Contents

List of Figures **v**

List of Tables **ix**

Nomenclature x

List of symbols x

1 Introduction **1**

1.1 Purpose of this thesis 1

1.2 Scope 2

1.3 Initial assumptions 3

1.4 Thesis outline 3

2 Fundamentals **5**

2.1 3D reconstruction 5

2.1.1 Camera model 6

2.1.2 Feature extraction and corresponding points matching 7

2.1.3 Epipolar geometry 9

2.1.4 Fundamental & Essential Matrix estimations 9

2.1.5 Essential matrix decomposition 12

2.1.6 Points Triangulation 13

2.2 Structure from Motion 14

2.2.1 3D Pose Estimation 15

2.2.2 Structure Adjustment 15

2.3 Mobile Sensors overview 15

2.3.1 Accelerometer 16

2.3.2 Gyroscope 16

2.3.3 Sensor Fusion 17

3 Related Work **20**

4 Problem definition and general implementation aspects **22**

4.1 Requirements 23

4.2 Choosing Environment 23

4.3 "Sensor Enhanced Images Camera" - Android Gradle based project . . . 23

4.3.1 Installation 24

4.3.2 Project Structure 24

4.3.3 User Interface 24

4.3.4 Rotation calculation 25

4.3.5 Custom Sensor Data File format 26

4.4 "Enhanced 3D Reconstructer" - OSX CMake-based project 27

4.4.1 Installation 27

4.4.2 Project Structure 27

4.4.3 User Interface 28

4.4.3.1 Test Efficiency 28

4.4.3.2 Test reconstruction 28

4.4.4 Rotation matrix generation 29

5 Reconstruction using only Sensor Fusion data **30**

5.1 Concept 30

5.1.1 Heuristic for translation estimation 31

5.2 Evaluation 32

5.2.1 Test Environment 32

5.2.2 Rotation calculation tests 32

5.2.3 Translation calculation tests 38

5.2.4 Epipolar line calculation 39

6 Reconstruction using Sensor Fusion rotation and 3-point translation **43**

6.1 Concept 43

6.2 Implementation 45

6.3 Evaluation 46

7 Enhanced 8-point and 5-point reconstruction **48**
7.1 Concept . . . 48
7.1.1 Requirements . . . 49
7.1.2 Enhancing epipolar geometry equations with initial rotation matrix 49
7.2 Implementation . . . 50
7.2.0.1 Enhancing epipolar equations . . . 51
7.3 Evaluation . . . 52
7.3.1 Rotation calculation tests . . . 52
7.3.2 Epipolar line calculation . . . 53

8 Structure From Motion with Sensor Fusion **59**
8.1 Concept . . . 59
8.1.1 Reconstruction process strategy . . . 59
8.1.2 Pose estimation . . . 60
8.1.2.1 Rotation enhancements . . . 60
8.1.2.2 Rotation & translation enhancements . . . 61
8.2 Implementation . . . 61
8.2.1 Enhancing pose estimation . . . 61
8.3 Evaluation . . . 62
8.3.1 Acquiring datasets . . . 62
8.3.2 Test Environment . . . 63
8.3.3 Testing initial pair reconstruction methods . . . 63
8.3.3.1 Accuracy - Epipolar lines correspondence . . . 63
8.3.3.2 Time comparison . . . 70
8.3.4 Testing reconstruction strategies . . . 71
8.3.4.1 Accuracy . . . 71
8.3.4.2 Execution time . . . 73
8.3.5 Effectiveness . . . 77

9 Conclusions **82**
9.1 Summary . . . 82
9.2 Dissemination . . . 84
9.3 Problems encountered . . . 84
9.4 Future work . . . 84

10 Additional materials **86**

References **93**

List of Figures

2.1 Model of the perspective camera with two coordinate systems: external W and internal K. Image taken from [**?**]. 6
2.2 OpenCV camera calibration board. Image taken from [**?**]. 7
2.3 Corresponding matches in "Warsaw Gallery Dataset" found for SIFT features and brute-force matcher using written by author desktop application 8
2.4 Outliers found during matching correspondences in two architecture pictures 8
2.5 Epipolar geometry schematic. Picture from [**?**]. 9
2.6 Epipolar lines found in an image of a vase 10
2.7 RANSAC fitting for 2D image 11
2.8 The four possible decompositions of **E** 13
2.9 3D reconstruction from 2 images 14
2.10 Bundle Adjustment process overview 16
2.11 Getting rotation angles from gravity vector 18
2.12 Difference in readings of Android's acceleration and linear acceleration . 19

4.1 Android "Sensor Enhanced Image Camera" User Interface overview . . 25

5.1 Schematic of test environment for capturing images along Sensor Fusion data 34
5.2 Image of testing environment for capturing images along Sensor Fusion data 35
5.3 Image of marker used to indicate position for test capturing. 35
5.4 Points matches selected by the author (Images 0.jpg and 1.jpg). 36

5.5 Points matches selected by the author with additional outliers (Images 0.jpg and 1.jpg). 36
5.6 Automatically matched points with usage of Sift descriptors and Brute-Force matcher (Images 0.jpg and 1.jpg). 36
5.7 Points matches selected by the author (Images 0.jpg and 2.jpg). 37
5.8 Points matches selected by the author with additional outliers (Images 0.jpg and 2.jpg). 37
5.9 Automatically matched points with usage of Sift descriptors and Brute-Force matcher (Images 0.jpg and 1.jpg). 37
5.10 Rotation tests comparison in Sensor only based approach - 1st example 38
5.11 Rotation tests comparison in Sensor only based approach -2nd example 38
5.12 Screenshot from table of results conducted on Sensor Fusion data based translation estimation during walking with smartphone. In red are indicated values, which were highly unexpected, especially in Y axis, which is perpendicular to earth surface 39
5.13 Visualisation of epipolar lines for 8-point Fundamental matrix estimation - 1st example 40
5.14 Visualisation of epipolar lines for 8-point Fundamental matrix estimation with outliers - 1st example 40
5.15 Visualisation of epipolar lines for 8-point Fundamental matrix estimation with outliers from automiatically matched correspondences - 1st example 40
5.16 Visualisation of epipolar lines calculated from rotation and translation estimated from Sensor Fusion readings - 1st example 41
5.17 Visualisation of epipolar lines for 8-point Fundamental matrix estimation - 1st example 41
5.18 Visualisation of epipolar lines for 8-point Fundamental matrix estimation with outliers - 1st example 41
5.19 Visualisation of epipolar lines for 8-point Fundamental matrix estimation with outliers from automiatically matched correspondences - 1st example 42
5.20 Visualisation of epipolar lines calculated from rotation and translation estimated from Sensor Fusion readings - 2nd example 42

6.1 Visualisation of epipolar lines for 3-point translation estimation and Sensor Fusion rotation approach - 1st example 47
6.2 Visualisation of epipolar lines for 3-point translation estimation and Sensor Fusion rotation approach - 2nd example 47

7.1 Rotation tests comparison for all proposed approaches - 1st example . . 53
7.2 Rotation tests comparison for all proposed approaches - 2nd example . . 54
7.3 Comparison of epipolar lines for 8-point standard and enhanced versions - 1st example 55
7.4 Comparison of epipolar lines for 5-point standard and enhanced versions - 1st example 56
7.5 Comparison of epipolar lines for 8-point standard and enhanced versions - 2nd example 57
7.6 Comparison of epipolar lines for 5-point standard and enhanced versions - 2nd example 58

8.1 Sum of Sampson Error depending on evaluated method 64
8.2 Sampson Error per corresponding point depending on evaluated method 65
8.3 The results of drawing estimated epipolar lines for the Warsaw University Dataset with 300 Sift points (1st group) 66
8.4 The results of drawing estimated epipolar lines for the Warsaw University Dataset with 300 Sift points (2nd group) 67
8.5 The results of drawing estimated epipolar lines for the Warsaw University Dataset with 1000 Sift points (1st group) 68
8.6 The results of drawing estimated epipolar lines for the Warsaw University Dataset with 1000 Sift points (2nd group) 69
8.7 Execution time of the proposed algorithms depending on initial SIFT feature set size 70
8.8 Influence of Bundle Adjustment on the models produced with different reconstruction strategies 72
8.9 Total reconstruction execution time per SIFT features set size 74
8.10 Total execution time of reconstruction with Bundle Adjustment per SIFT features set size 75

8.11 3D point clouds before and after Bundle Adjustment for enhanced 8-point with enhanced pose estimation 76
8.12 Reconstructed models for the proposed initial reconstruction methods and 4000 SIFT features 78
8.13 Reconstructed models for the proposed initial reconstruction methods and 400 SIFT features 79
8.14 Fail test case of Standard 8-point triangulation in comparison to fortunate reconstruction 79
8.15 Pose estimation methods comparison 80
8.16 Reconstruction results from known translations and rotations from different angles 81

10.1 Reconstruction results of "Gate" dataset 89
10.2 Reconstruction results of "Pole" dataset 90
10.3 Reconstruction results of "Gallery Back" dataset 91
10.4 Reconstruction results of "Gallery Front" dataset 92

List of Tables

10.1 Efficiency table of proposed methods for 100 SIFT features in Warsaw University of technology dataset 88
10.2 Efficiency table of proposed methods for 500 SIFT features in Warsaw University of technology dataset 88
10.3 Efficiency table of proposed methods for 1000 SIFT features in Warsaw University of technology dataset 88
10.4 Efficiency table of proposed methods for 5000 SIFT features in Warsaw University of technology dataset. 88

Nomenclature

A The area of the needle point.

N The number of angels per needle point.

a The number of angels per unit area.

List of symbols

$nslats$ The number of slats [−]

Chapter 1

Introduction

Mobile and wearable devices are becoming more and more popular. Modern smartphones are not only equipped with first-rate cameras, but also use advanced sensors, like accelerometers, gyroscope, magnetometer, barometer etc. There is also a big need for Augmented Reality (AR) and Virtual Reality (VR) applications which is evidenced by the constantly growing market of such solutions. That is why image analysis and recognition, as well as 3D reconstruction techniques are important directions in the development of the modern technology and computer science. Unfortunately, algorithms that support these techniques are very time and memory consuming and this makes them difficult to be run on mobile devices, which have many limitations in terms of CPU speed and RAM memory capacity. Those algorithms can be improved with additional data, even if these data are noisy or have limited precision.

1.1 Purpose of this thesis

The author of this thesis will present the reader with an overview of the idea of 3D reconstruction techniques from multiple images and a description of related research conducted in this area. After analysis of efficiency, accuracy and common problems of reconstruction algorithms, the author will propose his own algorithm aimed at enhancing the process with the use of data acquired from Android Sensor Fusion. In particular, an approach of using such sensor data for initial camera pose estimation in order to resolve ambiguities of 3D reconstruction techniques will be presented. Such ambiguities are especially hard to resolve, when there is hard to find proper matching point

correspondences in images. The author shows how standard algorithms which focus on finding relative rotation matrix can be modified to use Sensor Fusion data in order to estimate rotation matrix and focus only on finding "error-correcting" rotation matrix. To prove author's theoretical assumptions two applications were created: Android Application to capture images along with Sensor Fusion data and desktop C++ standalone application to perform test of proposed algorithms. Both Android and desktop applications will be discussed as regards their essential implementation aspects. This study also explains the testing results in the applied environment and setup. Towards the end, the conclusions, encountered errors and problems are discussed. Finally, future plans for the research development are presented.

TODO This thesis will show how the prior knowledge of rotation or translation acquired via mobile sensor fusion can be used to enhance process of 3D reconstruction from a series of images. When it comes to relying on hand-held smartphones, the collected sensor data are very noisy. This thesis shows how even noisy information can be used in the reconstruction processes.

1.2 Scope

The author researched how the sensor fusion of accelerometer, gyroscope and magnetometer data (in particular that one performed on Android platform) can be used in order to improve fundamental and essential matrix calculations as well as their decomposition to the relative pose estimation. In the beginning author had an idea to relay solely on rotation and translation estimation from sensor data, but unfortunately, the raw data sensors are too noisy and therefore not reliable enough to rely solely on them in the course of enhancing the 3D reconstruction [**?**]. Nevertheless these heuristic measurements of positions, especially translation between cameras were also conceptualised and implemented by author and are part of this thesis. The alternative 3-point algorithm for camera translation estimation has also been implemented by author and is discussed herein. Although this "3-point translation" algorithm used along with Sensor Fusion camera rotation estimation proved to be not accurate enough, it was important step of author's research and is described in this thesis. In this research Sensor Fusion data enhanced 8-point and 5-point reconstruction algorithms are introduced, which are the derivative of theoretical discussion on using additional Sensor Fusion

data in spatial reconstruction techniques. It is also researched how the improved versions of the standard 8-point and 5-point algorithms influence convergence speed and error reduction of Bundle Adjustment, which is also a part of modern 3D reconstruction process. This thesis covers not only initial images pair reconstruction, but also relative pose estimation methods for Structure from Motion computation, when series of images are available. Unfortunately, the best methodology for images corresponding feature matching is not the subject of this research. This is also the reason, why author does not discuss, implement or present dense 3D reconstruction pipeline. Therefore all presented reconstructed models consist only from white supporting points and have no texture rendered on them.

