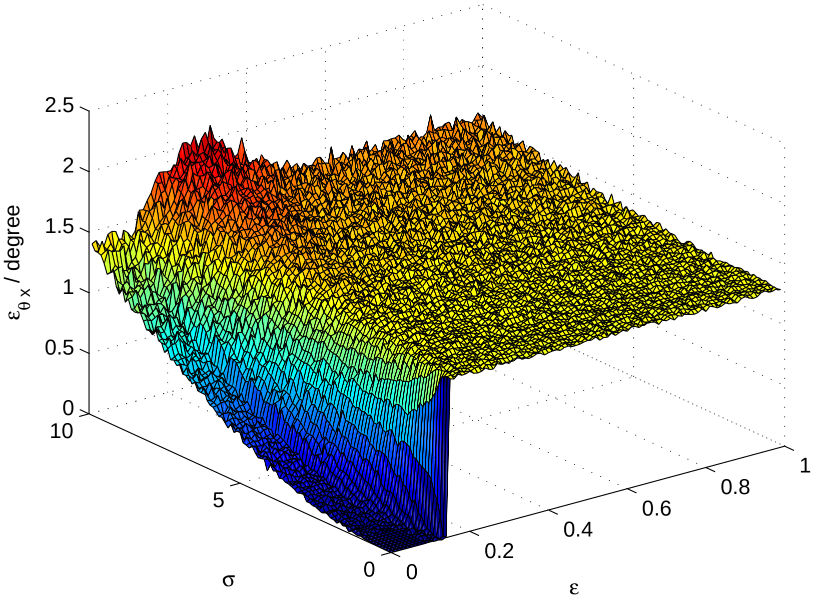


Figure 6·18: Root-mean-squared error of rotation around x axis.

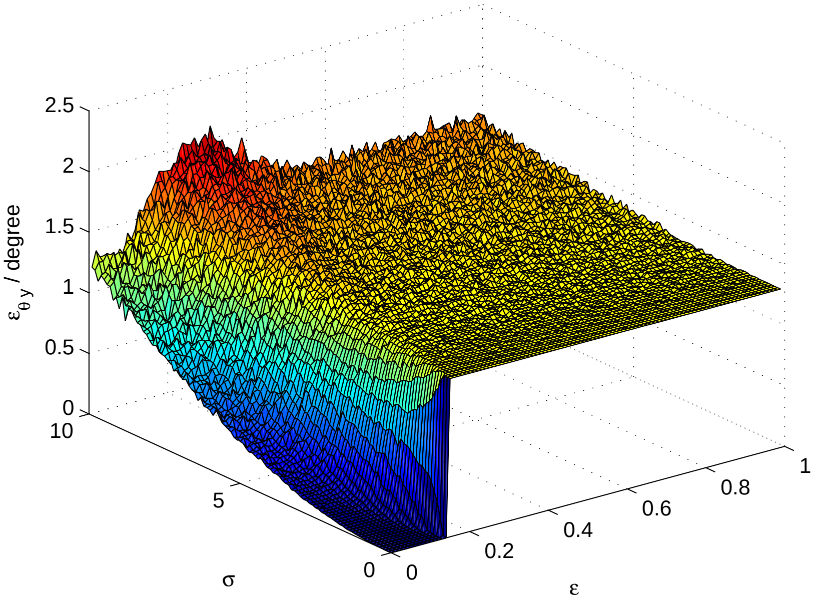


Figure 6·19: Root-mean-squared error of rotation around y axis.

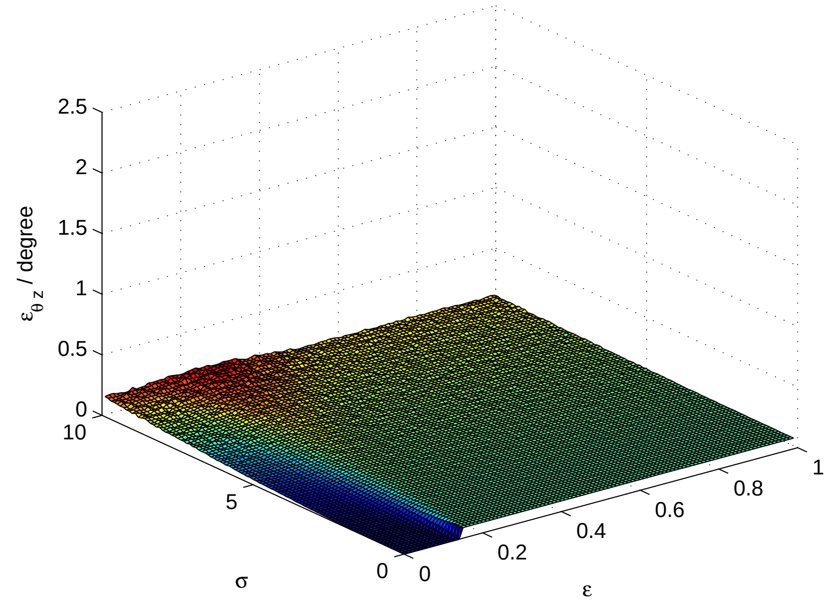


Figure 6·20: Root-mean-squared error of rotation around z axis.

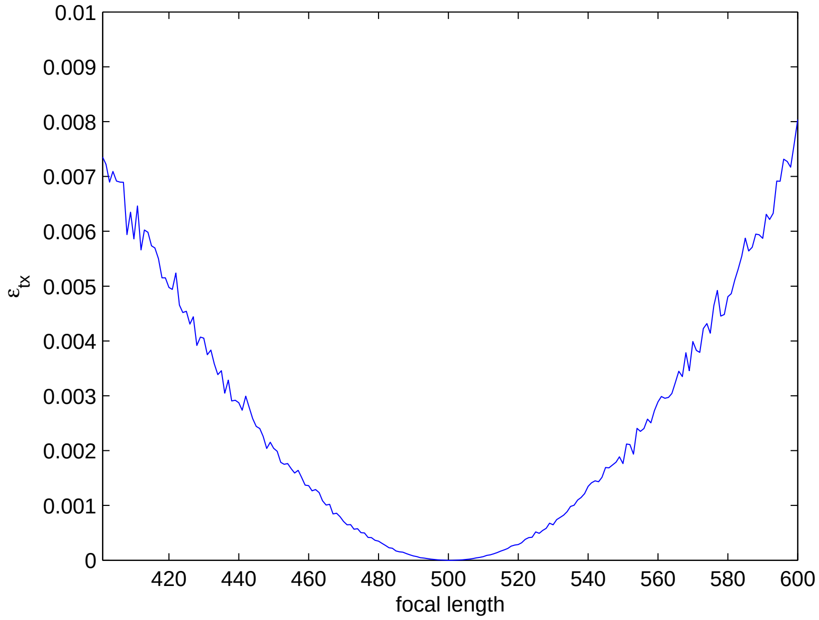


Figure 6·21: Root-mean-squared error of translation in x direction as a function of assumed focal length f (500 is the ground truth).

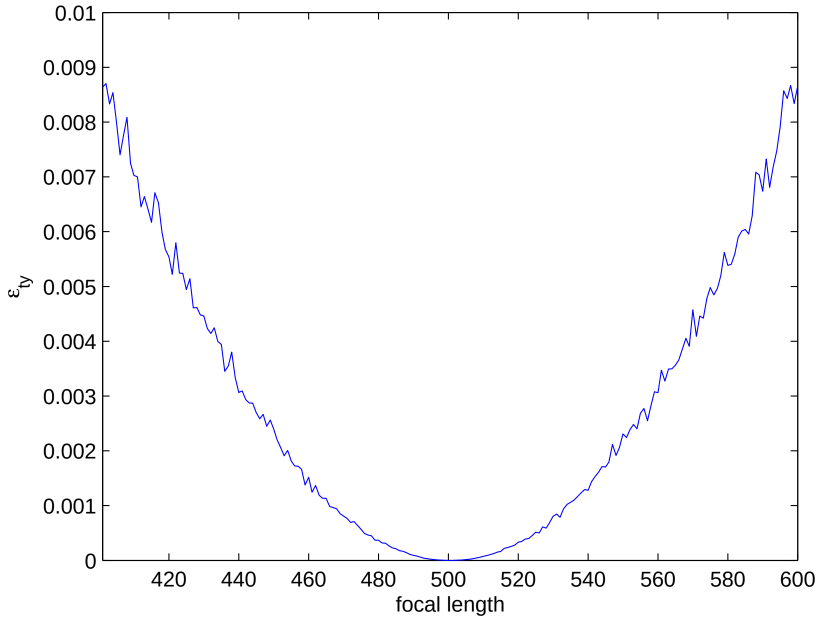


Figure 6·22: Root-mean-squared error of translation in y direction as a function of assumed focal length f (500 is the ground truth).

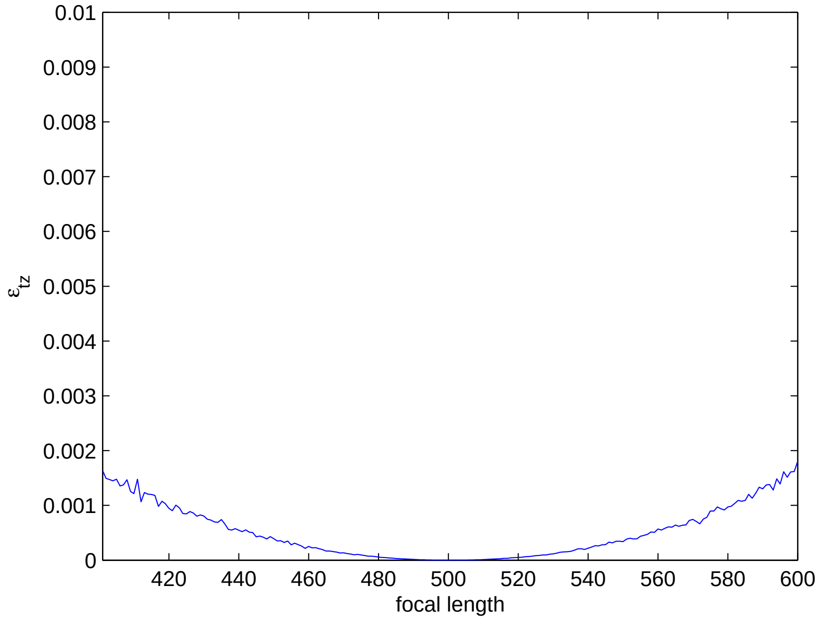


Figure 6·23: Root-mean-squared error of translation in z direction as a function of assumed focal length f (500 is the ground truth).

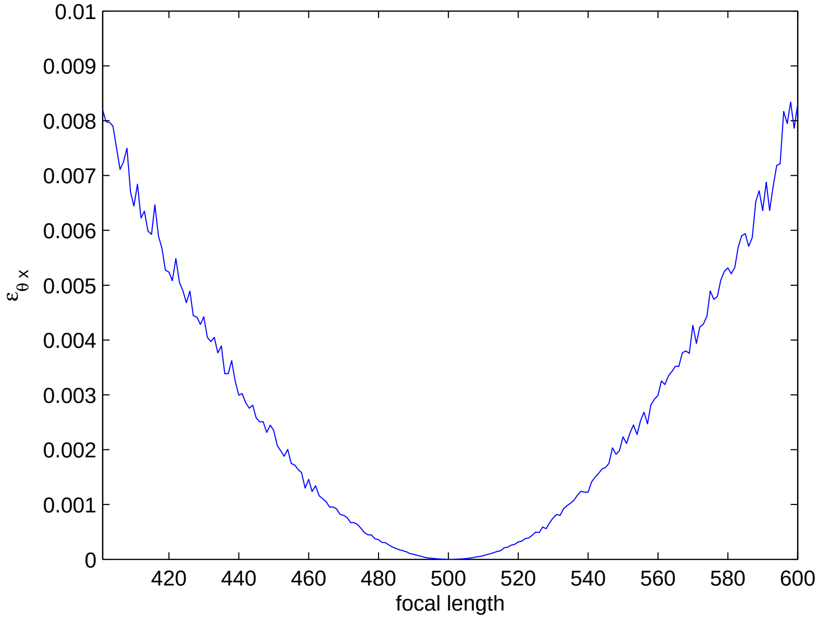


Figure 6·24: Root-mean-squared error of rotation around x axis as a function of assumed focal length f (500 is the ground truth).

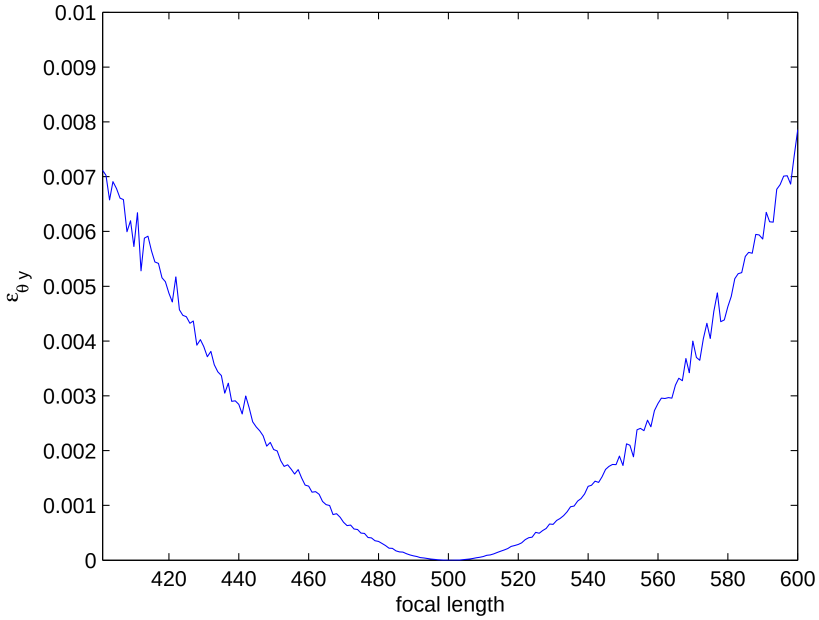


Figure 6·25: Root-mean-squared error of rotation around y axis as a function of assumed focal length f (500 is the ground truth).

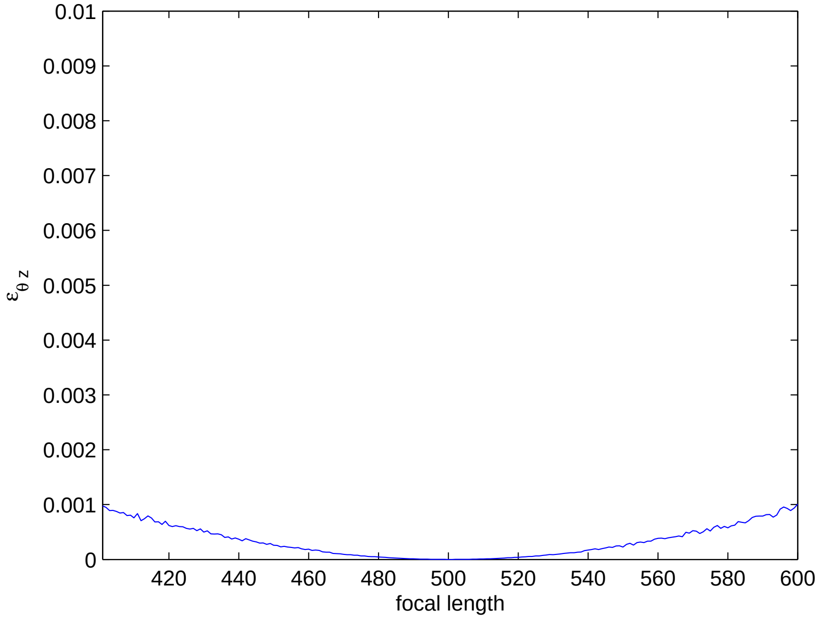


Figure 6·26: Root-mean-squared error of rotation around z axis as a function of assumed focal length f (500 is the ground truth).

6.2 "Ground-truth" experiments

Having tested the accuracy of the algorithm in a simulated environment, we now test it on real data, i.e., using a camera (Logitech C905) pointed at a planar target (chessboard pattern). In this section, we first present initial results obtained in a simple table-top experiment, and then we repeat the experiments using a recently-built platform that allows more precise control of camera movements.

6.2.1 Table-top experiments

While our test platform was being built, we performed initial experiments on a lab table that we adjusted to be as close to level as possible. We taped a chessboard pattern to an adjacent wall and placed a square on the table as a guidance for camera movement. Figure 6·27 illustrates how we set up our table-top experiment in the lab. We set the world coordinate system's z-axis perpendicular to the target plane,

x-axis parallel to the table surface pointing to the right and y-axis perpendicular to the table pointing up.

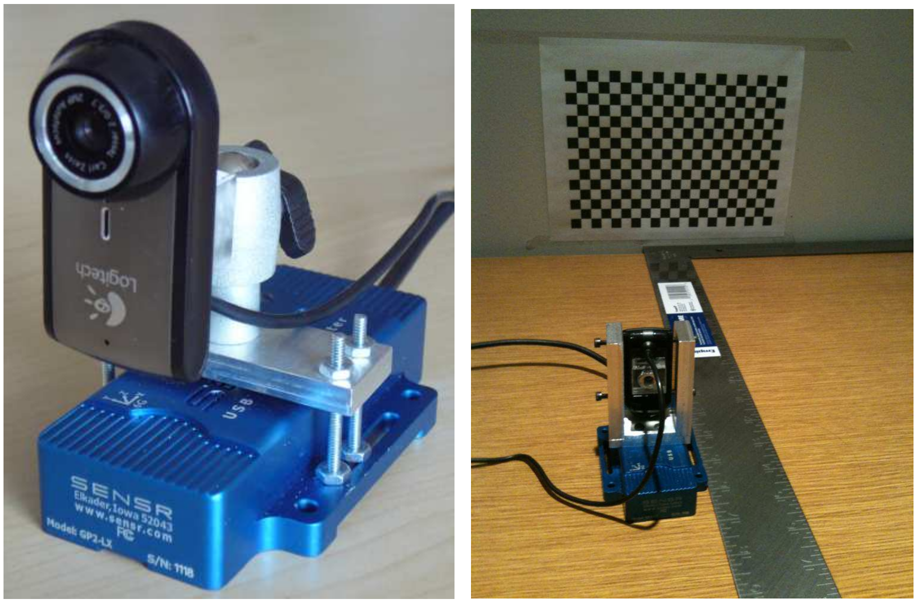


Figure 6·27: Camera setup for real-data experiments

We tested the homography decomposition method using the above setup in 6 cases (the unit for translation vector is 1 inch and the unit for rotation vector is 1 degree):

- **Case 1:** The camera coordinate system is aligned with the world coordinate system, and the camera is moved along the x axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$ (Figure 6·28 shows the setup in this case and Figure 6·34 shows some sample frames from the captured video).

- **Case 2:** The camera coordinate system is aligned with the world coordinate system, and the camera is moved along the z axis of the world coordinate system by $\vec{t}_g^{(k)} = (0, 0, -k)^T$ (Figure 6·29 shows the setup in this case and Figure 6·35 shows some sample frames from the captured video).

- **Case 3:** The camera is tilted down to miss-align the y and z axes of the camera coordinate system with respect to the world coordinate system by angle

β_1, and the camera is moved along the z-axis of the world coordinate system by $\vec{t}_g^{(k)} = (0, 0, -k)^T$. (Figure 6·30 shows the setup in this case and Figure 6·36 shows some sample frames from the captured video).

- **Case 4:** The camera is panned to the right to miss-align the x and z axes of the camera coordinate system with respect to the world coordinate system by angle β_2, and the camera is moved along the x-axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$. (Figure 6·31 shows the setup in this case and Figure 6·37 shows some sample frames from the captured video).

- **Case 5:** The camera is rotated around the z-axis clockwise to miss-align the x and y axes of the camera coordinate system with respect to the world coordinate system by angle β_3, and the camera is moved along the x-axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$. (Figure 6·32 shows the set-up in this case and Figure 6·38 shows some sample frames from the captured video).

- **Case 6:** In this scenario, the camera is first moved away from the target to the end of the square, then is moved to the other side of the square, and finally it is moved towards the target. Figure 6·33 shows how the camera is being moved in this case. The black camera indicates the reference position for our estimation, which is the first frame we took. Figure 6·39 shows some frames of the captured image sequence.

Figures 6·40 to 6·45 show the plots of translation and rotation estimates in all cases. Since we always move the camera inch by inch, we expect the translation estimates to either increase or decrease linearly. From the results of the first two cases (Figures 6·40 and 6·41), we can see that the estimates of t_x and t_z change almost linearly, however, t_y, t_z in the first case and t_x, t_y in the second case remain zero as expected. In addition, estimates of the rotation angles change within a small

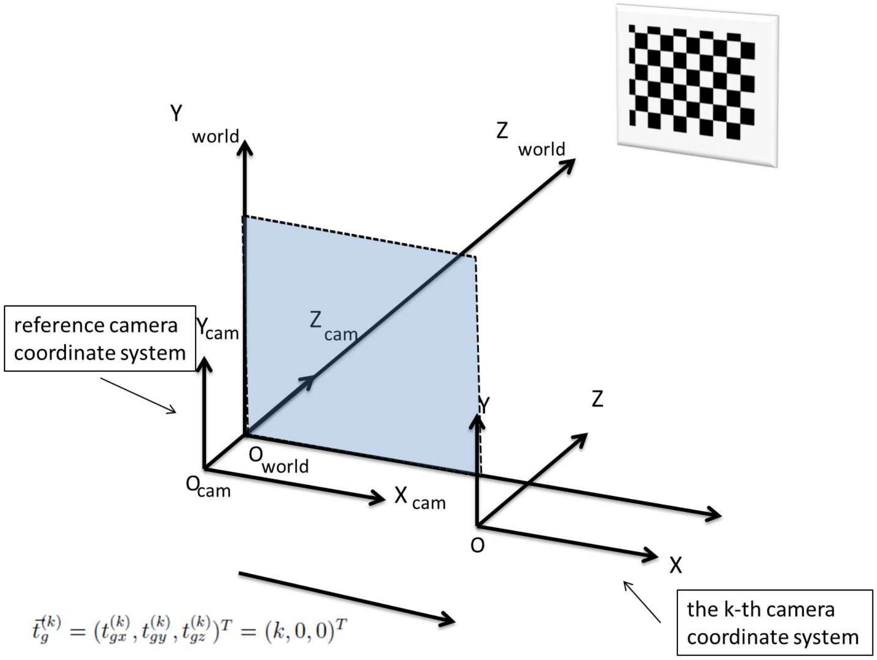

Y world
Z world
reference camera coordinate system
Y cam
Z cam
O world
O cam
X cam
Y
Z
O
X
$\vec{t}_g^{(k)} = (t_{gx}^{(k)}, t_{gy}^{(k)}, t_{gz}^{(k)})^T = (k, 0, 0)^T$
the k-th camera coordinate system

Figure 6·28: Translation of the camera along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$ in case 1.

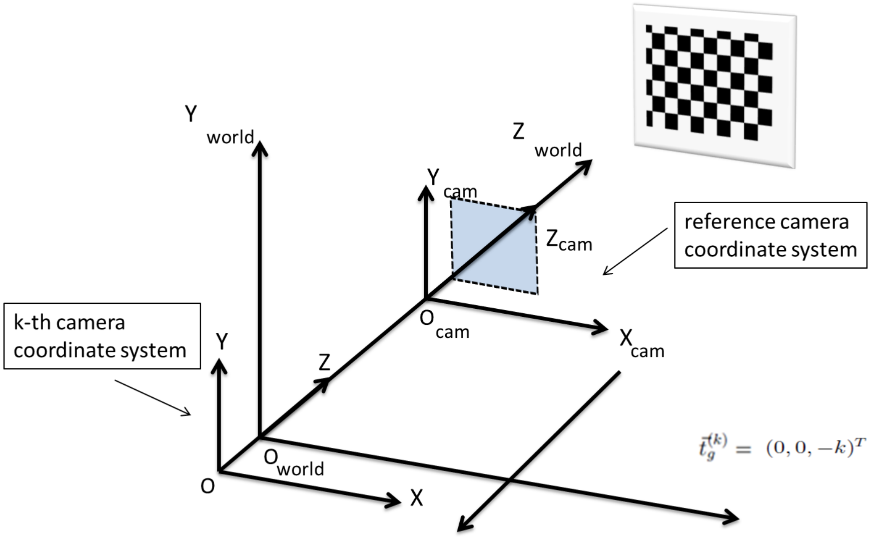

Y world
Z world
Y cam
Z cam
reference camera coordinate system
O cam
X cam
k-th camera coordinate system
Y
Z
$\vec{t}_g^{(k)} = (0, 0, -k)^T$
O world
O
X

Figure 6·29: Translation of the camera along z axis of the world coordinate system by $\vec{t}_g^{(k)} = (0, 0, -k)^T$ in case 2.

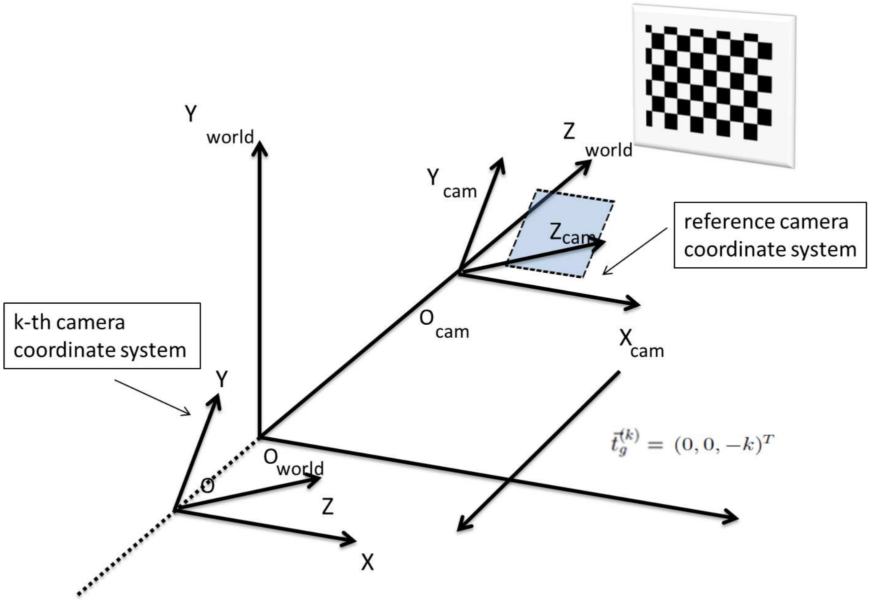

Y world
Z world
Y cam
Z cam
reference camera coordinate system
O cam
X cam
k-th camera coordinate system
Y
O
O world
Z
X
$\vec{t}_g^{(k)} = (0, 0, -k)^T$

Figure 6·30: Rotation around x-axis by angle β_1 followed by translation of the camera along z axis of the world coordinate system by $\vec{t}_g^{(k)} = (0, 0, -k)^T$ in case 3.

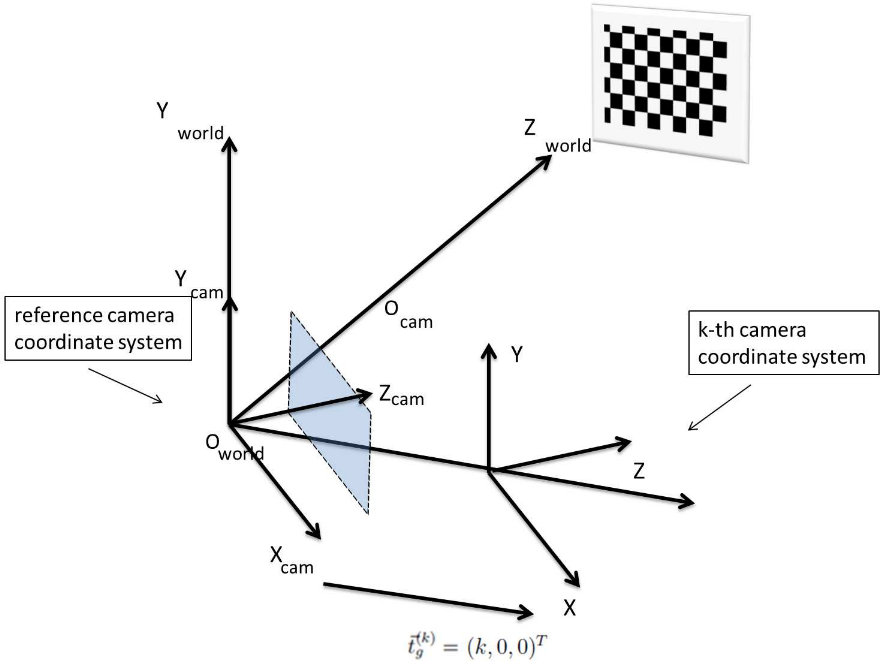

Y world
Z world
Y cam
reference camera coordinate system
O cam
k-th camera coordinate system
Y
Z cam
O world
Z
X cam
X
$\vec{t}_g^{(k)} = (k, 0, 0)^T$

Figure 6·31: Rotation around y-axis by angle β_2 followed by translation of the camera along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$ case 4.