1.3 Initial assumptions

The field of 3D reconstruction is quite broad, that is why the author of the thesis focuses primarily on certain aspects thereof. First of all it was not the author's intention to write all algorithms anew, but to built his versions based on the OpenCV library and open-source project called "Relative Pose Estimation" created by Bo Li [**?**]. In terms of sensor fusion, the current state-of-the-art approach can be found and accessed in Android platform SDK. Due to limitations of compilation and debugging of C++ code on Android platform, 3D reconstruction will be processed in a separate desktop application. What is important that such approach does not influence value of this research. The main goal of this document is to research possibilities of enhancing standard algorithms used for 3D reconstruction with additional sensor data. Once this research is done all coding work regarding implementing new 3D reconstruction pipeline in C++ can be easily ported with help of Android Native Development Kit (NDK).

1.4 Thesis outline

In **Chapter 2** the fundamental theory behind 3D reconstruction and Structure from Motion is discussed. Also sensors used to perform sensor fusion on Android platform are introduced and their strengths and weaknesses are described.

Chapter 3 contains a overview of related research. The discoveries of utmost importance, which lead to creating the concept of the proposed methods are also described

there.
In **Chapter 4**
Chapter 5
TODO
TODO
TODO

The objective of **Chapter 9** is to present and discuss the evaluation of the proposed initial pair reconstruction methods and different strategies of Structure From Motion computations.
Chapter 10 presents the author's conclusions observed in the course of investigating the subject and implementing ideas, as well as the encountered problems. The author also outlined the further development plans for the researched solutions and approaches.

Chapter 2

Fundamentals

This chapter explains the basic theory behind 3D reconstruction and Structure from Motion. All information referred to herein can be found in [**?**]. It also includes a brief overview of sensors, which can be used to calculate global or relative orientation of the smartphone camera. It is also explained, why it is necessary to use accelerometer, gyroscope and magnetometer together by combing them into Sensor Fusion.

2.1 3D reconstruction

In general 3D reconstruction is the process of automatic creation of 3 dimensional objects models from images. There are many possibilities of performing 3D reconstruction, starting from two-view reconstruction, where only two images are used to multiple-view reconstruction, where image sequence is bigger than 2. Reconstruction can be performed either with a single hand-held camera from images sequence or simultaneously working and synchronised multiple cameras. It can be done either on spot in real time or later for instance in laboratory. As few as two pictures taken from different angles of a single object are sufficient to perform 3D model generation. In general reconstruction process consists of the following steps:

1. **Image Acquisition**, where images frames are acquired
2. **Feature extraction and corresponding points matching**, where distinctive features are extracted from the images and compared
3. **Fundamental & Essential Matrices computation**, where matrices meeting the requirements of basic epipolar geometry are calculated

4. **Camera parameters estimation**, where external and internal camera parameters, like translation and rotation are estimated
5. **Triangulation**, where camera projection matrices are composed and used in order to calculate 3D cloud points

2.1.1 Camera model

To get reader acquainted with nomenclature used when reading about camera related operations following section was written. In general two coordinations system are related to camera model, which can be seen in Figure 2.2:

1. The external coordinate system (denoted here with a subscript **W** for world) which is independent of placement and parameters of the camera.
2. The camera coordinate system (denoted by **C**, for camera).

Relation between those two coordinate systems is expressed by translation matrix **T** and rotation matrix **R**. These two together are often referred as external camera parameters. Internal camera parameters are expressed by the following matrix:

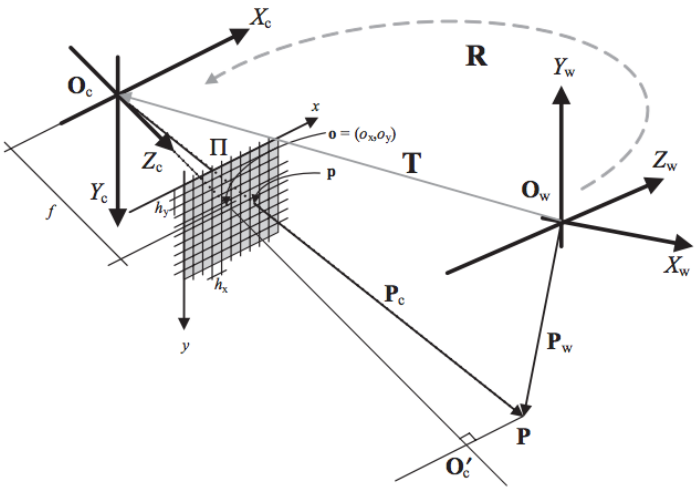

Figure 2.1: Model of the perspective camera with two coordinate systems: external W and internal K. Image taken from [**?**].

$$\mathbf{K} = \begin{bmatrix} h_x & 0 & o_x \\ 0 & h_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where h_x and h_y represent the focal length of the camera expressed in pixel dimensions in the x and y direction respectively. Similarly, point $\mathbf{o} = (o_x, o_y)$ is the principal point of camera in pixel dimensions. These parameters need to be calculated only once for each camera model. Once they are known, camera can be described as calibrated. Calibration of cameras can be done with special reference boards of the known dimensions and characteristics. One of such boards (Figure) as well as sample code that allows to calibrate camera is available on OpenCV website [**?**]. Knowledge of external and

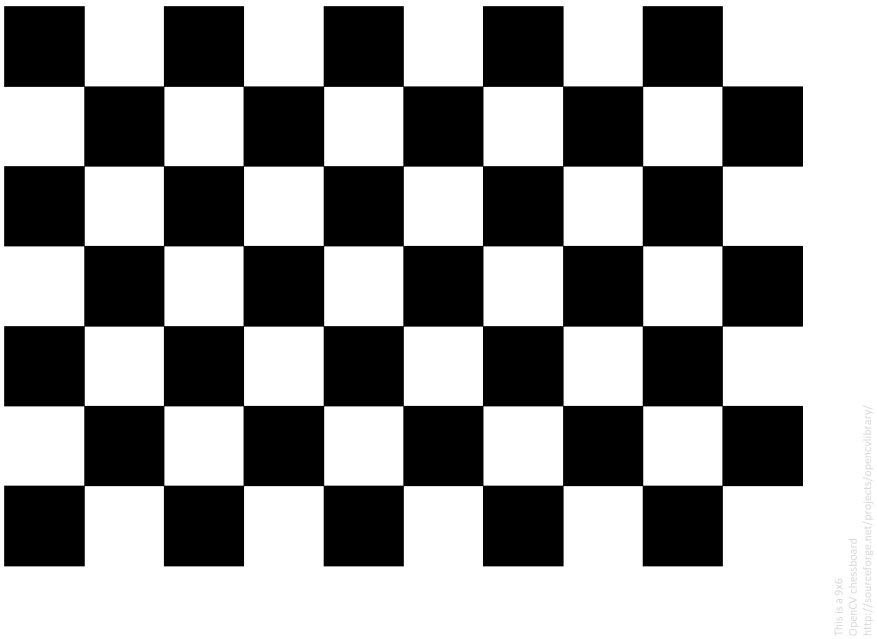

Figure 2.2: OpenCV camera calibration board. Image taken from [**?**].

internal camera parameters allows us to switch between camera and world coordinates system and is necessary for proper 3D model reconstruction.

2.1.2 Feature extraction and corresponding points matching

Usually, each image used in reconstruction has to be analysed in order for the distinctive features to be found. Afterwards all features in images are compared in order to find corresponding matches between them (Figure 2.3). There are multiple features detectors and extractors available for use [**?**]. Some of them are more suitable for edge detection, while others are best used for corner or blob based feature detection. One of the most popular and robust feature detection method is scale-invariant feature

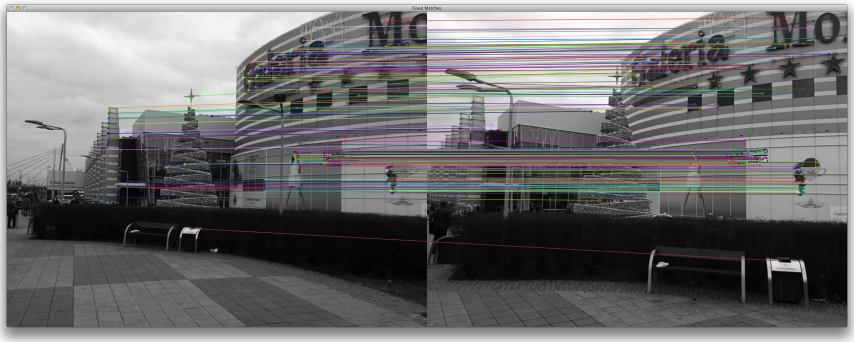

Figure 2.3: Corresponding matches in "Warsaw Gallery Dataset" found for SIFT features and brute-force matcher using written by author desktop application

transform (SIFT) [**?**]. The use of these descriptors allows for detecting local features in images and describing them with special metrics, which are scale, rotation and translation invariant. Unfortunately even using such complex descriptors this process still is really hard and difficult. Algorithms that do these types of matching are usually $O(n^2)$ and therefore it takes a lot of time to find proper correspondences, especially when good object (image) resolution is needed. What is more even the most distinctive descriptors do not give global unique values and thus a lot of outliers can be produced during matching process (Figure 2.4). This is one of the first problems, which do not allow for perfectly accurate and error free 3D reconstruction.

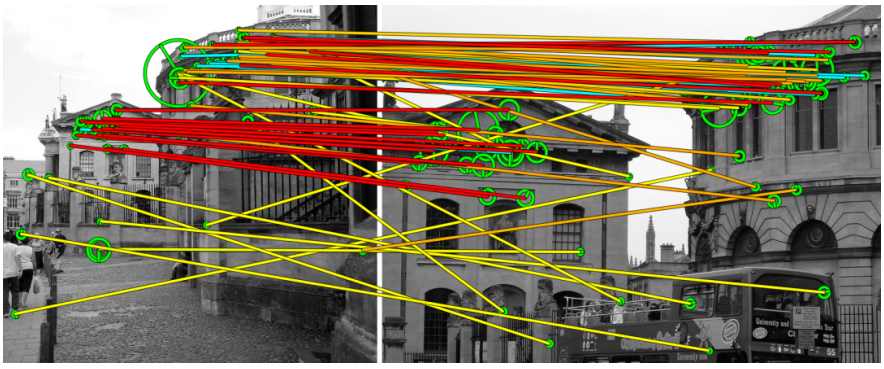

Figure 2.4: As shown in image example a lot of points in images can be improperly matched. This can influence further effectiveness of 3D reconstruction [**?**]

2.1.3 Epipolar geometry

In Figure 2.5 epipolar geometry schematic is shown. Baseline is the line connecting two cameras center points. Points in which baseline is crossing cameras image planes are called epipoles ($\mathbf{e}_l$ and $\mathbf{e}_r$). Lines that are going through epipoles and corresponding feature points $\mathbf{p}_l$ and $\mathbf{p}_r$ are called epipolar lines. Rays coming from cameras center points through corresponding feature points cross in position of 3D point $\mathbf{P}$ of photographed object. Figure 2.6 shows different perspective on epipolar lines and example

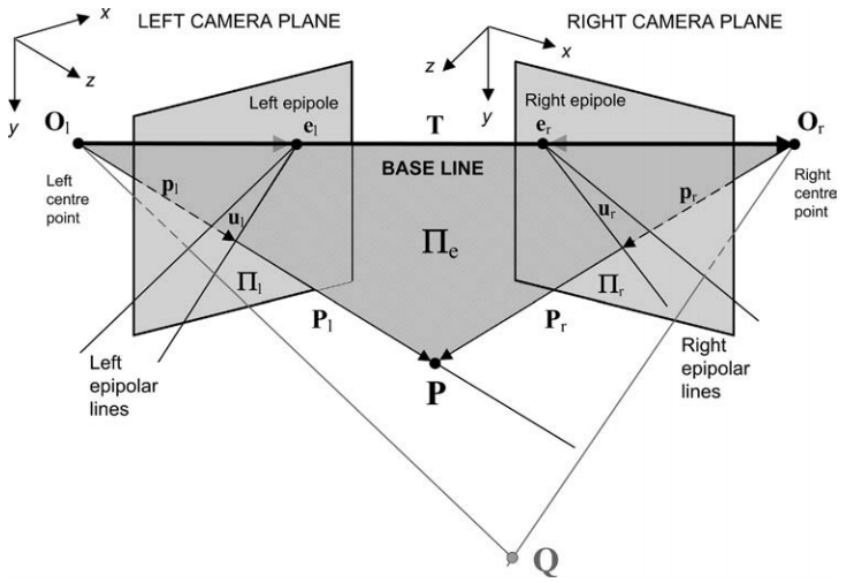

Figure 2.5: Epipolar geometry schematic. Picture from [**?**].

of their visualisation in image planes. These lines cross the exactly same points in both images and can be used for dense feature matching since matches need to be searched exclusively in the surroundings of these lines. However they can be calculated only if essential or fundmental matrix is determined.

2.1.4 Fundamental & Essential Matrix estimations

Once proper feature matches are found in two images, it can be proven that there exists Fundamental matrix F for which the following equation is satisfied:

$$\mathbf{x}'^{T}\mathbf{F}\mathbf{x} = 0 \tag{2.2}$$

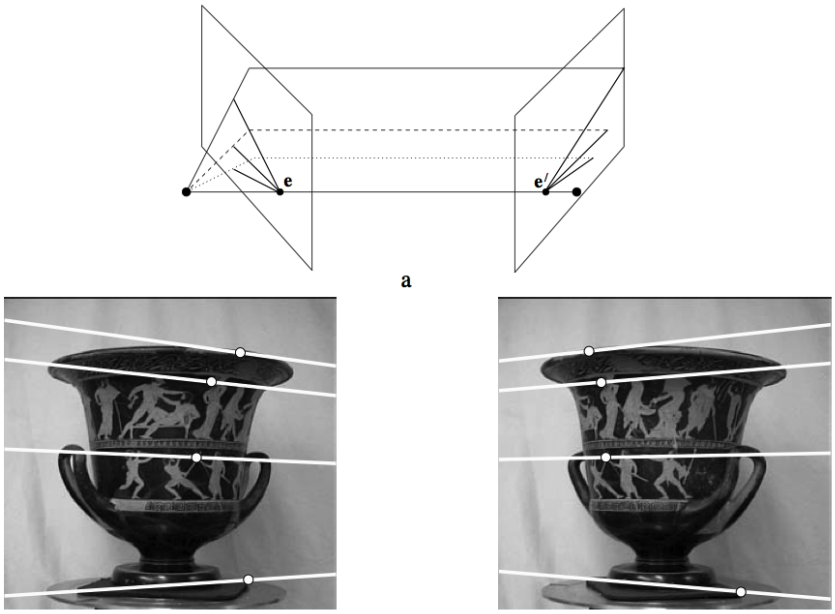

Figure 2.6: Epipolar lines found in an image of a vase [**?**]. It is good to spot that all points that lay on corresponding epipolar lines and are visible in both images can be easily matched

where x and $x^{'}$ are uncalibrated notions of points correspondence [**?**]. It is known that solutions of this equation are highly sensitive to the occurrence of outliers. Usually, to make fundamental matrix estimations more accurate some outliers removing algorithms need to be used. One of the most robust approaches includes the use of of RANdom SAmple Consensus (RANSAC) [**?**]. Its basic idea relies on choosing a random subset from among all matches, solving a problem of reduced dataset and establishing how many points from the original set satisfy the equation. When enough inliers are found, points that do not satisfy the equation can be removed from further processing. The example 2 degree-of-freefom problem of sample fitting can be seen in Figure 2.7.

When internal camera parameters K are known, the image points found can be calibrated and expressed in camera reference position system. Such calibrated points satisfy the following essential matrix E equation:

$$\mathbf{x}_c^{'T}\mathbf{E}\mathbf{x}_c = 0 \tag{2.3}$$

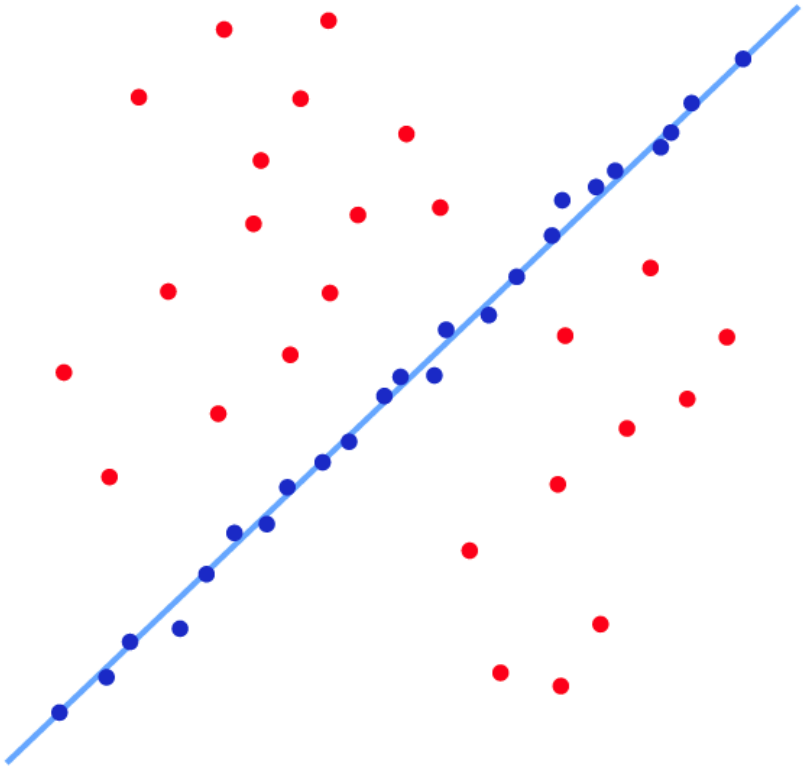

Figure 2.7: RANSAC fitting for 2D image[**?**]. In this example all red points are some existing outliers. Blue line is decided by iterative process of fitting line to randomly chosen subset of all points. When enough percentage of all points lies in the neighbourhood of such fitted line(blue subset), process stops and unnecessary points (red points) can be removed for further processing.

which is very similar to a fundamental equation 2.2. This results in:

$$\mathbf{E} = \mathbf{K}^T\mathbf{F}\mathbf{K} \tag{2.4}$$

with **K** being matrix representing the internal camera parameters.

The most known methods for Fundamental and Essential matrix calculations are:

1. **8-point algorithm** - 8 corresponding points pairs in image must be found and all of them are used in order to resolve 8DOF matrix problem and find fundamental matrix. More informations can be found in [**?**].

2. **5-point algorithm** - for calibrated cameras only 5 points pairs have to be used to resolve 5DOF matrix problem and find proper essential matrix. More informations can be found in [**?**] or [**?**].

Using Equation 2.4 one can go from Fundamental to Essential matrix and back. Next section will explain, how essential matrix is used in order to find relative translation and rotation between cameras.

2.1.5 Essential matrix decomposition

In section 9.6.1 of Multiple View Geometry in Computer Vision ([**?**]) it is shown in detail, how essential matrix can be decomposed using Singular Value Decomposition (SVD) to relative camera rotation **R** and translation **t**. In general SVD relay on decomposition one matrix to three other matrices, which multiplied give the initially decomposed matrix and where middle one is diagonal matrix.
Unfortunately, there are four possible solutions for such decomposition of essential matrix and it is not always possible to identify the correct one, especially when there is lot of unsuccessfully removed outliers present in corresponding points set. Figure 2.8 presents the four possible situations of decomposition **E** to **R** and **T**.
However it is good to note, that only in (a) is the reconstructed point in front of both cameras. Usually to determine proper solution it is enough using each of corresponding feature point to calculate its 3D positions and perform simple test checking if it lies in front of both cameras image planes. It may seem simple task, but from computer perspective it's hard to determine proper decomposition, when not all of outliers were removed during feature matching phase. Often outliers pair will show itself in front of both cameras for not correct solution. This is the place in reconstruction pipeline where additional sensor data can be used to determine proper solution. In fact Sensor Fusion data can limit E matrix decomposition only to two solutions - (a) and (b) cases from Figure 2.8). What is most important that even in presence of many outliers it will produce proper perceptible models (cases (a) and (b) are baseline reversal). This will be explained in detail, when discussing enhancements to 8 and 5 point algorithms.

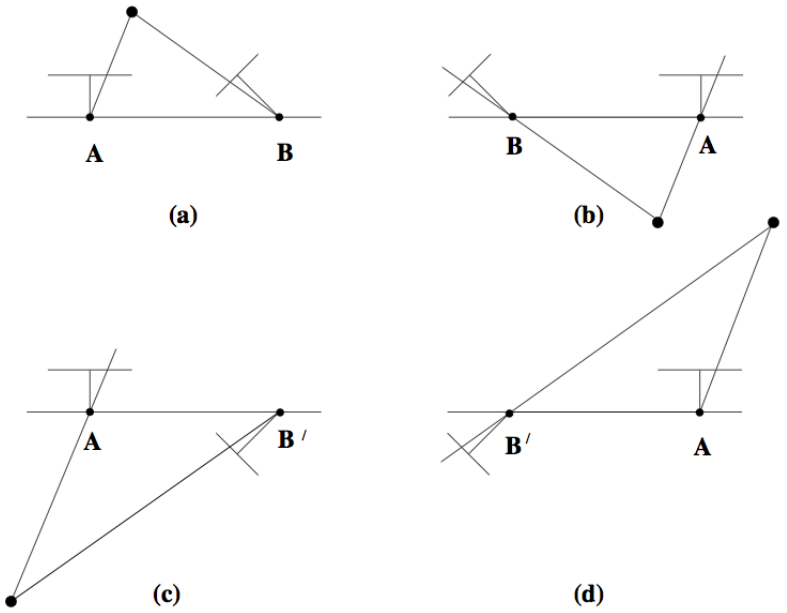

Figure 2.8: The four possible decompositions of **E**. Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about baseline. Picture from page 260 of [**?**].

2.1.6 Points Triangulation

Once internal and external camera parameters are calculated, the triangulation can be performed in order to acquire an affine reconstruction model (Figure 2.9). Using **R** and **T** acquired from **E** decomposition and **K** from camera calibration, relation between 2D and 3D points can be expressed with following perspective projection matrices:

$$\mathbf{P}_1 = \mathbf{K}\left[\mathbf{I} \mid \mathbf{0}\right] - \text{for first image,} \tag{2.5}$$

$$\mathbf{P}_2 = \mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right] - \text{for second image,} \tag{2.6}$$

where **I** is identity matrix, **0** is zero column vector, **R** is matrix representing relative rotation between cameras and **t** is relative translation vector between cameras. And now respectively:

$$\mathbf{x} = \mathbf{P}_1\mathbf{X} - \text{for first image,} \tag{2.7}$$

$$\mathbf{x}^{'} = \mathbf{P}_2\mathbf{X} - \text{for second image,} \tag{2.8}$$

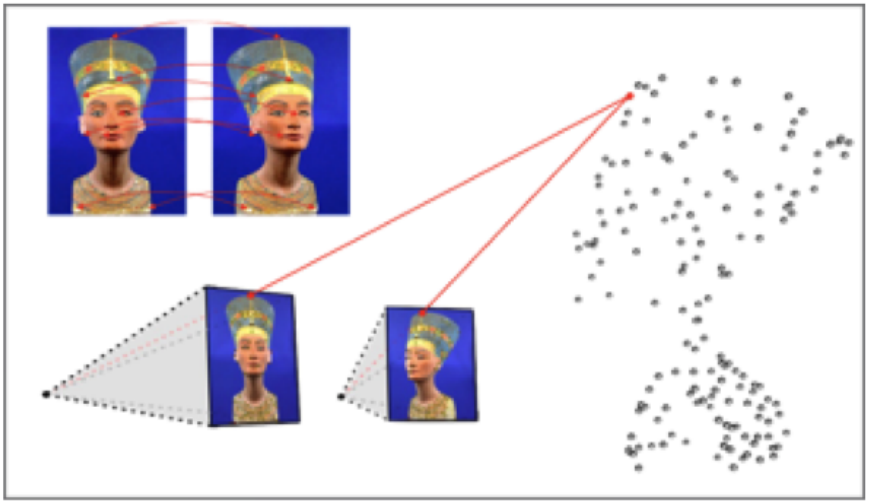

Figure 2.9: 3D reconstruction from 2 images. Two rays crossing cameras centres and corresponding images points also cross in position of 3D point. [Slide from Photogrammetric Computer Vision from 2013/2014 winter semester at TU Berlin]

where **X** is 3D position of point in space.

Basically having relative cameras positions fully established (known internal parameters, rotation and translation) all rays connecting camera centres and corresponding image pairs will cross in place of their 3D location. Thus, it is good to note here that resolution and density of 3D models rely on picture resolution and number of matching correspondences found. The only element that cannot be determined in such a case is the scale, because only relative positions between cameras can be established. This process is described in more details in Chapter 10 of Hartley's book[**?**].

2.2 Structure from Motion

The term "Structure from Motion (SfM)" refers to the reconstruction performed from the consecutive sequences of a moving camera. It is a popular research topic and the two main approaches, namely the Pose Estimation and Homography Estimation, can be used for the purposes of reconstructing a 3D model of an object visible throughout images sequence. Lecture of this section give solid ground for understanding what Chapter8 is about.