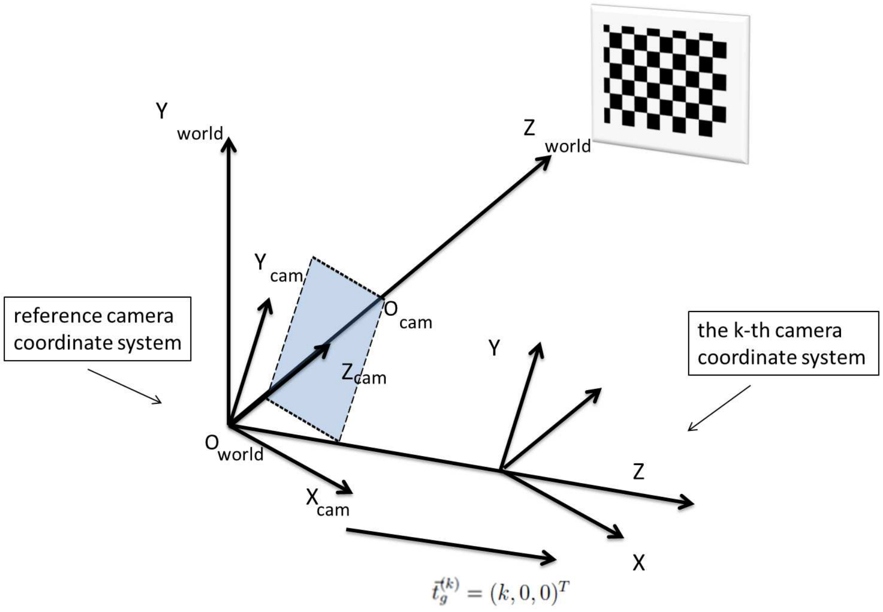

Y world
Z world
Y cam
O cam
reference camera coordinate system
the k-th camera coordinate system
Y
Z cam
O world
Z
X cam
X
$\vec{t}_g^{(k)} = (k, 0, 0)^T$

Figure 6·32: Rotation around z-axis by angle β_3 followed by translation of the camera along x axis of the world coordinate system by $\vec{t}_g^{(k)} = (k, 0, 0)^T$ in case 5.

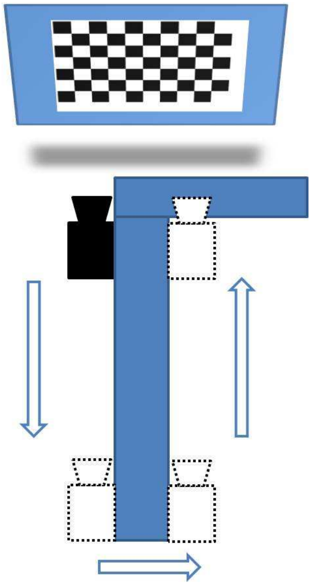


Figure 6·33: Camera movements in case 6.

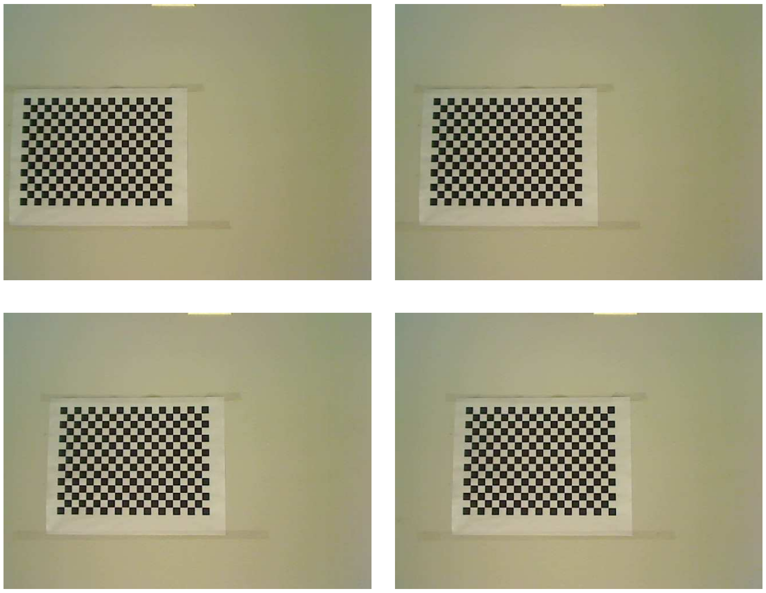


Figure 6·34: Sample frames from a video captured in case 1.

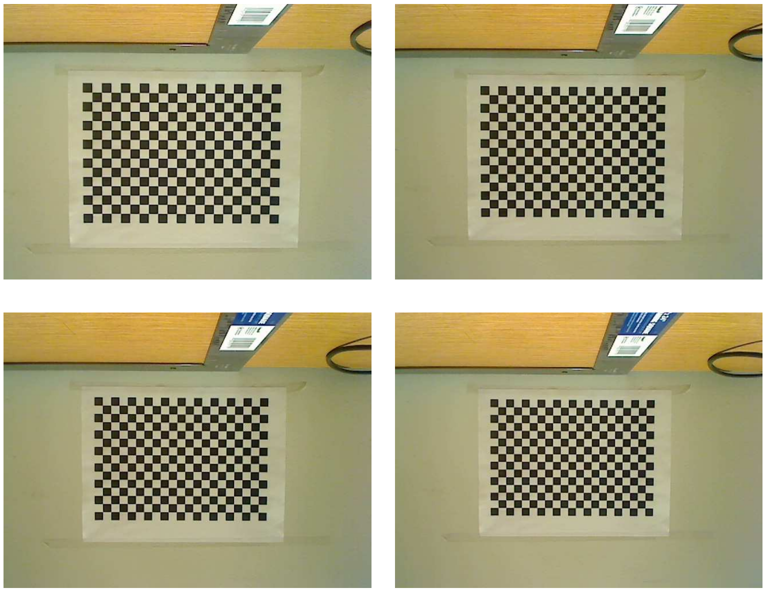


Figure 6·35: Sample frames from a video captured in case 2.

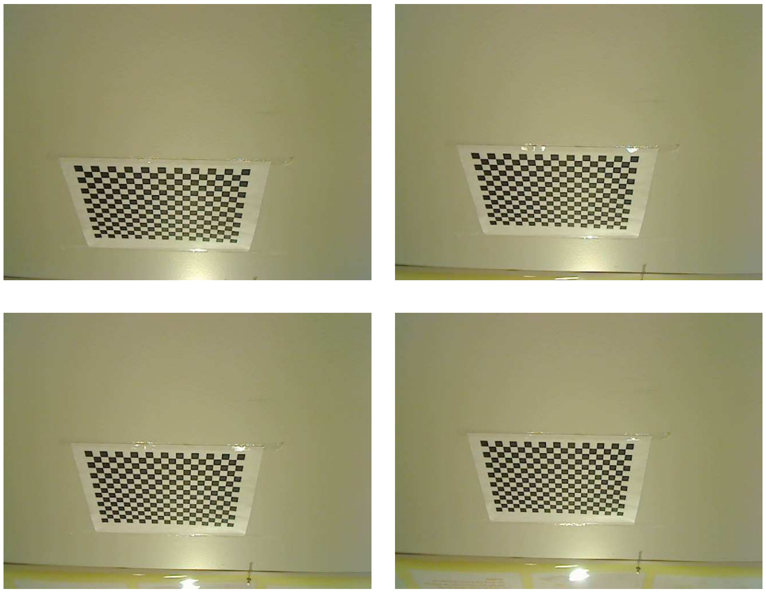


Figure 6·36: Sample frames from a video captured in case 3.

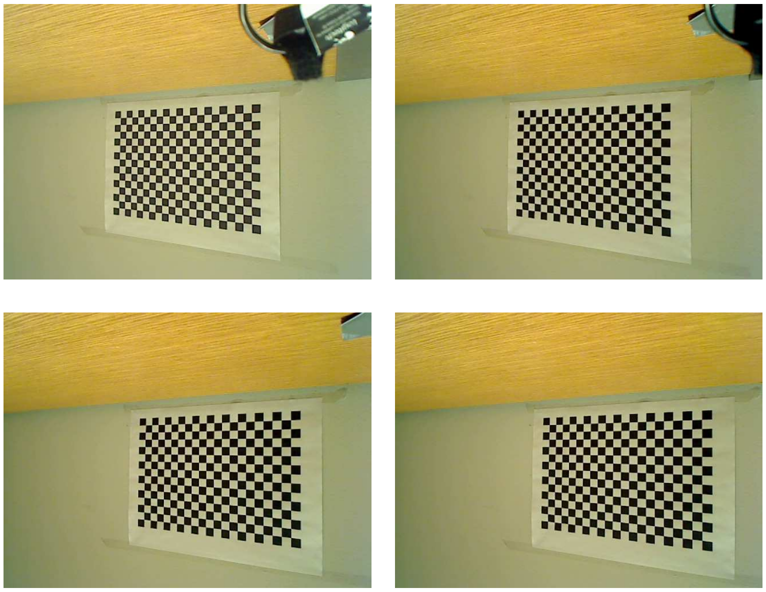


Figure 6·37: Sample frames from a video captured in case 4.

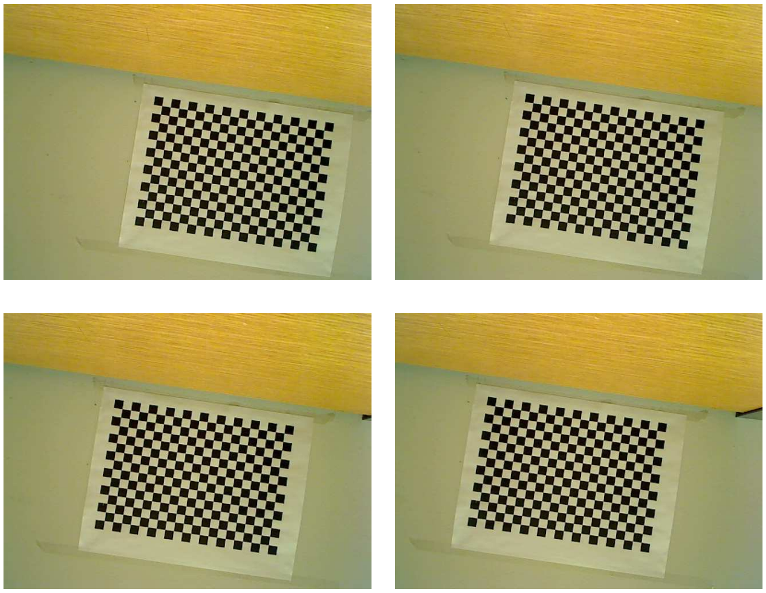


Figure 6·38: Sample frames from a video captured in case 5.

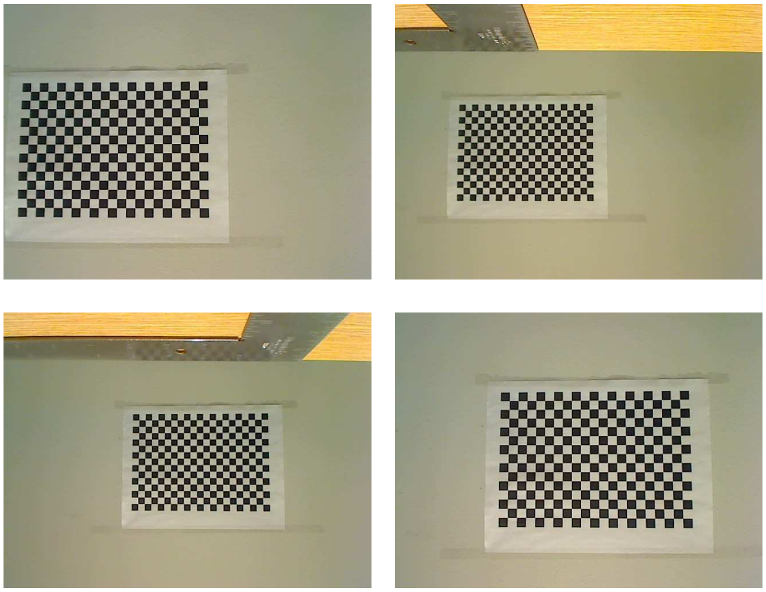
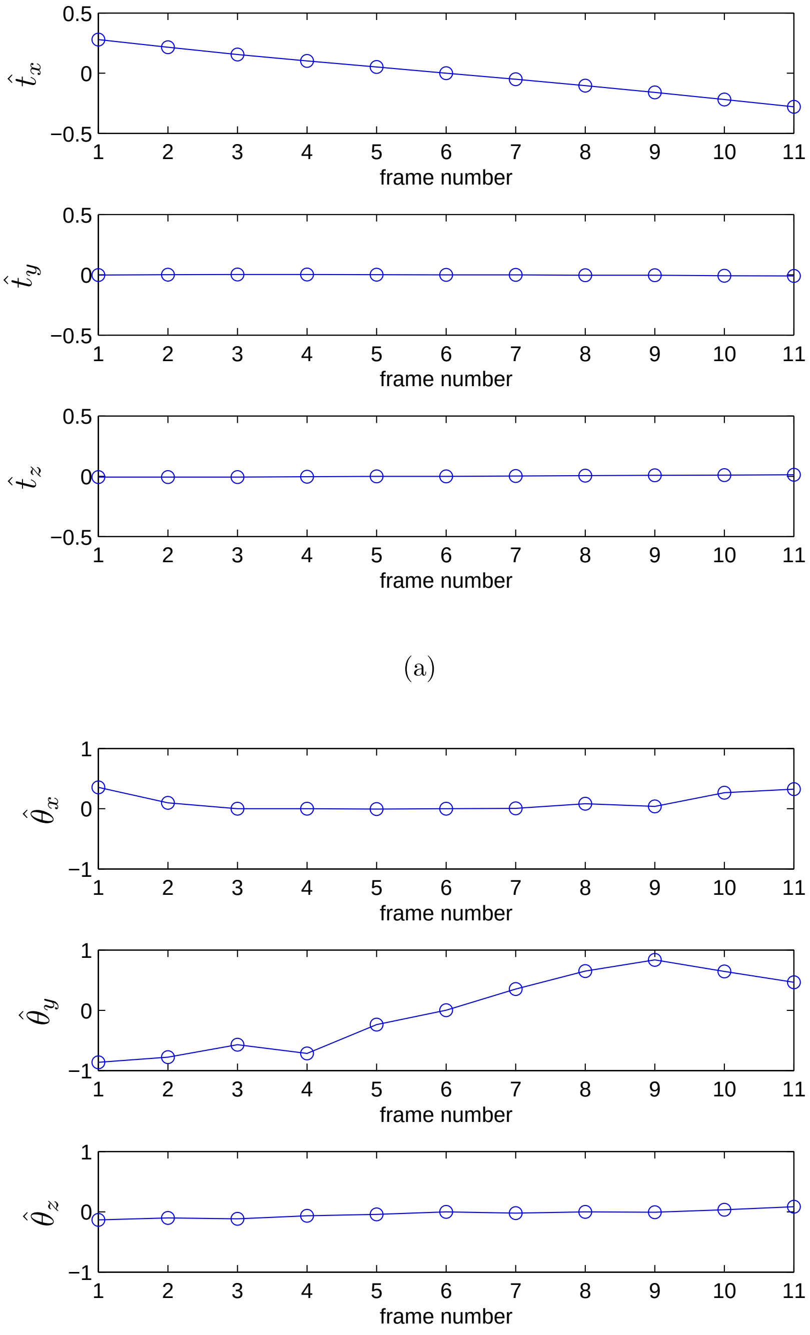