2.2.1 3D Pose Estimation

Assuming that some of the 3D cloud points are already known, the matches between 2D features in a new image and 3D point cloud positions can be established. Such 3D-2D matches can be used to estimate the camera position. This allows for reconstructing new 3D points and merging them smoothly into a functional model. Unfortunately, this process is also highly sensitive to the occurrence of outliers, therefore adequate measures have to be undertaken to reduce their influence. One of the main advantages of this method is its speed. On the other hand, its effectiveness relies strongly on the initially existing 3D cloud quality.

2.2.2 Structure Adjustment

Special refinement methods can be used to compensate for an error of mismatched points propagating through images. They include. among others, Bundle Adjustment (BA). The algorithm, using the information concerning the corresponding matches between multiple sets of images, iteratively modifies either both the camera external parameters and 3D points positions or one of them[**?**]. The main disadvantage of the method is its execution time. It is too time-consuming to be used in real-time applications. The basic idea of BA is expressed in Figure 2.10. Theoretically usage of Sensor Fusion initial rotation and translation estimation should produce close to actual values and thus result in better convergence speed during BA.

2.3 Mobile Sensors overview[?]

There are many sensors available in the nowadays smartphones, such as accelerometer, gyroscope, magnetometer, barometer, GPS, etc. All of them have their advantages and disadvantages, and thus the errors of some might be compensated with the strengths of the others in order to, for example, accurately compute camera rotation angles. Very good overview of sensors accuracy on example of Android can be found in Google Tech Talk called: "Sensor Fusion on Android Devices: A Revolution in Motion Processing" [**?**].

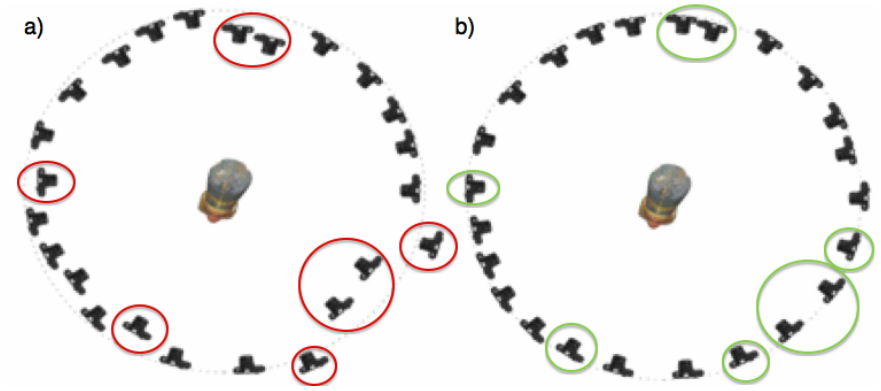

Figure 2.10: Bundle Adjustment process overview. Initially pictures around Pharaoh head from exactly the same distance on the circle. Due to presence of outliers, some of the estimated cameras were improperly estimated and it could look like they were taken from different distances a). However after Bundle Adjustment process whole setup estimation was corrected to case b). Red circles indicates cameras, which positions were corrected in the process to the green ones. [Slide from Photogrammetric Computer Vision 2013/2014 winter semester at TU Berlin]

2.3.1 Accelerometer[?]

An accelerometer is a device that measures acceleration along 3 axes of the device. Generally, an accelerometer allows for measuring total acceleration by sensing what force is applied to its micro strings. An accelerometer, which lies still on a flat surface parallel to the Earth's surface will indicate approximately the value of earth's gravity acceleration (1G) - around 9.8 $\frac{m}{s^2}$. This gravitation vector can be used to calculate the relative camera rotation, but it is difficult to define to which direction the gravity vector points when the device is moving in not a linear manner and acceleration readings are the sum of gravity and movement acceleration. The gravity vector can be obtained from the accelerometer data, when the device is still and later tracked during unexpected movements with the usage of gyroscope sensor data. In case of Android accelerometers acceleration is returned in $\frac{m}{s^2}$.

2.3.2 Gyroscope[?]

A gyroscope is a device for measuring or maintaining orientation, based on the principles of conservation of angular momentum. A standard gyroscope consists of a spinning wheel mounted on two gimbal rings, which allows it to rotate in all three axes. The

spinning wheel will resist changes in orientation, due to an effect of the conservation of angular momentum. A conventional gyroscope measures orientation, in contrast to MEMS (Micro Electro-Mechanical System) types, which measure angular rate, and are therefore called rate-gyros. MEMS gyroscopes contain vibrating elements to measure the Coriolis effect. In the end the angular velocity can be calculated in each axis. It is important to note that whereas the accelerometer and the magnetometer measure acceleration and angle relative to the Earth, gyroscope measures angular velocity relative to the body. In case of Android rate of rotation is returned in $\frac{rad}{s}$.

2.3.3 Sensor Fusion[?]

Sensor fusion is the process of combining sensory data derived from disparate sources in a way that the resulting information is to some extent more valuable than it would be possible should these sources be used individually. The term "more valuable" in this case may mean "more accurate", "more complete" or "more dependable", or refer to the result of an emerging view, such as stereoscopic vision (calculation of depth information by combining two-dimensional images from two cameras at slightly different viewpoints). In Android API available Sensor Fusion is made by combining accelerometer and gyroscope data. It allows on tracking gravity vector, which can be obtained when the device is not moving through rapid movements thanks to gyroscopic reading. The gravity vector can be decomposed in order to estimate the camera's rotation angles. In Figure 2.11 reader can see overview of how gravity vector can be used to calculate rotation angles of the device. Unfortunately as shown for instance in [**?**] and [**?**] the calculated rotation angles may be few degrees drifted even when using Sensor Fusion.

Subtracting gravity from the actual acceleration measurements allows for estimating linear acceleration (Figure 2.12). The use of linear acceleration makes it possible to measure the relative change of the device's translation. From fundamental physics distance traveled during acceleration movement moment is expressed by the following equation:

$$\mathbf{s}_{\delta t} = \mathbf{V}_t \cdot \delta t + \frac{\mathbf{a} \cdot \delta t^2}{2}, \tag{2.9}$$

where $\mathbf{V}_t$ represents current velocity of the device, δt representing time difference, in which acceleration has happened and $\mathbf{a}$ representing current linear acceleration vector.

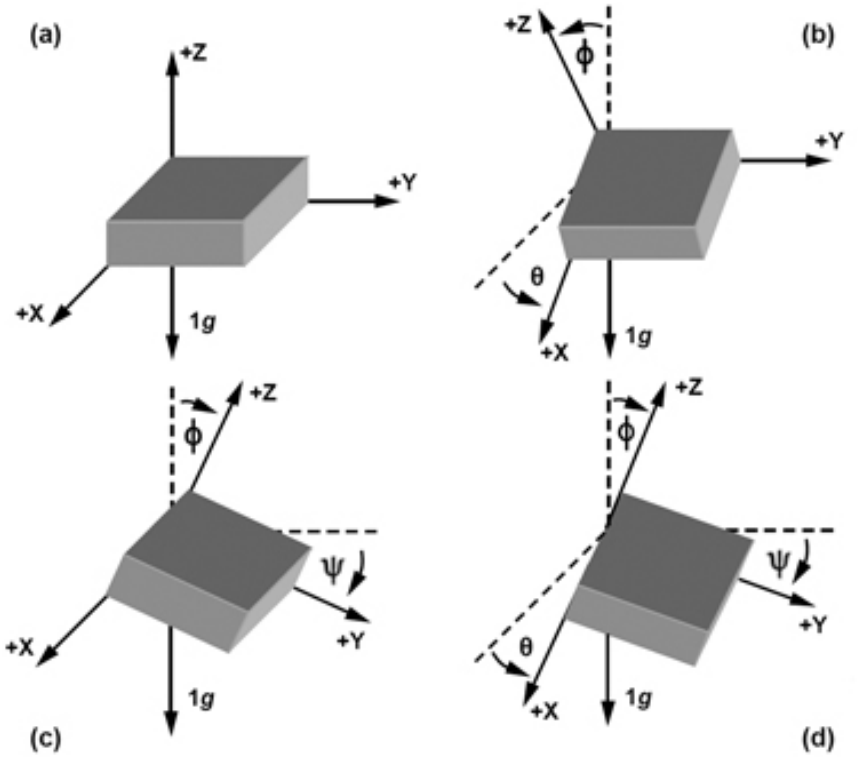

Figure 2.11: Getting rotation angles from gravity vector. In situation a) when device is not moving it is possible to determine the reference gravity vector values along each device axis. Later for situation b) to d) it is possible to calculate angles between current and reference gravity vector values on the corresponding device axes. Image from [**?**]

During acceleration velocity of the device changes according to following equation:

$$\begin{aligned} \mathbf{V}_{\delta t} &= \mathbf{a} \cdot \delta t \\ \mathbf{V}_{t+\delta t} &= \mathbf{V}_t + \mathbf{V}_{\delta t}, \end{aligned} \tag{2.10}$$

However, as reader can see to get translation it is required to integrate acceleration over time twice, which can result in largely increasing the translation's estimation errors, when noisy data are available. As mentioned already in this section, [**?**] presents in particular how noisy mobile sensors prove to be in reality, especially when used during movement, thus translation estimation can be highly inaccurate. Unfortunately there is no good official documentation, where Android linear acceleration reading error was calculated, but one of comprehensive study on Android Sensor Fusion and distance calculation using linear acceleration can be found in [**?**]. Also Chapter 5 includes author's experiments on rotation and translation estimation using Sensor Fusion data.

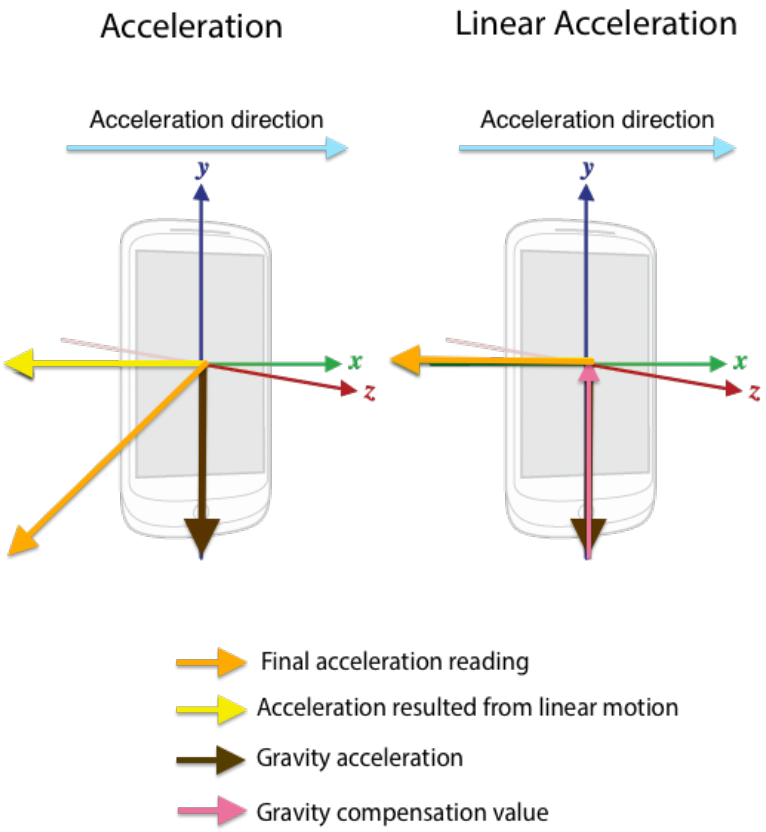

Figure 2.12: Difference in readings of Android's acceleration and linear acceleration. Having gravity vector pointing down (brown) and device accelerating in the right direction, acceleration vector (yellow) will show its minus influence on axis - x. For left situation when gravity vector is not known, final device acceleration readings will indicate orange value. However once gravity vector is known, its influence can be subtracted from acceleration reading, leaving pure device linear acceleration reading - right situation.

Chapter 3

Related Work

There are many approaches to the reconstruction problems, starting with the raw one-by-one pixel analysis for dense reconstruction, through high level abstraction of objects recognition and extraction and ending with the light and shadows-based reconstruction [**?**][**?**]. However this thesis does not focus on high-level abstraction reconstruction, but on refining relative poses estimation steps. As described in the theoretical part, in order to estimate the relative positions of two cameras, an essential matrix must be decomposed. Since the basic epipolar geometry equation discovery, many scientist have introduced various ways to solve this linear problems, which differed primarily in terms of the accuracy and speed. One of the first was the 8-point algorithm [**?**], which can be used to compute a fundamental matrix. This can be done without any prior knowledge of the scene or camera external and internal parameters. Fundamental matrix has to be converted to essential matrix in order for the relative camera positions to be found and that is why it is necessary to calculate internal camera parameters. Next, the 5-point algorithm approaches were introduced [**?**] [**?**]. They used internal camera parameters for the proper essential matrix estimation. In his paper M. David Nister showed that the 5-point algorithm outperforms almost all similar algorithms in terms of accuracy and speed. Only the 8-point algorithm can be regarded as equally efficient. One of the state-of-art real-time and robust approaches is the iterative 5-point algorithm created by Vincent Lui and Tom Drummond[**?**].

Most of the algorithms are very sensitive to the occurrence of outliers. One of the most common approaches to use to tackle this problem is RANSAC modelling [**?**].

A research group from Technishe Universitt Berlin made an interesting comparison and evaluation of the methods discovered by 2008 [**?**]. It was discovered that estimation of camera rotation is much more stable than translation calculation, which means that reconstruction effects are more sensitive to translation estimation errors than rotation matrix estimation. Furthermore, there are a lot of ambiguities regarding the choice of the correct solution of epipolar geometry equation, which cannot be easily resolved especially, when lot of outliers are present in matched features set.

In certain situations, where external camera parameters as rotation and translation can be measured, more accurate algorithms were proposed. In 2011 D. Scaramuzza from Zurich proposed a 1-point algorithm [**?**], which shows how to describe and use a model of a camera mounted on a car to enhance 3D reconstruction. The 4-point algorithm introduced in 2013 using information of rotation angle in certain axis from additional sensor, as shown in paper [**?**], can outperform even the 5-point algorithms. Lately the scientists have been creating more complex models to estimate relative stereometry, e.g. a team from Zurich proposed a way to enhance reconstruction with additional 6DOF sensor [**?**].

There are also non standard approaches like [**?**], where it is shown how to estimate the relative pose from three lines with two of the lines being parallel and orthogonal to the third one. Very accurate estimations can also be obtained for special camera model cases, when there is no rotation[**?**]. All these references show that enhanced models often help to achieve more accurate or faster solutions to standard epipolar geometry problems.

Some of the researches drafted many scientific papers regarding rotation-improved solution space searching [**?**] [**?**]. There are even some approaches involving the use of additional IMU accelerometer for the reconstruction process enhancement [**?**]. This is what inspired author of this thesis to experiment with usage of Sensor Fusion data to enhance 3D reconstruction techniques.

Chapter 4

Problem definition and general implementation aspects

Summing up when it's hard to determine ideal point correspondences between images, it also is hard to determine the proper solution of essential matrix decomposition. However modern cameras for instance in smartphones have an additional sensors, that allow for tracking position in space. It follows from the analysis of the theory and related works that both epipolar equations and pose estimation techniques can be improved by additional rotation and translation information data. This is why initially author wanted to resolve issues of **E** decomposition by direct calculation of **R** and **T** from Sensor Fusion data. Seperate tests for rotation and translation accuracy estimations were made and unfortunately, the first attempts to perform reconstruction proved it to be not accurate enough. This is where author decided to try few different approaches of combining Sensor Fusion data with already known algorithms. Chapters 5 - 7 document conceptual parts, additional important aspects and tests performed during development of algorithms for cameras rotation and translation estimation created by author's research. Chapter 8 includes additional tests performed on images sequence to check their efficiency and influence on 3D spatial (sparse) model creation in pipeline of Structure From Motion methods. But first to help reader go through lecture of this thesis this chapter gives overview of research environment implemented by author.

4.1 Requirements

To perform necessary tests author needed an input in the form of a series of images with additional information about the position of the camera - the euclidean rotation and translation. This is why it was necessary to create an mobile application, which would be able to register Sensor Fusion data during images capturing phase. It wouldn't have to be necessary a smartphone, however in the moment of author's research there was none camera, which would calculate it's rotatation and translation during movement.

4.2 Choosing Environment

Currently, Android has one of the best and open APIs which allows programmers to acquire sensor fusion data of rotation and linear acceleration. This is also why author's wanted to research particularly, how Android Sensor Fusion can be used to enhance 3D reconstruction techniques.
In order to avoid creating anew all epipolar geometry algorithms, C++ version of OpenCV library was used. However image acquiring process was separated from the model reconstruction. In general, the project was split into two sub-projects: Android for images capturing and df for reconstruction tests. This allowed to save a lot of time in error debugging which is highly difficult in native C++ development on Android. Native code can not be easily breakpointed, which makes it hard to determine, what exactly the error is. Also after every code recompilation, it is necessary to upload application to the device, which usually takes few minutes. Fortunately once debugged, reconstruction algorithms written in C++ code can be easily ported to Android thanks to Native Development Kit (NDK).

4.3 "Sensor Enhanced Images Camera" - Android Gradle based project

In order to capture images and associate them with sensor data, a custom photo capture application called "Sensor Enhanced Images Camera" was created. Using this application, the camera orientation angles can be tracked in real-time. Also linear acceleration, current velocity and relative translation from the location of the last picture taken can be tracked. It was expected that translation calculation error will rise

quickly, therefore translation is measured relatively to previously taken image. Also movement heuristic constraint was proposed to avoid unexpected rise of device velocity. In particular this research and applications were adapted for human walking model for translation estimation. This will be explained in detail in Chapter 5.

4.3.1 Installation

In order to compile and distribute Android application, a Gradle based built system was used. This is currently a recommended way to maintain Android-based projects [**?**]. At present this application supports the devices with Android 4.0 and above only. In order to compile this project it is recommended to install Android Studio, which already contains the required SDK and also has a built-in Gradle support. More information about configuration, compilation and installation steps can be found in README.md in the main catalog of the attached source code.

4.3.2 Project Structure

Source codes of applications can be found on an attached CD in Folder "Android - Sensor Enhanced Images Camera". In addition, some of the sample datasets were added to the "DataSets" folder in order to allow reader perform evaluations similar to the ones described in the next chapters. The most two important classes are:

1. **CameraSurface.java** - responsible for camera preview setup
2. **CameraActivity.java** - responsible for User Interface creation, Sensor Fusion data access and capturing pictures with these data. It also includes translation estimation code, which is not a part of Android API and has to be done programatically.

4.3.3 User Interface

As mentioned earlier the program allows to track all necessary external camera parameters like rotation angles and relative position in all axes in real-time. In figure 4.1 it can be seen how the User Interface looks like. In order to capture a photo, a user only has to click on the screen center. No additional configuration is required. Reset button can be use to reset translation estimations, when errors are becoming to big. To use the acquired datasets in another program, the smartphone has to be connected to the

Figure 4.1: Android "Sensor Enhanced Image Camera" User Interface overview

computer. For each object capture separate folders are created in the main catalog of an internal SD card.

4.3.4 Rotation calculation

Android SDK already has a built-in API which can be used to access sensor fusion data. In particular, for measuring the rotation, $Sensor.TYPE\_ROTATION\_VECTOR$ was used. It returns 9-degree quaternion, which means that it references rotation to geomagnetic north pole. In order to decompose it to Euler angles a helper methods from Android API were in following code fragment used:

```
import android.hardware.SensorManager;
...
private float[] euclidanAnglesFromQuaternion(float[] quaternion) {
        ...
        //Converts quaternion to 4x4 rotation matrix
        SensorManager.getRotationMatrixFromVector(rotMat, quaternion);
        ...
        //Extracts euclidean Angles in radians (pitch, azimuth, roll)
        SensorManager.getOrientation(rotMat, orientation);
        ...
        return orientation;
}
```

4.3.5 Custom Sensor Data File format

Upon the photo capture the current camera rotation and relative translation are saved in a custom JSON file. A corresponding image is saved along with this JSON file in a folder, which is created with actual time suffix upon every program startup. It allows for taking different dataset captures without them being mixed up.
As mentioned earlier each photo sensor information is stored in a separate file. By default the following information is stored for each captured image:

- **ID**, which indicates order of the photos
- **Path**, which relates the path to a corresponding image file
- **Euler Angles - pitch, roll, azimuth**
- **Relative translation to last taken picture**

All these pieces of information are stored as a list and when the user has completed taking pictures, entire information set is saved into JSON file named sensor.txt. A sample file can be found in the Additional Materials (Listing **??**).

4.4 "Enhanced 3D Reconstructer" - OSX CMake-based project

In order to evaluate algorithms there was a need to prepare a comfortable project environment which would allow for numerical and visual comparison of multiple datasets. CMake was chosen in order to simplify building and compilation process. The entire code architecture was developed and tested on Apple MacbookAir with OSX 10.9 Mavericks.

4.4.1 Installation

In order to compile this project the following elements need to be installed first:

- CMake 2.8
- OpenCV 2.4.10
- Point Cloud Library (PCL) 1.7 [**?**]
- Boost 1.55
- cvsba [**?**]

4.4.2 Project Structure

Source codes of both applications can be found on an attached CD and in the Github repository at the following web address: `https://github.com/KrzysztofWrobel/MasterThesisSource.git`. In addition, some of the sample datasets were added to the dataset folder in order to allow reader perform evaluations similar to the ones described in the next chapters. //TODO TODO TODO

Source codes of applications can be found on an attached CD in Folder Android - Sensor Enhanced Images Camera. In addition, some of the sample datasets were added to the DataSets folder in order to allow reader perform evaluations similar to the ones described in the next chapters. The most two important classes are: 1. CameraSurface.java - responsible for camera preview setup 2. CameraActivity.java - responsible for User Interface creation, Sensor Fusion data access and capturing pictures with these data. It also includes translation estimation code, which is not a part of Android API and has to be done progra- matically.

4.4.3 User Interface

There are two targets defined in CMake file:

1. **Test efficiency**, which was used to evaluate pair reconstruction methods, draw epipolar lines in images and calculate Sampson error distances
2. **Test reconstruction**, which was used to evaluate proposed initialisation pair reconstruction methods with different pose estimation methods

4.4.3.1 Test Efficiency

There is console interface for reconstruction parameters configuration. To perform necessary tests, where was not need to create graphical interface. The following parameters have to be configured to perform efficiency test, which includes Sampson Errors calculation and epipolar lines visualisations in chosen images:

1. **Enhanced Photo Data folder path**
2. **Initial SIFT features set size**

The calculated execution time and Sampson error measurements are printed to the console output.

4.4.3.2 Test reconstruction

In this case also there is only console interface for reconstruction parameters configuration. All necessary parameters have to be configured from the command-line interface. The following variables need to be configured:

1. **Enhanced Photo Data folder path**
2. **Initial SIFT features set size**
3. **Init pair reconstruct method**
    - Standard 8-point algorithm
    - Enhanced 8-point algorithm
    - Standard 5-point algorithm
    - Enhanced 5-point algorithm
    - Alternative 3-point translation estimation
    - None (use existing rotations and translations informations)

4. **Pose estimation method:**
    - Standard OpenCV Pose Estimation
    - Rotation enhanced Pose Estimation
    - Rotation and translation enhanced Pose Estimation
    - None (use existing rotations and translations)
5. **Whether drop outliers or not**
6. **Whether use Bundle Adjustment or not**

The output received by the user is a reconstructed model in a file with *.asc extension. The reconstructed model is also visible in PCL Visualiser integrated into the code. The user can navigate through the model with mouse and 'F' key, which centres the camera view on a selected point.

4.4.4 Rotation matrix generation

To parse Android generated sensor data file Boost::JsonParser was used. As mentioned earlier, Android saves decomposed Euler angles. In order to properly multiply this angles to acquire a proper rotation matrix the following code inspired by MathWorld Wolfram's definition of Euler angles [**?**]. Following listing shows the implementation of equations, that can be found on the pointed website:

```
Mat getRotation3DMatrix(double pitch, double azimuth, double roll) {
    Mat D = (Mat_<T>(3, 3) <<
            cos(roll), -sin(roll), 0,
            sin(roll), cos(roll), 0,
            0, 0, 1);

    Mat C = (Mat_<T>(3, 3) <<
            cos(azimuth), 0, -sin(azimuth),
            0, 1, 0,
            sin(azimuth), 0, cos(azimuth));
    Mat B = (Mat_<T>(3, 3) <<
            1, 0, 0,
            0, cos(pitch), -sin(pitch),
            0, sin(pitch), cos(pitch));
    //Important
    return B * C * D;
}
```

Chapter 5

Reconstruction using only Sensor Fusion data

Initially, the author assumed the usage of the camera rotation and translation estimations from Android Sensor Fusion data only for 3D reconstruction purposes. However, the first attempts to perform reconstruction proved it to be not sufficient and additional rotation error matrix estimations were developed. Unfortunately when it comes to relying on hand-held smartphones, the collected sensor data are very noisy. This chapters explains the concept of how Sensor Fusion can be used in order to obtain camera rotation and translation estimation. Then important implementation aspects behind those estimations are introduced and finally the results of conducted tests of rotation and translation accuracy are presented.