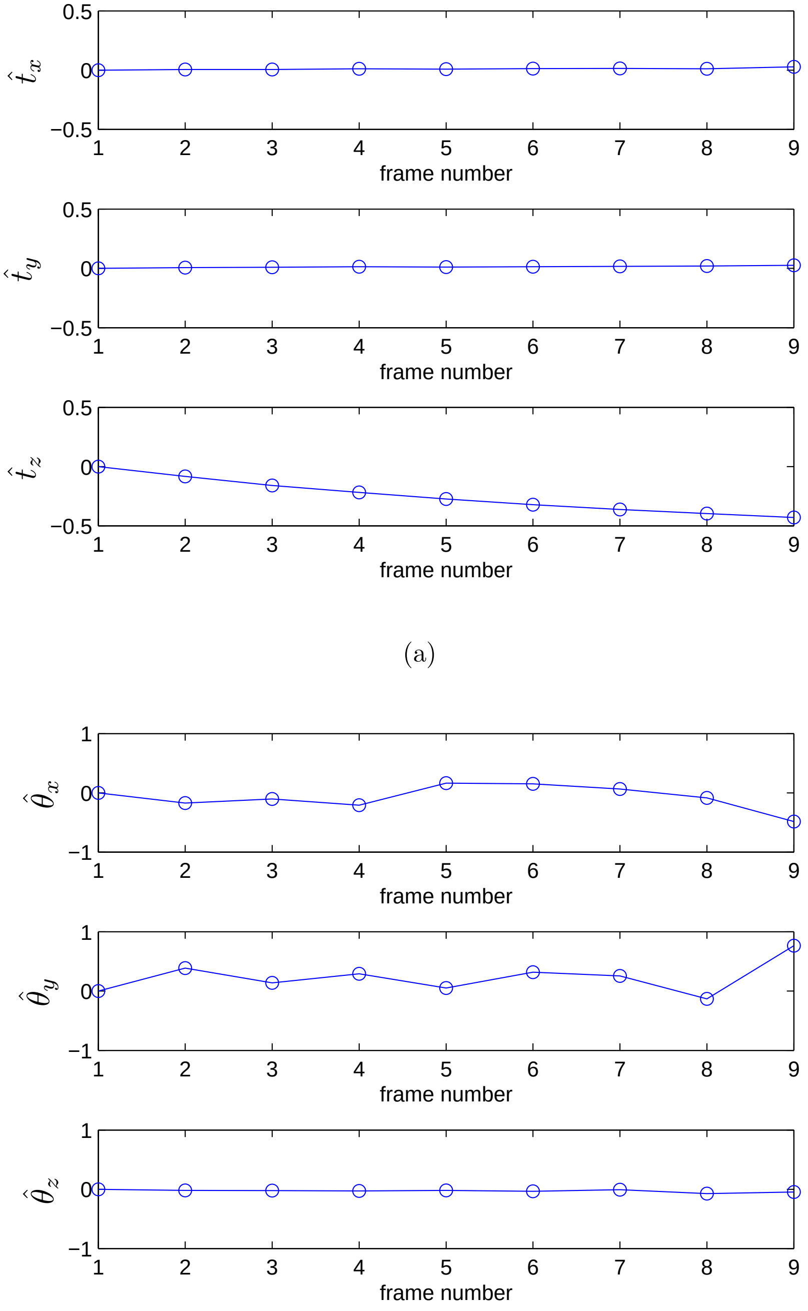
Figure 6·39: Sample frames from a video captured in case 6.



(a)

(b)

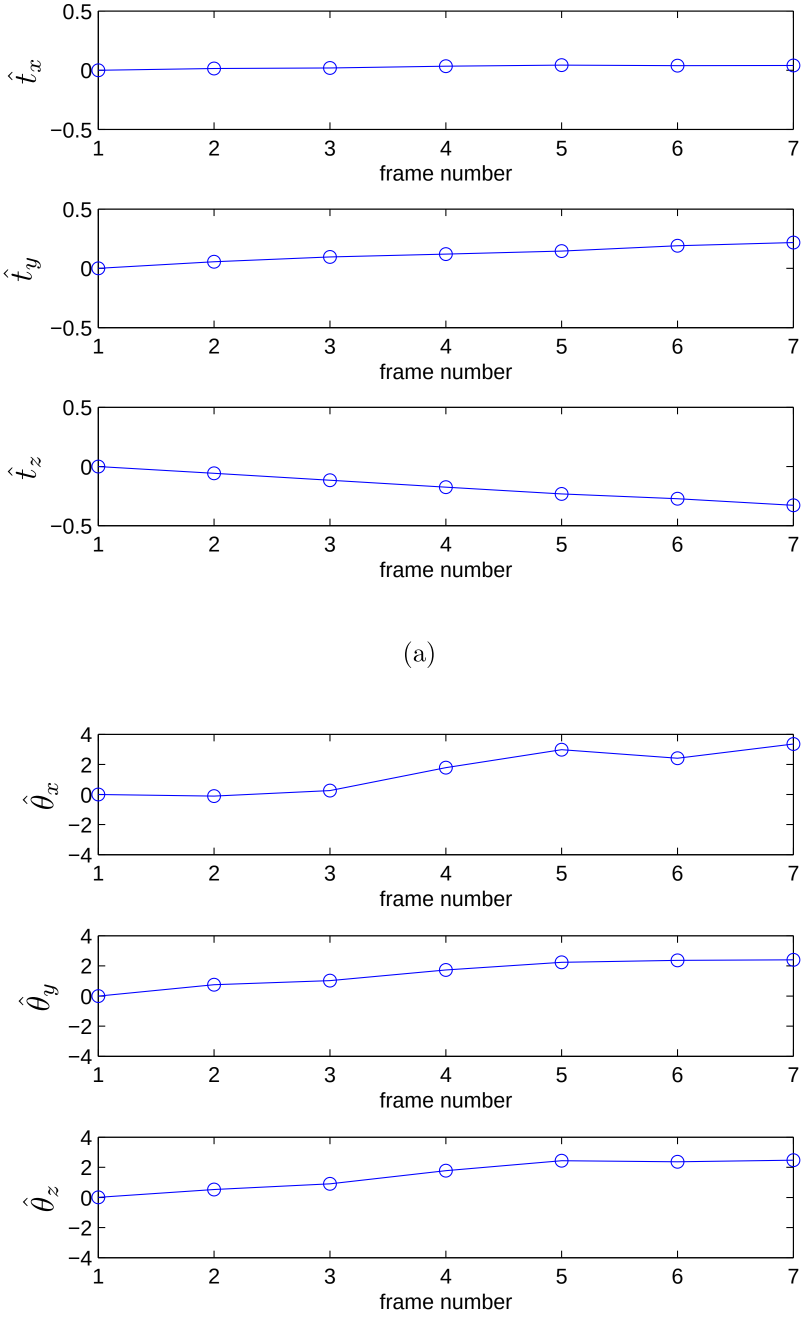
Figure 6·40: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 1.



(a)

(b)

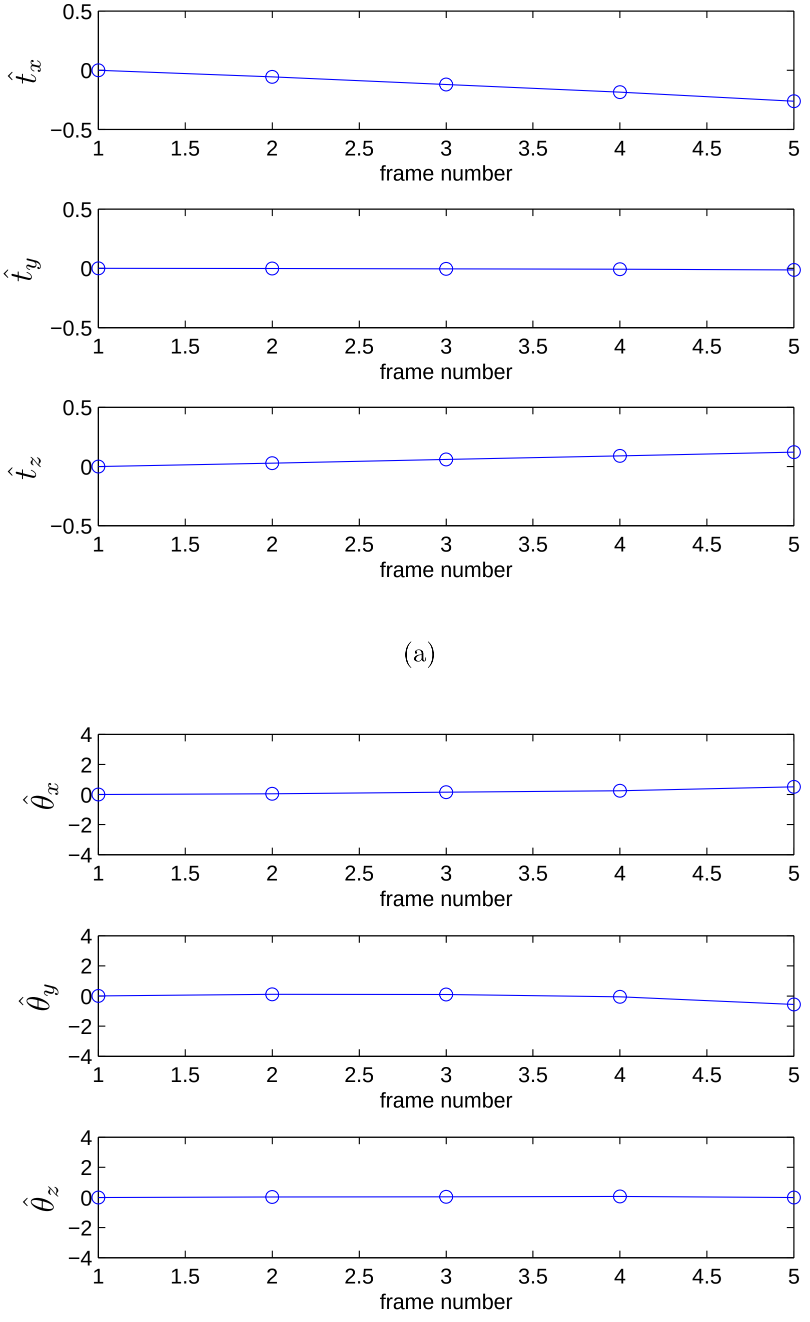
Figure 6·41: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 2.



(a)

(b)

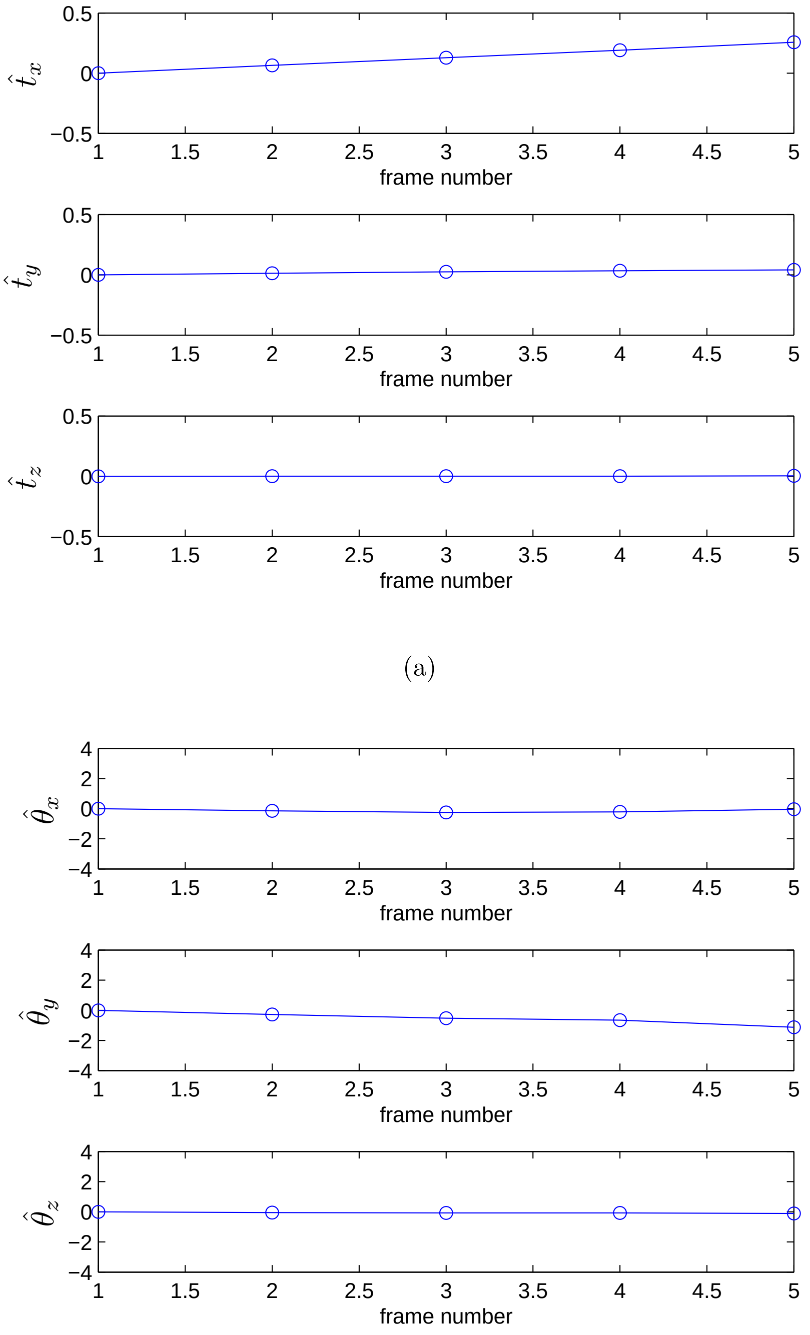
Figure 6·42: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 3.



(a)

(b)

Figure 6·43: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 4.



(a)

(b)

Figure 6·44: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 5.

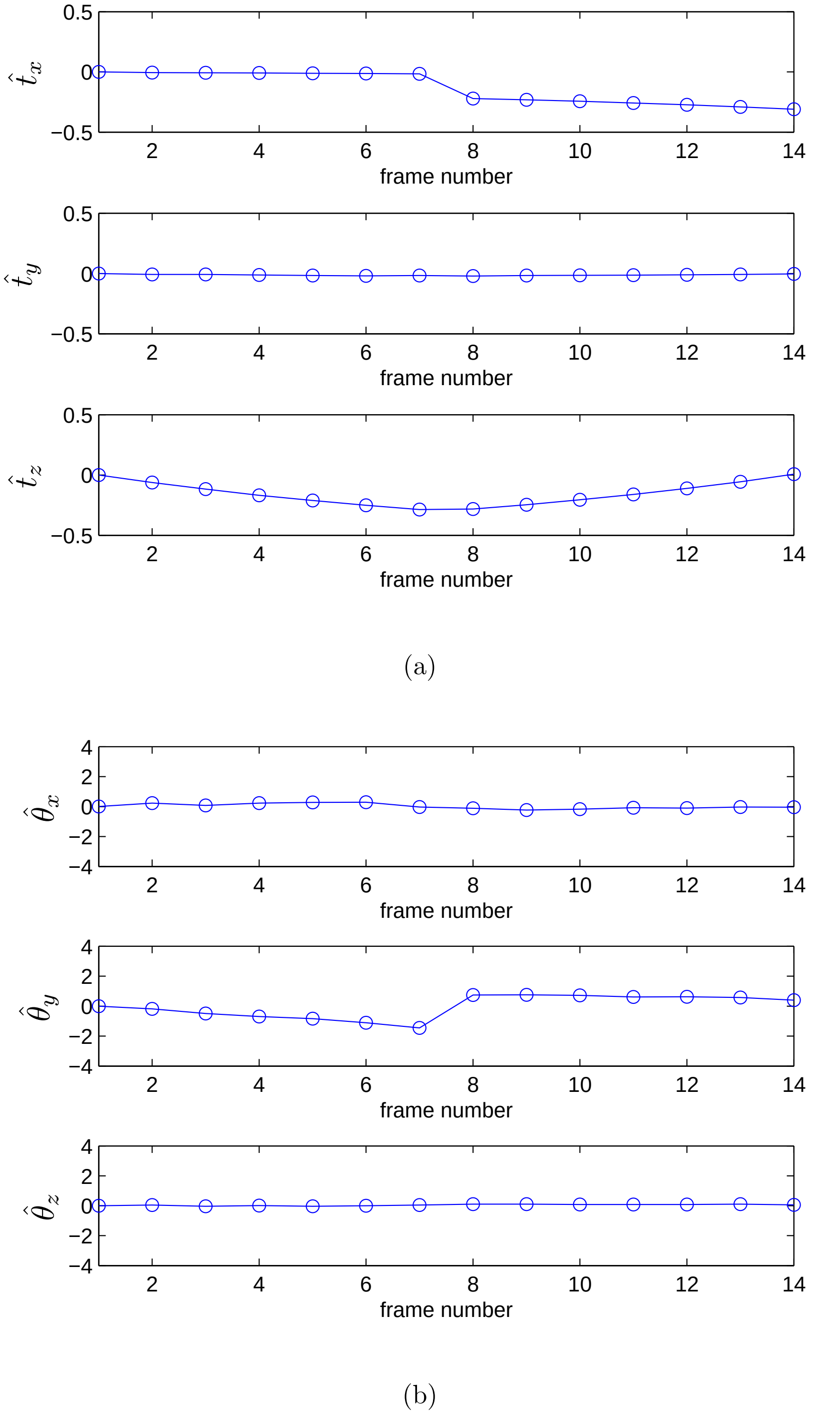


Figure 6·45: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera movement in case 6.

range due to errors introduced by feature extraction and estimation.

By developing cases 3 and 4, we want to emphasize that the estimated 3D translation and rotation are relative to the reference camera coordinate system rather than the world coordinate system. For example, results in Figure 6·42 show that both t_y and t_z are linearly changing because we miss-aligned the y and z axes of the camera coordinate system with respect to the world coordinate system. As we move the camera along the z axis of the world coordinate system, the position of the camera actually moves to the point where the y coordinate increases to a positive value and z coordinate decreases to a negative value in the coordinate system of the reference camera. It is difficult to see any changes in t_y in Figure 6·44 due to the fact that the rotation angle is not large enough to clearly see the impact. By testing under scenario 6, we expected that t_x would remain constant until we placed the camera on the other side of the set square, t_y would remain zero all the time and t_z would decrease linearly when the camera moved away from the pattern and increase linearly when it moved towards the pattern. Results in Figure 6·45 confirm that, so we can conclude that this algorithm works well under combinations of motions.

Since the estimated translations are only proportional to the ground truth vector $\vec{t}_g$, we cannot calculate the root-mean-squared error. Instead, we calculated the mean μ_t and standard deviation σ_t of the translation increments estimated between consecutive video frames for each case. If the estimates are consistent across time, we should have very small standard deviations compared to the means. As shown in Table 6.2, the standard deviations are at most 1/8-th of the corresponding means, which indicates a good consistency of the algorithm.

We did not test rotations in the initial table-top experiments for the lack of suitable rotation angle measurement device.

Table 6.2: Means μ_t and standard deviations σ_t of the corresponding translation for 5 cases.

	Case 1	Case 2	Case 3	Case 4	Case 5
μ	-0.055	-0.0538	-0.0548	-0.0655	0.0642
σ	0.0043	0.0018	0.0071	0.0082	0.0018

6.2.2 Platform-based experiments

Our table-top experiments did not allow precise camera control or table leveling, and were performed while we awaited completion of a precise mechanical platform being built to our specifications. The platform, shown in Figure 6·46, allows precise movement in x-z plane and rotation around all three axes.

First, we repeated the initial translation experiments where the camera was moved along x and z axes inch by inch. Then, we performed rotation experiments since the platform is equipped with two goniometers and one rotation stage allowing rotation around the x, y and z axes. Between each two consecutive images captured, we incremented angles by 4 degrees around the x and z axes, and by 2 degrees around the y axis. Figures 6·47 to 6·51 show sample video frames for each of the experiments, while Figure 6·52 to 6·56 show the resulting translation and rotation estimate plots. As expected, the rotation remained zero while the camera moved along the x and z axes, and translation remained zero while the camera rotated around the x, y and z axes. Furthermore, the translation and rotation estimates change linearly in each case tested.

Again, in order to check consistency of the estimates we calculated the mean μ_t and standard deviation σ_t of the translation increments estimated between consecutive video frames when camera moved along the x and z axes. Table 6.3 shows that standard deviations are less than 1/20-th of the corresponding means indicating

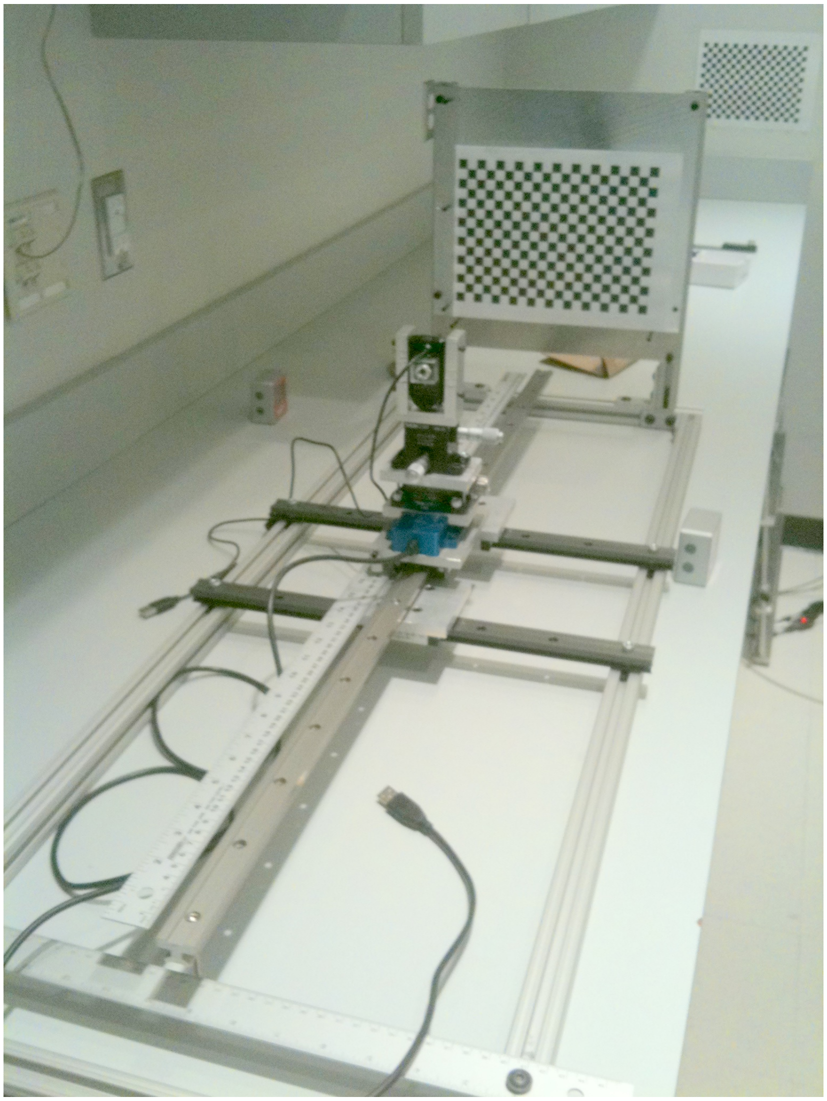


Figure 6·46: Mechanical platform developed to allow precise camera translations and rotations.

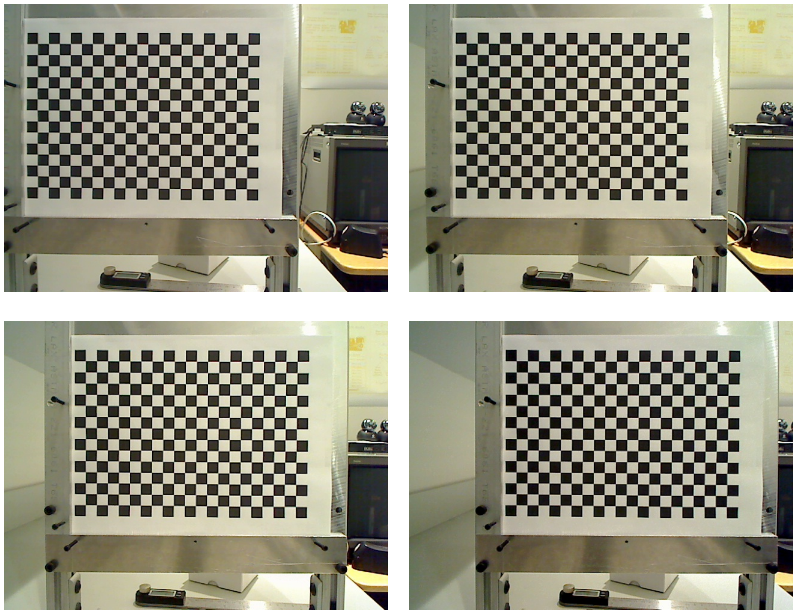


Figure 6·47: Sample frames from a video captured while moving the camera along x-axis.

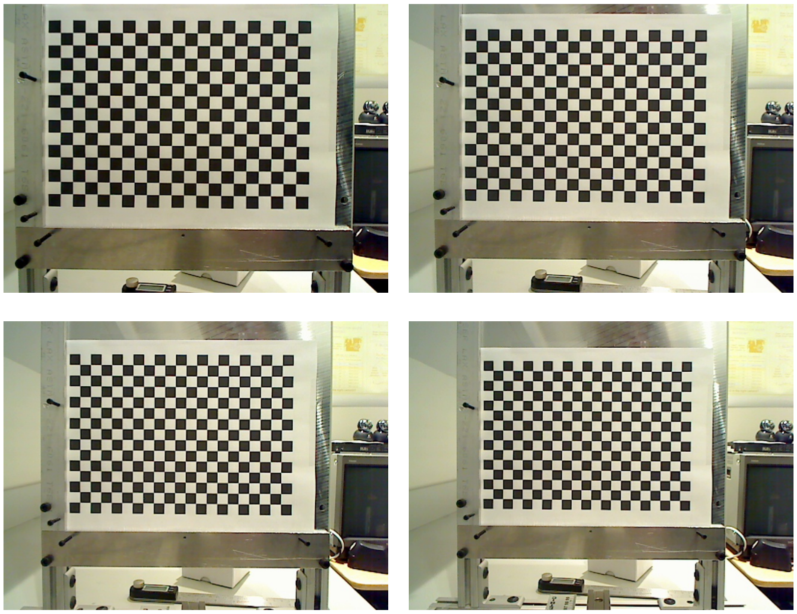


Figure 6·48: Sample frames from a video captured while moving the camera along z-axis.

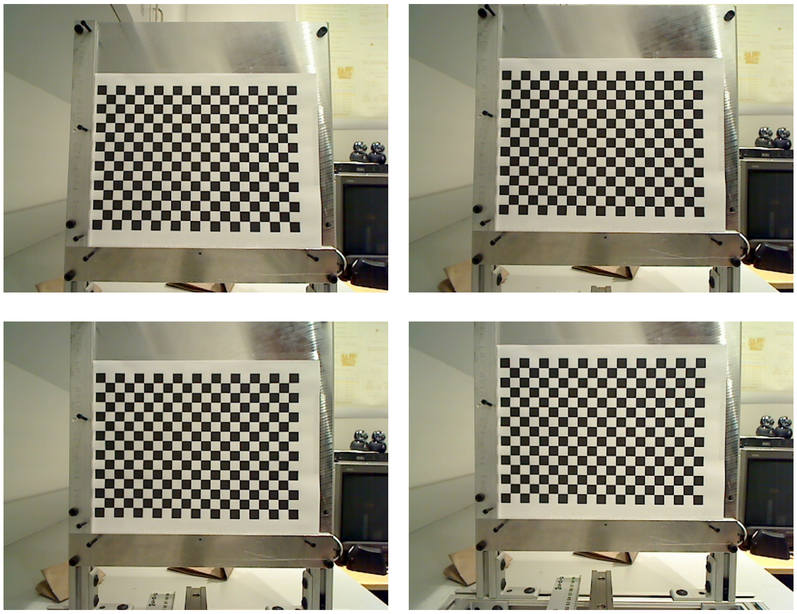


Figure 6·49: Sample frames from a video captured while rotating the camera around x-axis.