5.1 Concept

The internal camera parameters need to be calculated before the reconstruction process commences. Where accurate rotations and translations of cameras are known, no additional pose calculations are needed. As shown in 2.1.6 knowledge of rotation and translation between calibrated cameras is sufficient to perform triangulation in 3D reconstruction. While retrieving rotation from Android Sensor Fusion is quite easy (Sections 4.3.4 and 4.4.4), it is harder to calculate translation. Android provides access to linear acceleration data, but to get translation it has to be double integrated over time (2.9).

5.1.1 Heuristic for translation estimation

Using the currently available Android sensor data it is difficult to estimate the relative translation of the device. First it has to be done continously, even between taking images. Also linear acceleration measurements, which can be accessed with $Sensor.TYPE\_LINEAR\_ACCELERATION$ constant, are very noisy[**?**]. Such noisy data with double integration over time results in quite big errors after some time. First integration is required to acquire current velocity change and second integration over velocity is required to calculate position change in particular moment. However, it can be improved using human-walking model description with the following heuristic constraints over the calculated velocity:

1. When new linear acceleration sensor data is ready, first apply low pass filtering in order to reduce some high frequency noise. Every new linear acceleration value used for calculation should be weighted average of previous and current acceleration reading.
2. Change sensor data vector from local camera coordinates to global reference system (multiply with inverted rotation matrix). Normally Android returns linear acceleration readings along device axes, using device rotation information they can be changed to readings that are independent from device current orientation.
3. Decide depending on current state, if deviceMovementState has changed:
   (a) If device was previously IDLE and incoming Acceleration value is bigger than $0.5\frac{m}{s^2}$ change device state to MOVING and reset current velocity to $0\frac{m}{s}$
   (b) If device was previously MOVING and incoming Acceleration value is smaller than $0.1\frac{m}{s^2}$ change device state to IDLE
4. Only if device is currently moving:
   (a) Update device velocity by multiplying current acceleration reading by time elapsed from previous reading (2.10).
   (b) Check current speed with walk constraint and eventually scale it down to maximum walking speed value (People walk with average speed of $1.5\frac{m}{s}$)
   (c) Update device position according to equation 2.9.

A code snippet of the described translation estimation heuristics from provided Android application can be found in listing 5.1. Maximum walking speed can be adjusted, but by default it should be set to around $1.5\frac{m}{s}$ [**?**]. The most important factor of the accurate reconstruction is correlation between movements in X,Y and Z axes. Noise in sensors is equally distributed on each axis, so its estimated translation should keep these movements' correlations properly.

5.2 Evaluation

In order to verify the accuracy of camera rotation and translation calculation using Sensor Fusion data additional specific tests were conducted. Few pictures were taken in special test environment with known camera translation between images and thus allowing to evaluate the differences between:

1. rotation calculated using 8-point algorithm from known image points,
2. rotation calculated using 8-point algorithm where to known points few outliers were added to see how exactly this affects rotation estimation.
3. rotation calculated from automatically matched points using standard OpenCV methods
4. rotation calculated directly from Sensor Fusion data.

5.2.1 Test Environment

Pictures were taken from 9 different positions on 2x3m area. Those images can be found on attached CD in "DataSets" folder. Their positions and corresponding names are visualised in Figure 5.1. Grey and brown rectangles on this schematic represent photographed object and blue stars are position, from which images were taken. In figure 5.2 it is shown how this setup environment looked in real life. Red circles indicate positions of visible markers, from which pictures were taken. Marker used to note the capture positions can be seen in Figure 5.3.

5.2.2 Rotation calculation tests

After the pictures were taken many tests were made in order to verify, what differences one can expect in terms of rotation angles estimation. As mentioned four cases were researched:

```
1   public void onSensorChanged(SensorEvent event) {
2           ...
3       } else if (event.sensor == mLinearAcceleration) {
4           ...
5           //Filtering out some noise
6           newGlobalAcceleration = lowPass(newGlobalAcceleration,
                currentGlobalAcceleration);
7           //Switch linear acceleration from phone local coordinates to
                global coordinates
8           Matrix.multiplyMV(newGlobalAcceleration, 0,
                invertedRotationMatrix.clone(), 0, newGlobalAcceleration,
                0);
9
10          double distance = getLength(currentGlobalAcceleration);
11          long currentTimeMillis = System.currentTimeMillis();
12          //Decide state of the device. Distance is in m/s^2
13          if (distance > 0.5 && deviceState == State.IDLE) {
14              if (currentTimeMillis - movingEndTime > 300) {
15                  deviceState = State.MOVING;
16                  currentGlobalVelocity = {0,0,0};
17              }
18          } else if (distance < 0.1 && deviceState == State.MOVING) {
19              if (currentTimeMillis - movingStartTime > 350) {
20                  deviceState = State.IDLE;
21              }
22          }
23
24          if (deviceState == State.MOVING) {
25              //Update current device velocity
26              currentGlobalVelocity += currentGlobalAcceleration * dT;
27
28              double velocity = getLength(currentGlobalVelocity);
29              //Check and adjust current velocity. People walk with
                    average speed 1.5\frac{m}{s}
30              if (velocity > WALKING_MAX_VELOCITY) {
31                  currentGlobalVelocity /= velocity /
                        WALKING_MAX_VELOCITY;
32              }
33
34              //Update device relative position, s = V0 * t + a * t
                    ^2/2;
35              currentRelativePosition += currentGlobalVelocity * dT +
                    currentGlobalAcceleration * dT * dT / 2;
36          }
37          ...
38  }
```

Listing 5.1: Snippet from Android source code position estimation heuristic

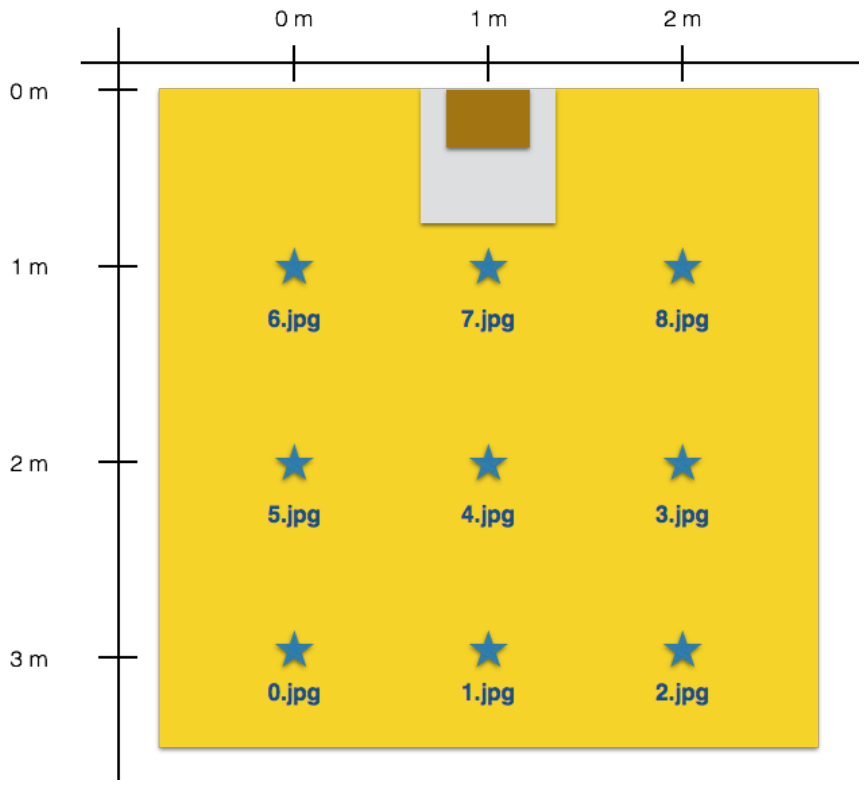

Figure 5.1: Schematic of test environment for capturing images and Sensor Fusion data simultaneously. White and brown rectangles represent photographed object, blue stars with corresponding file name label positions from which test images were taken.

1. Selecting proper corresponding points by user from image
2. Selecting proper corresponding points and adding few outliers by user
3. Automatic correspondence matching with Sift descriptors and BruteForce matching from OpenCV
4. Rotation angles calculation directly from Sensor Fusion data

In 4th case it was not necessary to calculate fundamental matrix and later decompose it to rotation angles. In others first fundamental matrix was calculated using standard 8-point algorithm implementation. In combination with internal camera matrix it allowed to calculate essential matrix. **E** was later was decomposed in the already described way to acquire proper rotation matrix. In figures 5.4 - 5.5 and 5.7 - 5.8 it can be seen, which points were selected by the author for two example image pairs to perform camera rotation estimation tests. In both cases red line means proper correspondence

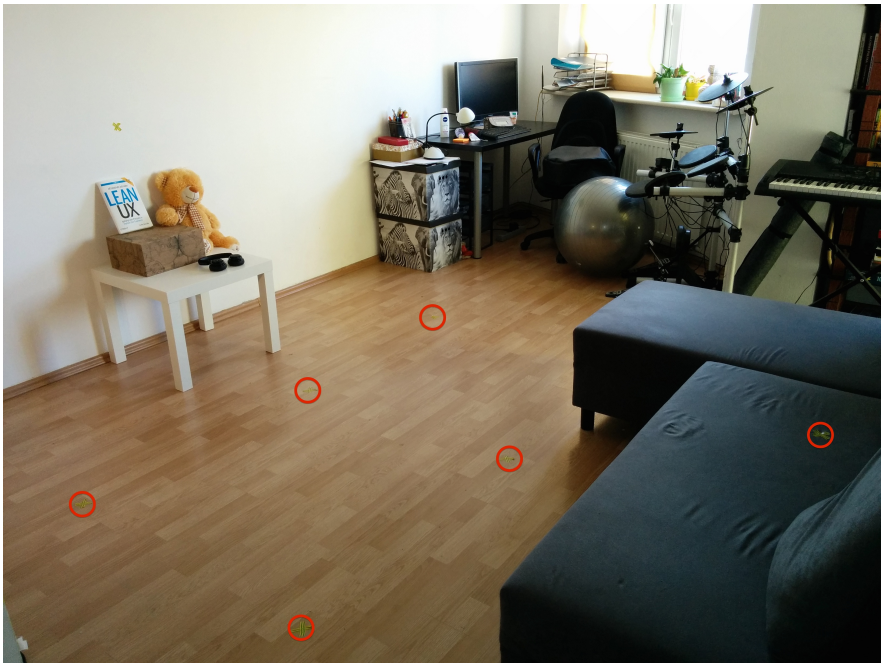

Figure 5.2: Image of testing environment for capturing images along Sensor Fusion data. Red circles indicates positions of visible markers.

Figure 5.3: Image of marker used to indicate position for test capturing.

and yellow means improper match. In figures 5.6 and 5.9 reader can see, what types of results usually automatic matching produces. These results are quite good, but unfortunately have some outliers.

For all mentioned in 5.2.2 situations corresponding F and E matrices were found and then they were decomposed to rotation matrices and finally transformed to Euler angles. In the following figures 5.10 and 5.11 these calculations were gathered.

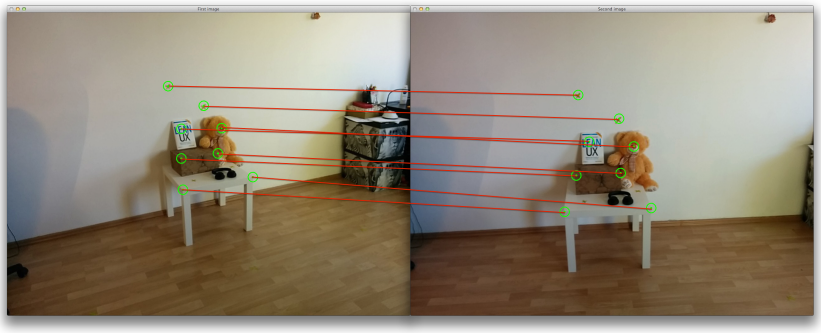

Figure 5.4: Points matches selected by the author (Images 0.jpg and 1.jpg).

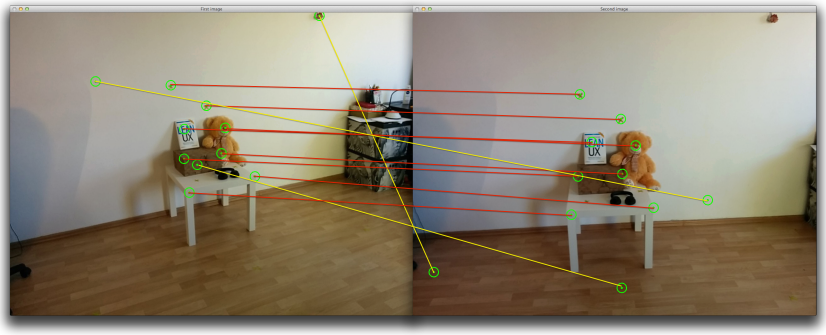

Figure 5.5: Points matches selected by the author with additional outliers (Images 0.jpg and 1.jpg).