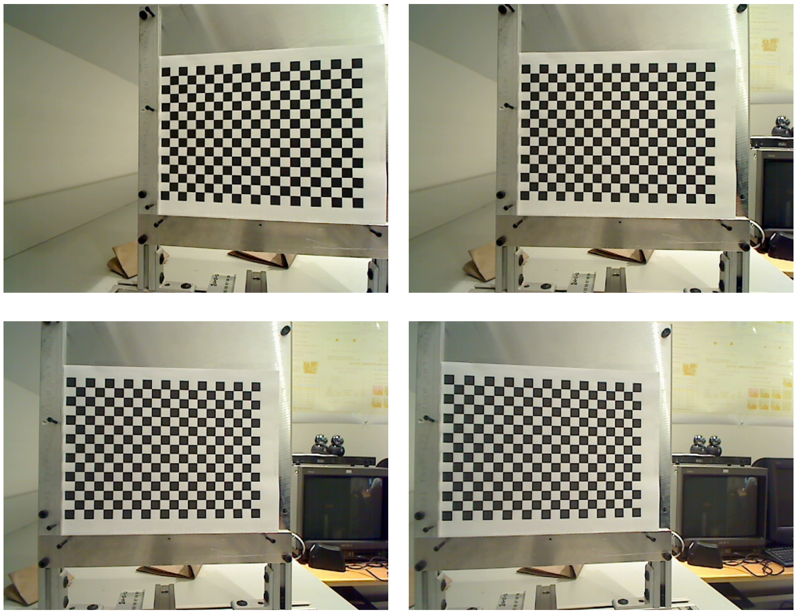


Figure 6·50: Sample frames from a video captured while rotating the camera around y-axis.

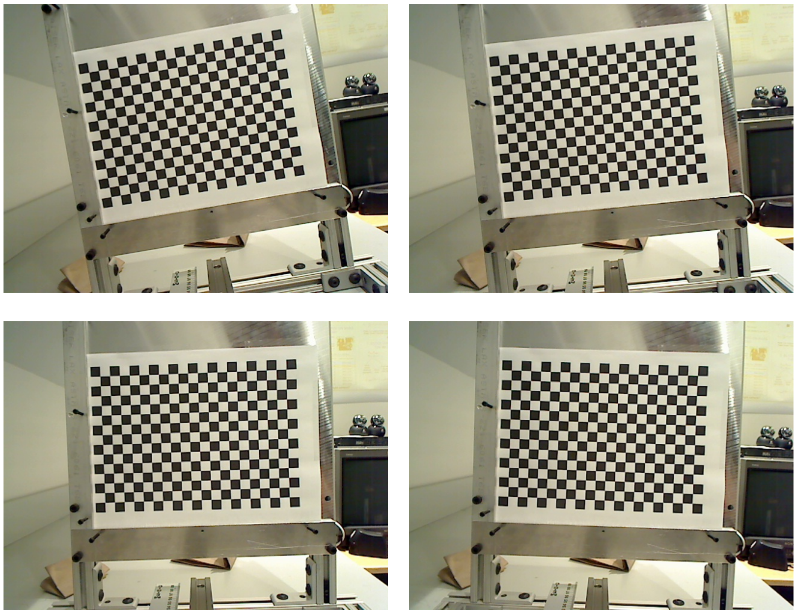
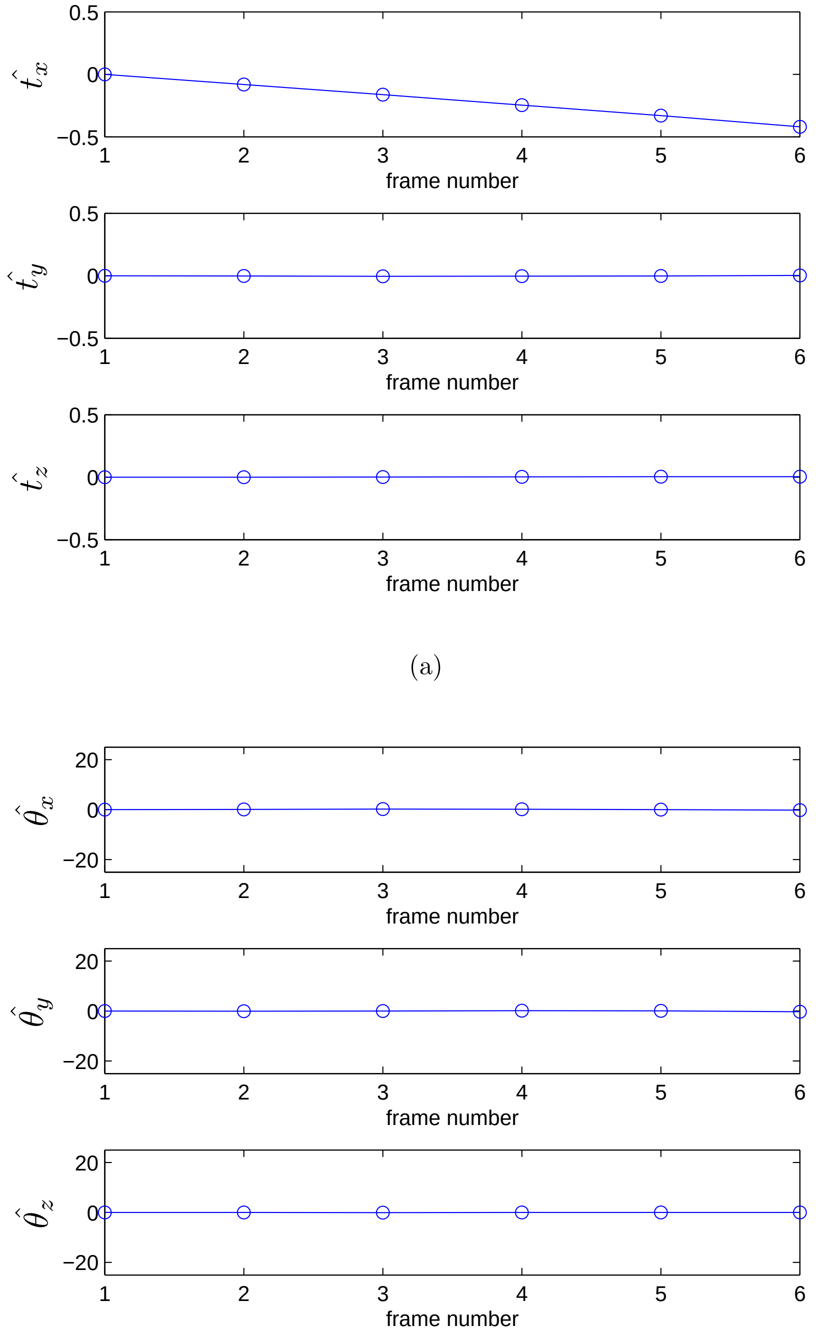


Figure 6·51: Sample frames from a video captured while rotating the camera around z-axis.

even better estimate consistency than in the table-top experiments. This was to be expected since the platform was desgined to allow more precise calibration and movements. Since the unit of the estimated rotations is the same as for the ground-truth, namely 1 degree, we are able to calculate the root-mean-squared errors $\bar{\varepsilon}_{\theta}$. As can be seen in Table 6.4 the root-mean-squared errors are small enough to conclude that the estimates of rotation are quite accurate.

Table 6.3: Mean μ_t and standard deviation σ_t of the corresponding translation along x and z axes.

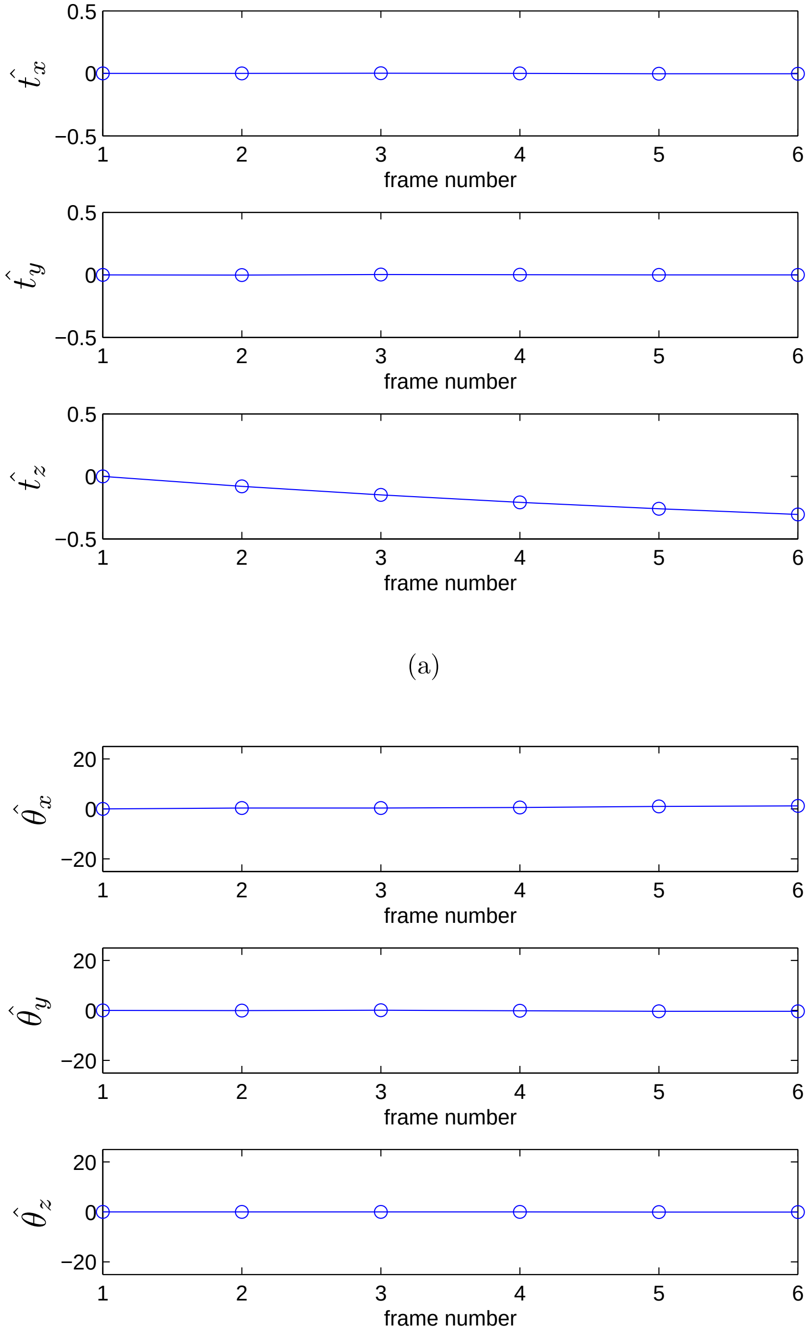
	x-axis translation	z-axis translation
μ_t	-0.0839	-0.0610
σ_t	0.0035	0.0032



(a)

(b)

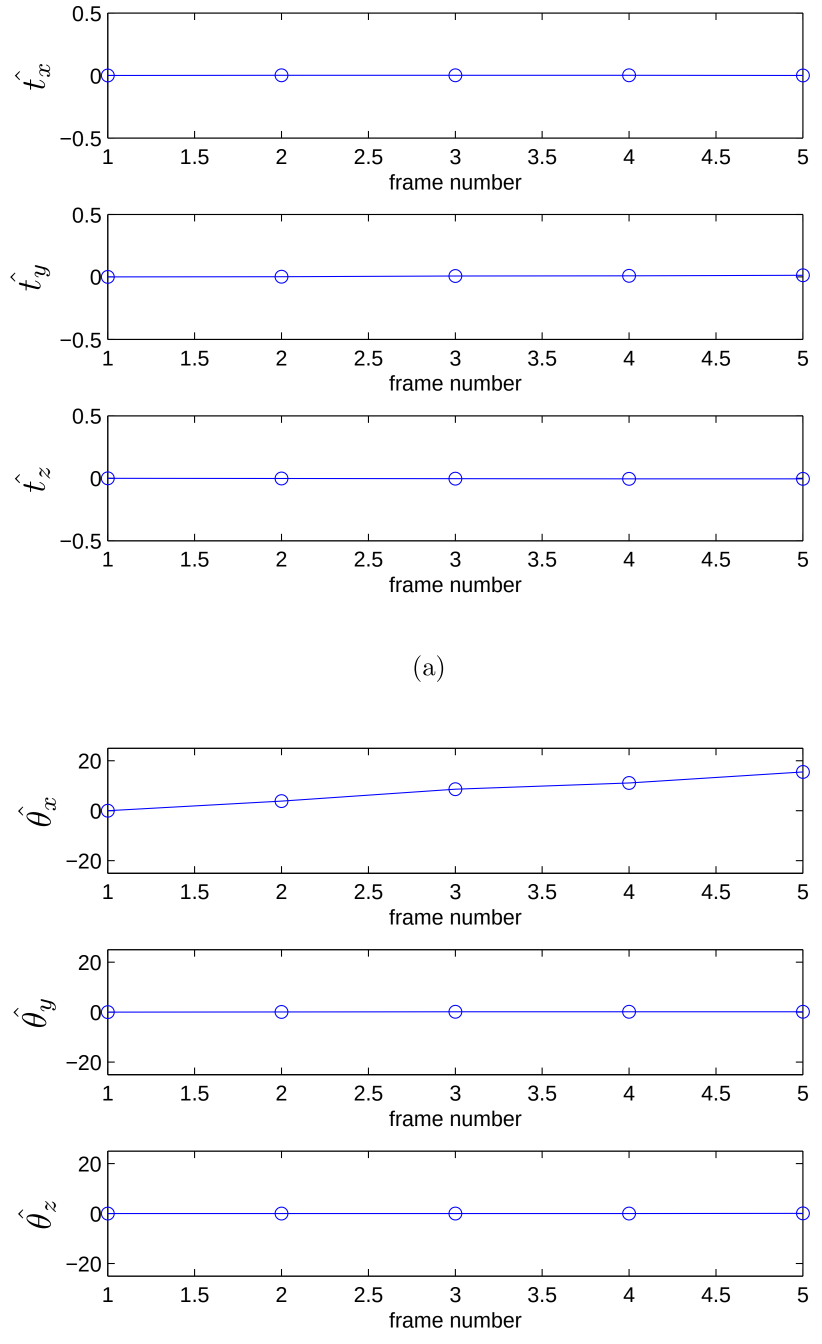
Figure 6·52: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera moved along x-axis



(a)

(b)

Figure 6·53: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera moved along z-axis



(a)

(b)

Figure 6·54: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera rotated around its x-axis

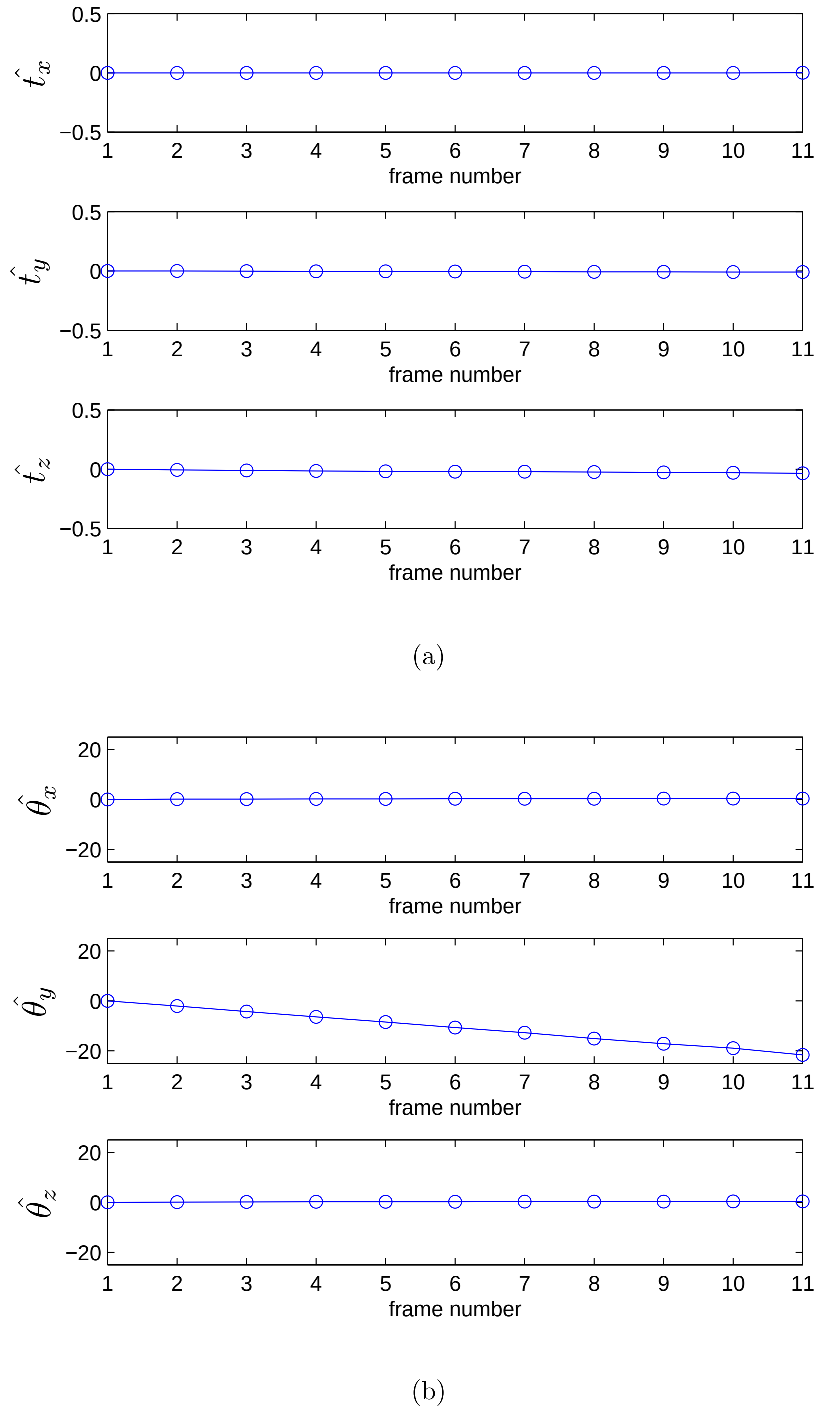
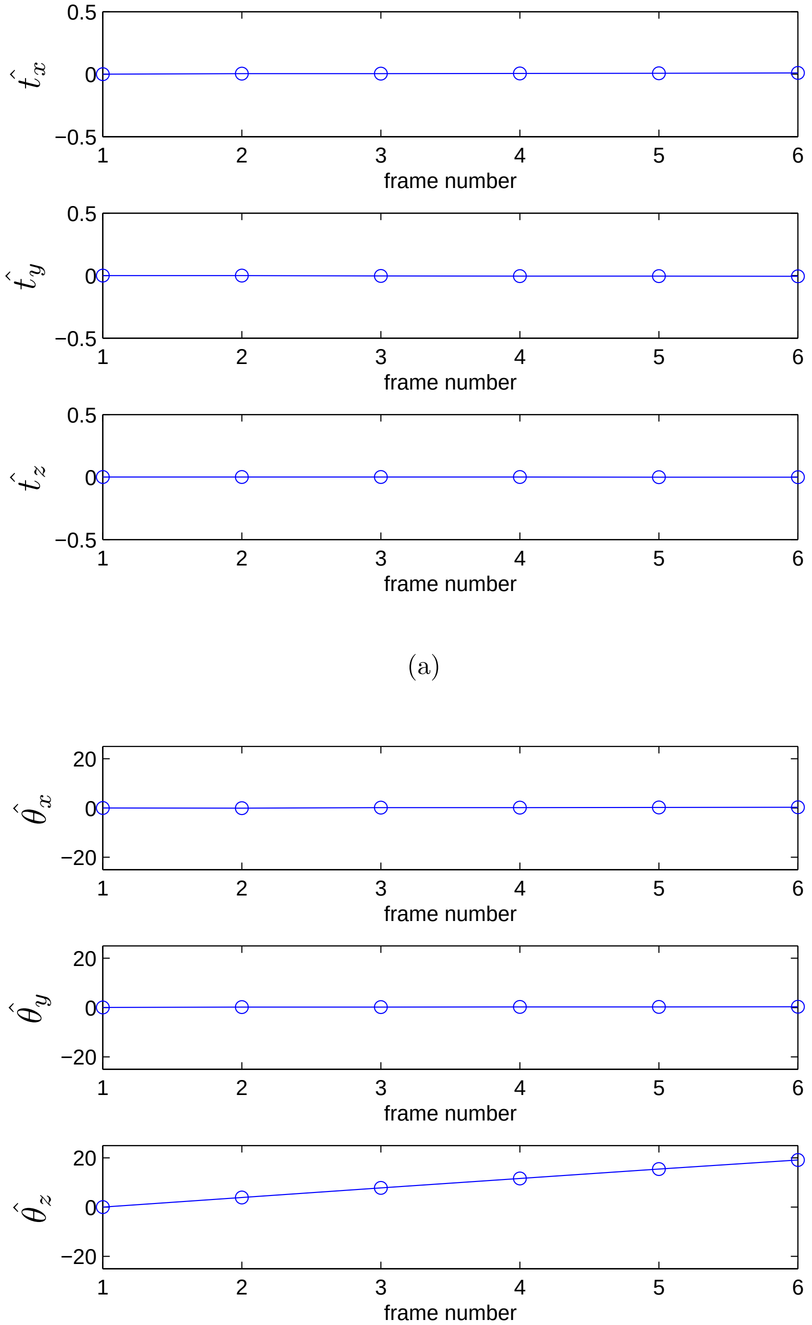


Figure 6·55: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera rotated around its y-axis



(a)

(b)

Figure 6·56: Plots of estimates (a) $(\hat{t}_x, \hat{t}_y, \hat{t}_z)^T$ and (b) $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)^T$ under camera rotated around its z-axis

Table 6.4: Root-mean-squared errors $\bar{\varepsilon}_theta$ between ground truth and estimated rotations around x, y and z axes.

	x-axis rotation	y-axis rotation	z-axis rotation
$\bar{\varepsilon}$	0.4180	0.6809	0.3530

6.3 Accelerometer-based experiments

At last, we compare the estimated 3D translations and rotations with accelerometer measurements, such as the one in Figure 6·57 (a). We attached the same camera we used earlier (Logitech C905) to an accelerometer (Sensr GP2-LX) as shown in (Figure 6·27 and connected both to a computer *via* USB interface. We aligned their coordinate systems to the best degree we could (Figure 6·27). Then, we applied periodic movement to the accelerometer and camera by hand in both x and z directions of the world coordinate system (plane orthogonal to the direction of gravity). Since our accelerometer can only measure the acceleration in x, y and z directions of its coordinate system and cannot measure the rotation angle around it's center, we can only compare the translations obtained by camera and accelerometer, but not the rotations. We need to point out that acceleration measurements in the y direction are based on gravity, and thus unless the accelerometer is perfectly level, gravity will add a component to the acceleration in x and z directions. This introduces an acceleration bias in x and z directions, thus causing velocity and translation errors (drift) after integration (Figure 6·57 (b) and (c)). Since we cannot perfectly level the accelerometer to eliminate the bias, we remove it by a method describe below.

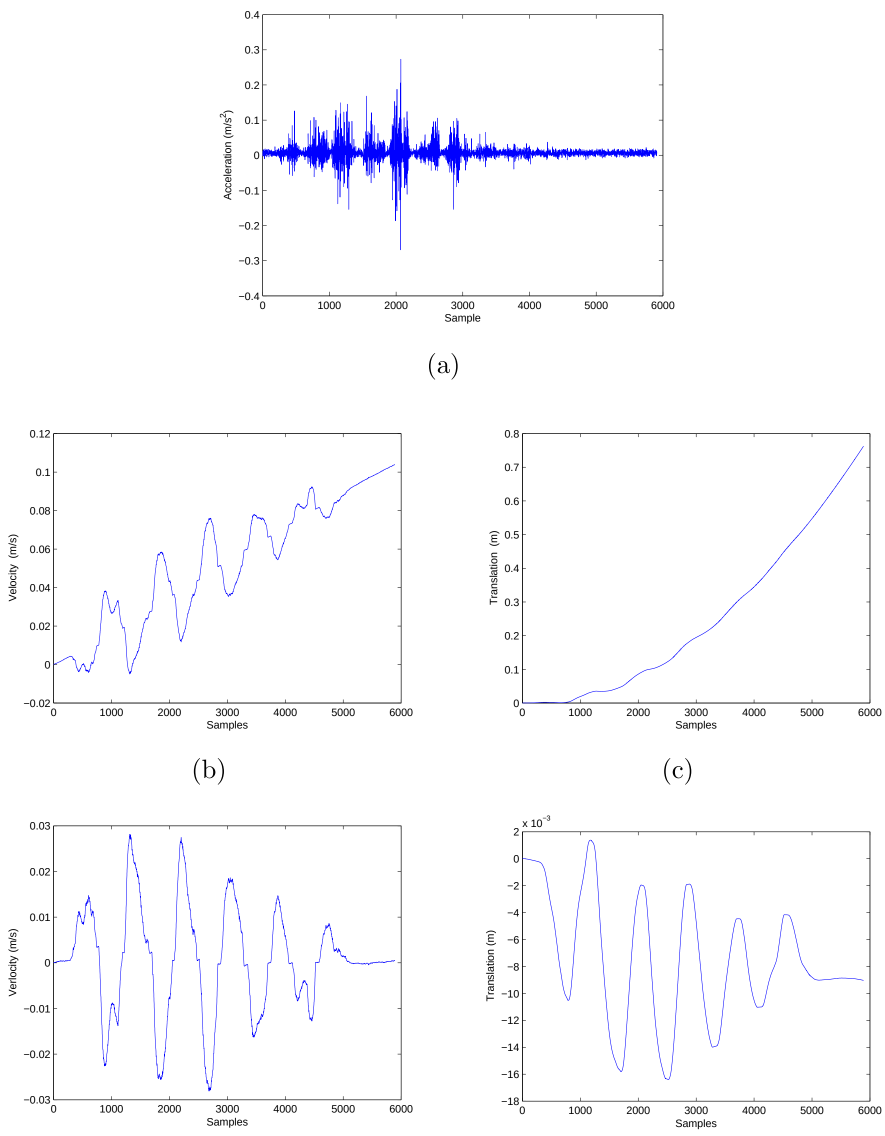
Let's denote acceleration before compensation as $a[n]$ and after compensation as $\hat{a}[n]$, where n is the sample number captured. From n_0 to n_1 and from n_2 to n_3 are the intervals in which the accelerometer is stopped, but the accelerometer is moving during the interval from n_1 to n_2. We compensate the bias in the acceleration as follows:

$$\hat{a}[n] = \begin{cases} a[n] - \frac{\sum_{n=n_0}^{n_1} a[n]}{n_1 - n_0} & n_0 \leq n \leq n_1, \\ a[n] - \frac{\sum_{n=n_0}^{n_1} a[n]}{n_1 - n_0} - (\frac{\sum_{n=n_2}^{n_3} a[n]}{n_3 - n_2} - \frac{\sum_{n=n_0}^{n_1} a[n]}{n_1 - n_0})\frac{n - n_1}{n_2 - n_1} & n_1 < n < n_2, \\ a[n] - \frac{\sum_{n=n_2}^{n_3} a[n]}{n_3 - n_2} & n_2 \leq n \leq n_3. \end{cases} \tag{6.2}$$

Clearly, we remove an average acceleration when the accelerometer is stopped during the intervals from n_0 to n_1 and from n_2 to n_3 (averages are different), and a linear ramp in the interval from n_1 to n_2, i.e., when we move the accelerometer.

After bias compensation in acceleration, we obtain a relatively drift-free velocity but not translation (Figure 6·57 (d) and (e)). Since we are unable to fully compensate the translation drift, we decided to compare the accelerometer results and camera results in terms of velocities.

We encountered another issue. While the temporal sampling frequency (frame rate) of our camera is 30Hz, the sampling frequency of the acclerometer is 400Hz. In order to make the estimates from the camera and the data from the accelerometer comparable, we first interpolate the estimated camera translations to the same sampling rate as the data from the accelerometer. Then, we take the first-order derivative of the interpolated translation to obtain velocity. At the same time, we compensate the bias in acceleration using equations (6.2) and integrate the compensated acceleration to obtain velocity. Then, we calculate normalized cross-correlation between the velocity obtained from camera and from accelerometer. This will provide us with a measurement of performance of the camera motion estimation algorithm.