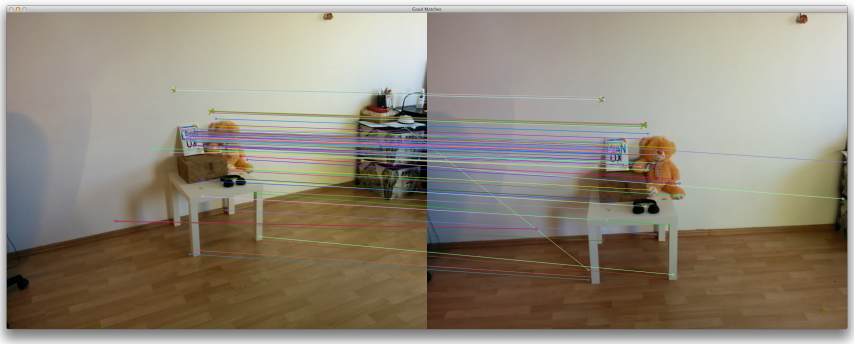

Figure 5.6: Automatically matched points with usage of Sift descriptors and BruteForce matcher (Images 0.jpg and 1.jpg).

In first row "known points" euler angles for proper user matching can be found. They also become the reference value for accuracy comparison for other cases.
What's important to notice that existence of only 3 outliers significantly influences the

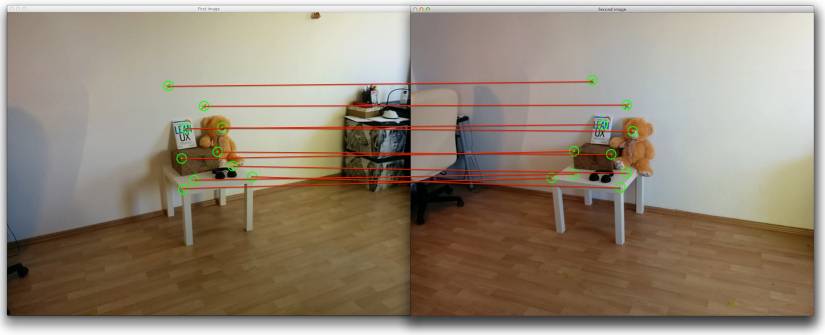

Figure 5.7: Points matches selected by the author (Images 0.jpg and 2.jpg).

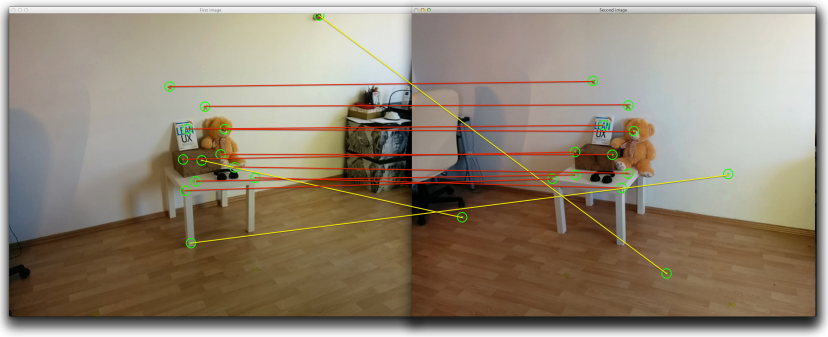

Figure 5.8: Points matches selected by the author with additional outliers (Images 0.jpg and 2.jpg).

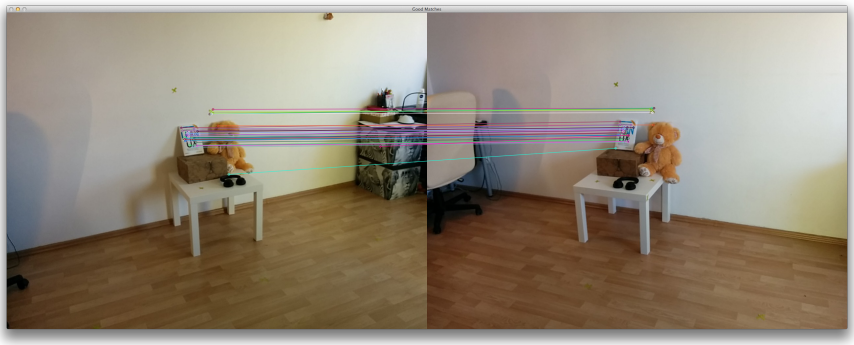

Figure 5.9: Automatically matched points with usage of Sift descriptors and BruteForce matcher (Images 0.jpg and 1.jpg).

camera rotation estimation. Using data from sensors also doesn't give close enough approximation of angles, that would allow for building reconstruction algorithms relaying solely on them.

But of course to reconstruction camera translation information is needed. Next section will present effects of Sensor Fusion based approach to translation calculation.

	Pitch	Azimuth	Roll	Pitch Difference	Azimuth Difference	Roll Difference	Differences sum
Known points	3,98	-12,96	-1,99	0,00	0,00	0,00	
Known point + Outliers	10,20	-9,64	-5,02	6,22	3,32	3,02	**12,56**
Sensors	0,55	-7,20	0,93	3,43	5,76	2,92	**12,11**
Auto matching	0,90	-8,85	-7,10	3,08	12,83	11,07	**26,98**

Figure 5.10: Rotation tests comparison in Sensor only based approach for 0.jpg and 1.jpg images. In first raw reference angles were calculated for proper point matching. Next raw shows how outliers influence those measurements. 3rd shows calculation using Sensor Fusion approach and 4th one measurements from automatic OpenCV supported matching. All angle differences are referenced to 1st row.

	Pitch	Azimuth	Roll	Pitch Difference	Azimuth Difference	Roll Difference	Differences sum
Known points	-5,98	-37,78	-13,30	0,00	0,00	0,00	
Known point + Outliers	-51,03	-44,27	1,68	45,05	6,49	14,99	**66,52**
Sensors	0,28	-29,41	-0,07	6,26	8,36	13,24	**27,86**
Auto matching	-1,38	-16,54	-12,13	4,60	10,56	6,15	**21,32**

Figure 5.11: Rotation tests comparison in Sensor only based approach for 0.jpg and 2.jpg images. In first raw reference angles were calculated for proper point matching. Next raw shows how outliers influence those measurements. 3rd shows calculation using Sensor Fusion approach and 4th one measurements from automatic OpenCV supported matching. All angle differences are referenced to 1st row.

5.2.3 Translation calculation tests

Implemented heuristic for movement tend to be more predictable than solely relaying on Sensor Fusion Linear Acceleration usage as described in [**?**]. Tests for this heuristic were conducted on 4 different walking distances on flat earth surface: 1m, 2m, 5m and 10m. In figure 5.12 results are presented. In all cases the distance estimated was very different than real one. For small distances it seems that proposed heuristic tends to be not sensitive enough having big difference between real and estimated distance traveled. Things are getting a little bit better for longer distances giving values that are much closer to real ones. However in some cases due to unbalanced up-down movement during walking "Pos Y", which is position in an axis perpendicular to earth surface, big distances are visible (Tests were conducted on a flat surface and it shows how big

Distance 1m				Distance 2m			
Pos X	Pos Y	Pos Z	Distance	Pos X	Pos Y	Pos Z	Distance
-0,13	-0,07	0,20	0,25	0,74	0,21	0,84	1,14
-0,57	0,23	-0,30	0,68	0,65	0,02	0,23	0,69
0,18	0,26	0,19	0,37	0,54	0,28	0,21	0,64
0,23	0,01	0,18	0,29	0,42	0,96	0,76	1,29
0,53	0,03	0,02	0,53	1,31	0,21	1,82	2,25
		Avarage	0,42			Avarage	1,20
Distance 5m				Distance 10m			
Pos X	Pos Y	Pos Z	Distance	Pos X	Pos Y	Pos Z	Distance
2,08	0,49	1,66	2,71	8,47	1,05	0,14	8,54
1,25	0,97	1,43	2,13	7,99	0,47	2,93	8,52
2,78	0,91	1,61	3,34	5,93	2,44	1,57	6,60
2,18	0,79	0,56	2,39	4,42	0,39	5,25	6,87
0,27	0,01	1,28	1,31	4,69	2,72	2,36	5,91
		Avarage	2,37			Avarage	7,29

Figure 5.12: Screenshot from table of results conducted on Sensor Fusion data based translation estimation during walking with smartphone. In red are indicated values, which were highly unexpected, especially in Y axis, which is perpendicular to earth surface

double integration of acceleration over time errors can be in reality).
The only thing that can be said, that in a case of unpredictable movement like walking it's hard to relay on translation estimation with usage of Sensor Fusion estimation.

5.2.4 Epipolar line calculation

To get reader acquainted, how small errors in either rotation or translation influence calculation and visualisation of epipolar lines for all of test cases from 5.10 and 5.11 adequate calculations were made. In figures 5.13 - 5.20 reader can see, how big those differences can be depending on method that was used. Methods that are image based, which means that **F** was calculated directly from points correspondeces give much accurate epipolars. In case of 5.16 and 5.20 were **F** was calculated from sensor fusion estimated **R** and **T** completely miss automatically matched point correspondeces. Evaluation of time efficiency and numerical accuracy are conducted in Chapter 8.3.3.
At this point of his research, author had an idea to try slightly different approach, which could produce quite good effects - usage of Sensor Fusioned rotation estimation and calculating missing translation information from image itself.

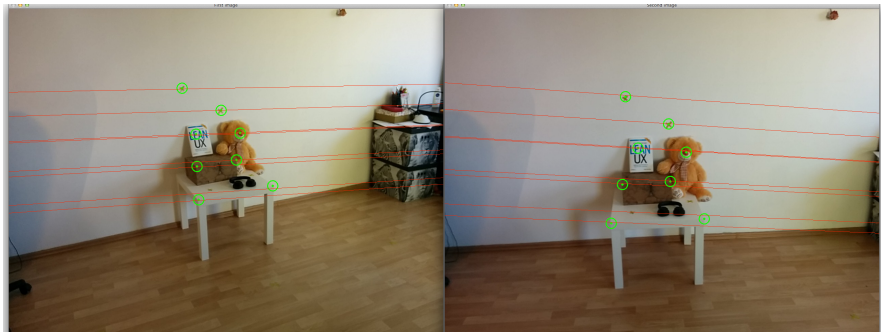

Figure 5.13: Visualisation of epipolar lines for 8-point Fundamental matrix estimation from manually selected points for images 0.jpg and 1.jpg in test dataset. No outliers are produced.

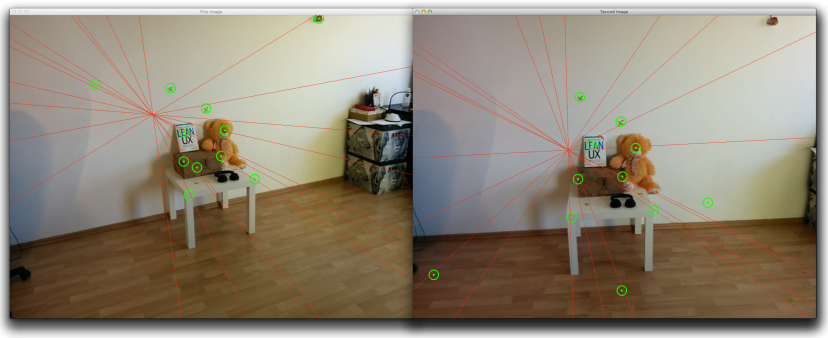

Figure 5.14: Visualisation of epipolar lines for 8-point Fundamental matrix estimation from manually selected points for images 0.jpg and 1.jpg in test dataset. Some outliers are produced and indicated by yellow lines.

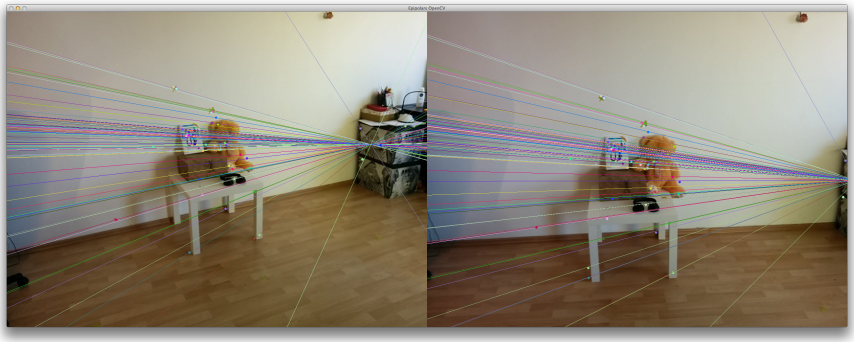

Figure 5.15: Visualisation of epipolar lines for 8-point Fundamental matrix estimation from automatically matched points for images 0.jpg and 1.jpg in test dataset. Some outliers are produced during feature matching process.