(a)

(b)

(c)

(d)

(e)

Figure 6·57: (a) Acceleration measured by accelerometer. (b) Velocity and (c) translation computed by integration and double integration, respectively, of acceleration from (a). Note the drift in velocity and translation caused by acceleration bias (see text for discussion). (e) Velocity and (e) translation obtained by integration after compensating the bias in acceleration. Cleraly, the drift has been significantly reduced.

6.3.1 Table-top experiments

In our initial experiments, we used an approximately level table top. We moved the camera-accelerometer assembly along either the x-axis or z-axis. We compared the x and z velocities obtained from the accelerometer and camera. As can be seen in Figure 6·58, the estimated velocity along x-axis of the world coordinate system is quite similar in both cases. The normalized cross-correlation between these waveforms is 0.4773. In case of the z-axis movement, the velocity waveforms are even more similar (Figure 6·59). This is confirmed by their high normalized cross-correlation factor of 0.9138. The relatively low normalized cross-correlation of 0.4773 in the former case is likely due to a bias that could not be fully compensated for by the method described by equation (6.2).

We also applied phase correlation to the video captured by the camera moving along x-axis and took the first-order derivative of the interpolated estimated translation to obtain velocity (Figure 6·60). Again, we calculated the normalized cross-correlation between the estimate from phase correlation and the measurement from accelerometer, and obtained 0.3001. Clearly, phase correlation does not perform as well in this case as the homography decomposition method. We did not test phase correlation on camera translation along the z-axis since the method cannot deal with the diverging/converging effects of such motion.

6.3.2 Platform-based experiments

Our recently-completed mechanical platform allows more precise leveling of the $x-z$ plane in which the camera-accelerometer assembly moves. This reduces the acceleration bias and allows a more accurate compensation using equations (6.2). Following the same procedure as before, we performed the tests by moving the camera-accelerometer assembly along x and z axes again. This time, the high correlation

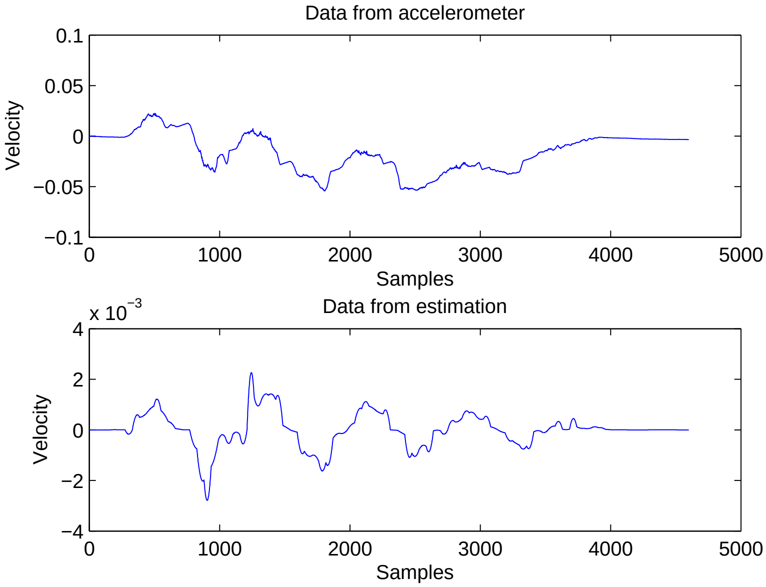


Figure 6·58: Velocities along the x-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method). Normalized cross-correlation: 0.4773.

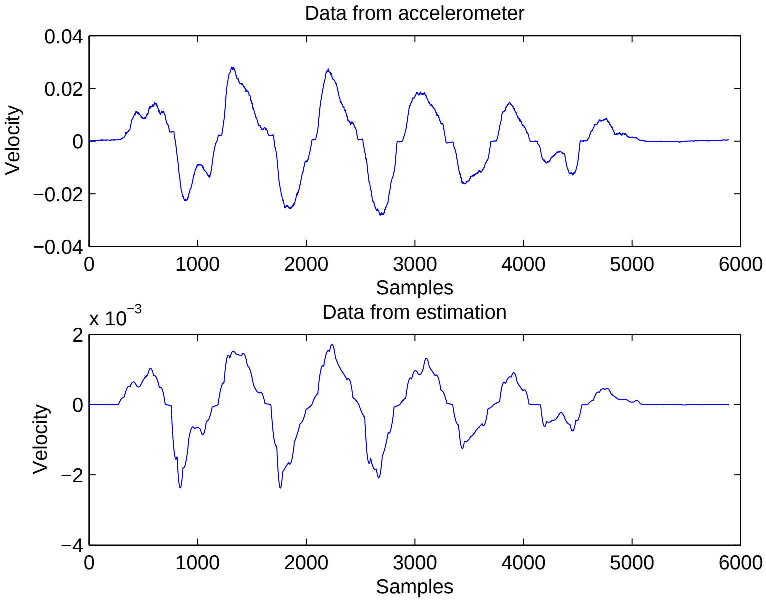


Figure 6·59: Velocities along the z-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method). Normalized cross-correlation: 0.9138.

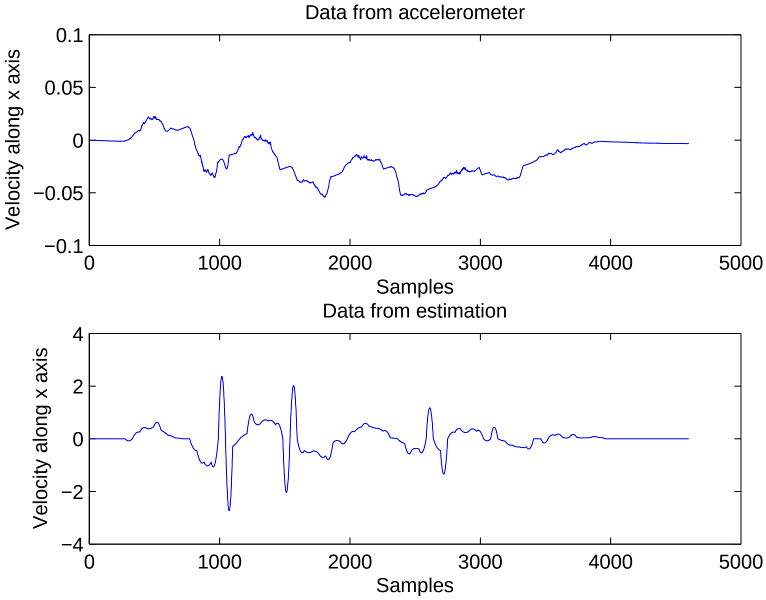


Figure 6·60: Velocities along the x-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (phase correlation). Normalized cross-correlation: 0.3001.

factors of 0.9212 for x-axis motion and 0.8291 for z-axis motion indicate that the homography decomposition method produces 3D camera motion estimates that closely follow accelerometer measurements, as long as the bias is compensated well (Figures 6·61 to 6·62). This suggests that, at least in our laboratory setting, a video camera can serve as a surrogate for accelerometer.

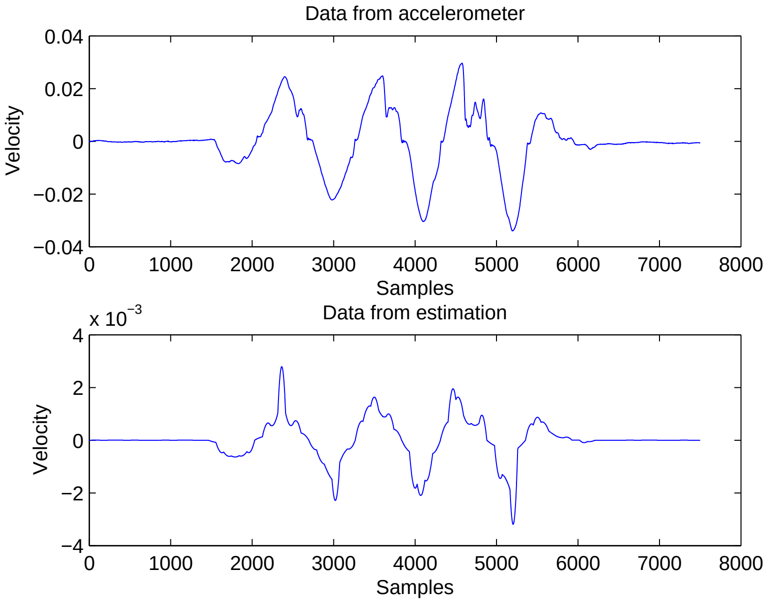


Figure 6·61: Velocities along the x-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method tested on calibrated platform). Normalized cross-correlation: 0.9212.

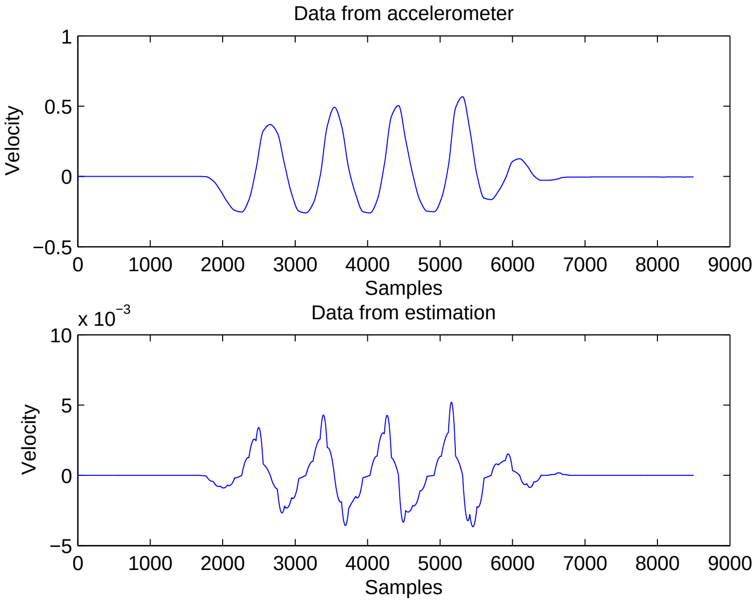


Figure 6·62: Velocities along the z-axis of the world coordinate system obtained from the accelerometer and from camera-based estimation (homography decomposition method tested on calibrated platform). Normalized cross-correlation: 0.8921.

Chapter 7

Conclusions

Below, we draw conclusions from the experimental results obtained by phase correlation, calibration method (Zhang, 98) and decomposition method (Faugeras and Lustman, 1988). In addition, we discuss possible future work in order to move our experiments from the laboratory setting to a real-world environment.

7.1 Discussion of results

Based on experimental results using using the decomposition method (Faugeras and Lustman, 1988) on synthetic data and real data, as well as using phase correlation and calibration (Zhang, 98), we can make the following conclusions:

1. Phase correlation discussed in Chapter 3 is a simple method to estimate 2D displacements from similar images. The advantages of phase correlation are it's computational efficiency and good performance under pure translations parallel to image plane. However, phase correlation cannot deal with 3D displacements, rotations or large objects with local (deformable) motions.

2. The calibration method is a powerful tool to relate the world coordinate system to the camera coordinate system or between two camera coordinate systems. It is an essential step in order to locate a camera or an object in the world coordinate system. With the method described by Zhang (Zhang, 98), it is very easy to calibrate a camera with the world using a chessboard pattern. During

the calibration procedure, we calculate both intrinsic parameters such as focal length and skew factor, and extrinsic parameters such as rotation matrix and translation vector. However, since we seek the extrinsic parameters, calculating intrinsic parameters is redundant and unnecessarily time consuming. In addition, this method requires 3D information about the target which is impossible to provide in practice (thousands of cameras).

3. The decomposition method (Faugeras and Lustman, 1988) seems the most suitable technique for our tasks. As long as the homography constraints are satisfied, i.e., the scene can be approximated by a planar surface or the scene is static and the camera motion is a pure rotation around its optical center, we can always decompose a homography into 3D translation and rotation. As the experimental results on synthetic data showed, the method is very precise in an ideal environment and robust to noise and focal-length inaccuracies. From the experimental results on real data, we can conclude that this method estimates translation and rotation accurately and the results are comparable to measurements from accelerometer. Furthermore, the high normalized correlations indicate that camera can serve as a vibration sensor under assumptions of sufficiently long zoom and object planarity in the field of view.

7.2 Future work

Although we demonstrated that estimation results on both synthetic and real data are quite accurate, in our experiments, we manually extracted corresponding feature points and the target object was limited to a chessboard pattern. In order to consider this system for real surveillance setting, a number of improvements are needed.

First, on must develop a method to identify planar objects in the field of view of the camera (e.g., buildings, road surfaces). With planar areas identified, one must be able to reliably identify unique features in those areas without human operator input. A number of feature descriptors can be considered for this purpose, such as Harris detector, SIFT (Lowe, 1999), etc. Finally, one must develop a robust method to establish correspondent between features in a video frame and a reference (e.g., initial) frame. This is a well studied problem by rich literature (Faugeras, 1993).

References

Burger, W. and Bhanu, B. (1994). A geometric constraint method for estimating 3-d camera motion. In *IEEE International Conference on Robotics and Automation*, pages 1155–1160.

Dufaux, F. and Konrad, J. (2000). Efficient, robust and fast global motion estimation for video coding. *IEEE Transactions on Image Processing*, 9(3):497–501.

Faugeras, O. (1993). *Three-Dimensional Computer Vision*. The MIT Press.

Faugeras, O. and Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment. Technical Report RR-0856, National Institute for Research in Computer Science and Control.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

Koga, T., Iinuma, K., Hirano, A., Iijima, Y., and Ishigur, T. (1981). Motion compensated interframe coding of video conferencing. *Image Processing National Telecommunications Conference*, pages G5.3.1–G.5.3.5.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, 2:1150C1157.

Malis, E. and Vargas, M. (2007). Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, National Institute for Research in Computer Science and Control.

Zhang, Z. (98). A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, Redmond, WA.

CURRICULUM VITAE

Yuecheng Shao

Business: 857-334-2285, ycshao0402@gmail.com, http://ycshao.blogspot.com/,
Date of birth: 4/2/1986

Educational Background

M.S., Electrical Engineering (Signal Processing and Communication) 2011
Boston University, Boston, MA, GPA: 3.9/4.0
B.S., Microelectronics 2009
Fudan University, Shanghai, China, GPA: 3.1/4.0

Project Experience

Experimental validation of surveillance camera as a vibration sensor
Measure vibration of cameras precisely based on content
Auto-reconstruct text on board using stereo cameras
Reconstruct text on board for online use
3D face localization and reconstruction from two cameras
Reconstruct human faces using stereo cameras
Google Challenge: Video genre classification
Classify videos based on their story lines extracted from relating main characters in videos
Template-based tracking
Find an object and track it using normalized correlation coefficient
Copy detection
Detect copies of original images using DCT-based and Covariance Matrix-based methods