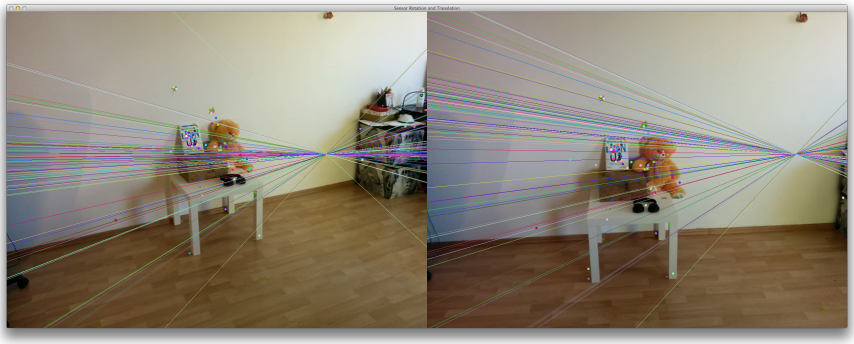

Figure 5.16: Visualisation of epipolar lines calculated from rotation and translation estimated from Sensor Fusion readings and automatically matched correspondeces for images 0.jpg and 1.jpg from test dataset. Some outliers are produced during feature matching process, but they do not disturb calculation for proper matches.

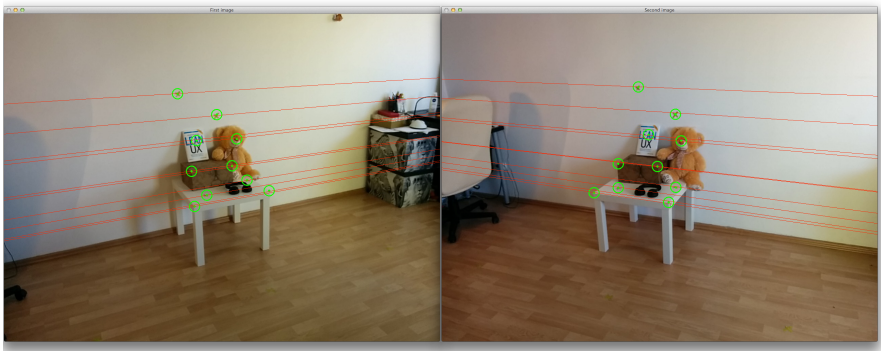

Figure 5.17: Visualisation of epipolar lines for 8-point Fundamental matrix estimation from manually selected points for images 0.jpg and 2.jpg in test dataset. No outliers are produced.

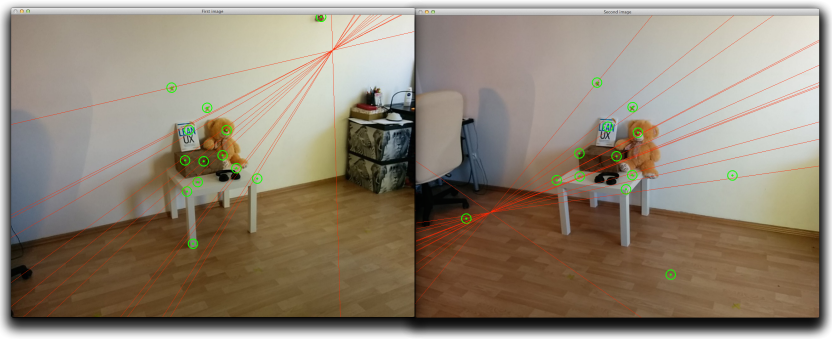

Figure 5.18: Visualisation of epipolar lines for 8-point Fundamental matrix estimation from manually selected points for images 0.jpg and 2.jpg in test dataset. Some outliers are produced and indicated by yellow lines.

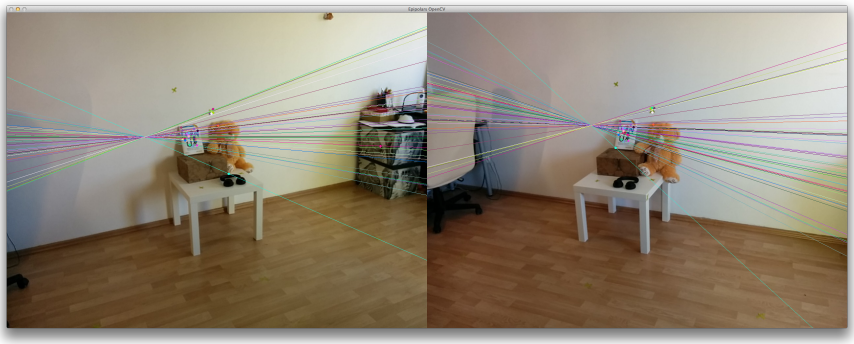

Figure 5.19: Visualisation of epipolar lines for 8-point Fundamental matrix estimation from automatically matched points for images 0.jpg and 2.jpg in test dataset. Some outliers are produced during feature matching process.

Figure 5.20: Visualisation of epipolar lines calculated from rotation and translation estimated from Sensor Fusion readings and automatically matched correspondeces for images 0.jpg and 2.jpg from test dataset. Some outliers are produced during feature matching process, but they do not disturb calculation for proper matches.

Chapter 6

Reconstruction using Sensor Fusion rotation and 3-point translation

As previously shown using sensors for camera translation estimation give quite big errors, resulting in especially poor epipolar line matching. This especially prevents one from getting highly accurate results in dense matching (which strongly relay on epipolar line calculation) and from getting highly detailed 3D models.
Because of that author in his research had an idea to combine traditional image based methods for translation estimation with Sensor Fusion approach for camera rotation. Following chapter describes his deliberations on that approach as well as evaluation performed for verification.

6.1 Concept

From section 9.2 of Multiple View Geometry [**?**] it is known that

$$\mathbf{x}_l^T \mathbf{F} \mathbf{x} = 0 \tag{6.1}$$

and also that for relative camera based system ($P = \left[I|0\right], P' = \left[R|t\right]$) fundamental matrix can be written as:

$$\mathbf{F} = \mathbf{K}^{-T} \left[\mathbf{T}\right]_x \mathbf{R} \mathbf{K}^{-1} \tag{6.2}$$

where

$$\left[\mathbf{T}\right]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \text{where } \mathbf{T} = \left[t_x, t_y, t_z\right] \tag{6.3}$$

By combing two above equations, it can be rewritten as:

$$\mathbf{x}_{\prime}^T \mathbf{K}^{-T} \left[\mathbf{T}\right]_x \mathbf{R}\mathbf{K}^{-1}\mathbf{x} = 0 \tag{6.4}$$

And finally as:

$$\mathbf{x}_{\prime}^T \mathbf{K}^{-T} \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \mathbf{R}\mathbf{K}^{-1}\mathbf{x} = 0 \tag{6.5}$$

Having:

$$\begin{array}{rcl} \mathbf{h}_{\prime}^T & = & \mathbf{x}_{\prime}^T \mathbf{K}^{-T} \\ \mathbf{h} & = & \mathbf{R}\mathbf{K}^{-1}\mathbf{x} \end{array} \tag{6.6}$$

and substituting those values in 6.8:

$$\begin{bmatrix} h_{\prime 1} & h_{\prime 2} & h_{\prime 3} \end{bmatrix} \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0 \tag{6.7}$$

When multiplying it one receives the following:

$$h_1 \cdot h_{\prime 2} \cdot t_z - h_1 \cdot h_{\prime 3} \cdot t_y - h_2 \cdot h_{\prime 1} \cdot t_z + h_2 \cdot h_{\prime 3} \cdot t_x + h_3 \cdot h_{\prime 1} \cdot t_y - h_3 \cdot h_{\prime 2} \cdot t_x = 0 \tag{6.8}$$

which can be grouped:

$$t_x \cdot (h_2 \cdot h_{\prime 3} - h_3 \cdot h_{\prime 2}) + t_y \cdot (h_3 \cdot h_{\prime 1} - h_1 \cdot h_{\prime 3}) + t_z \cdot (h_1 \cdot h_{\prime 2} - h_2 \cdot h_{\prime 1}) = 0 \tag{6.9}$$

and rewritten as:

$$\begin{bmatrix} t_x & t_y & t_z \end{bmatrix} \begin{bmatrix} (h_2 \cdot h_{\prime 3} - h_3 \cdot h_{\prime 2}) \\ (h_3 \cdot h_{\prime 1} - h_1 \cdot h_{\prime 3}) \\ (h_1 \cdot h_{\prime 2} - h_2 \cdot h_{\prime 1}) \end{bmatrix} = 0 \tag{6.10}$$

then solved for instance with SVD with only three points pairs, because the equation has only 3 unknown variables (translation in three axes). However, in such situation the overall accuracy strictly depends on the precise measurements of camera rotation, because value of **h** depends on it. Thus every error in rotation estimation will influence also translation estimation using proposed 3-point method.

6.2 Implementation

When it is known or assumed that the acquired rotation data is accurate, the translation can be calculated with the 3-point algorithm. The proposed equation 6.10 was implemented in a way similar to other fundamental matrix estimations methods implemented in OpenCV. This particular implementation also has the ability to properly filter outliers with the use of RANSAC algorithm and its listing, written in self-documenting style, can be seen below:

```
...
for (iterationNumber = 0; iterationNumber < niters; iterationNumber
    ++) {

    getSubsety(prev_points_raw, next_points_raw, point1s, point2s,
        300, modelPoints);
    Mat t = findTranslation(point1s, point2s, rotDiffGlobal, Kinv);
    Mat F1 = constructFundamentalMatrix(rotDiffGlobal, t, Kinv);

    int goodCount = findInliersy(prev_points_raw, next_points_raw,
        F1, errors, statuses, reprojThreshold);
    if (goodCount > maxGoodCount) {
        swap(statuses, goodStatuses);
        FEnhanced = F1;
        tEnhanced = t / t.at<double>(2);
        maxGoodCount = goodCount;
        niters = cvRANSACUpdateNumIters(confidence,
                (double) (count - maxGoodCount) / count, modelPoints
                    , niters);
    }

}
...

Mat findTranslation(std::vector<cv::Point2d> &points1, std::vector<
    cv::Point2d> &points2, Mat &rotDiff, Mat &Kinv) {

Mat hg1 = Mat::zeros(points1.size(), 3, CV_64FC1);
Mat hg2 = Mat::zeros(points2.size(), 3, CV_64FC1);
for (int i = 0; i < points1.size(); i++) {
    hg1.at<double>(i, 0) = points1[i].x;
    hg1.at<double>(i, 1) = points1[i].y;
    hg1.at<double>(i, 2) = 1;
    hg2.at<double>(i, 0) = points2[i].x;
```

```
        hg2.at<double>(i, 1) = points2[i].y;
        hg2.at<double>(i, 2) = 1;
    }

    hg1 = hg1 \cdot  (rotDiff \cdot  Kinv).t();
    hg2 = hg2 \cdot  (Kinv).t();

    Mat A = Mat::zeros(hg1.rows, 3, CV_64FC1);
    for (int i = 0; i < hg1.rows; i++) {
        A.at<double>(i, 0) = (hg2.at<double>(i, 2) \cdot  hg1.at<double
            >(i, 1) - hg2.at<double>(i, 1) \cdot  hg1.at<double>(i, 2));
        A.at<double>(i, 1) = (hg2.at<double>(i, 0) \cdot  hg1.at<double
            >(i, 2) - hg2.at<double>(i, 2) \cdot  hg1.at<double>(i, 0));
        A.at<double>(i, 2) = (hg2.at<double>(i, 1) \cdot  hg1.at<double
            >(i, 0) - hg2.at<double>(i, 0) \cdot  hg1.at<double>(i, 1));
    }

    SVD svd1(A);
    Mat tCalc = svd1.vt.row(2);

    //Translation between cameras estimated and Fundamental Matrix from
        that as well
    Mat T = (tCalc.t());

    return T;
```

6.3 Evaluation

In case of this approach previously conducted translation accuracy tests do not have much comparison. Currently calculated translation doesn't have realistic global scale, because there isn't anything it could be referenced to. The only thing that can be evaluated is whether epipolar line calculations are more accurate than in Sensor Fusion based approach.

In figures 6.1 and 6.2 are presented the epipolar lines calculated from Fundamental matrices constructed from translation estimated using proposed 3-point approach and rotation from sensors. It's good to compare them with epipolar lines visualised in figures 5.16 and 5.20. They are much more accurate keeping the condition of crossing the corresponding matches. Evaluation of time efficiency and numerical accuracy are conducted in Chapter 8.3.3.

At this point of his research, author had an idea to try another approach, which could produce quite even better effects - usage of Sensor Fusion rotation data in standard 5-point and 8-point algorithms for **F** calculation. This would allow on both image based translation estimation and possibly correction of sensor influenced rotation error matrix estimation.

Figure 6.1: Visualisation of epipolar lines for 3-point translation estimation and Sensor Fusion rotation approach for images 0.jpg and 1.jpg in test dataset. It can be seen that in comparison to 5.16 epipolar lines cross or at least are close to corresponding points.

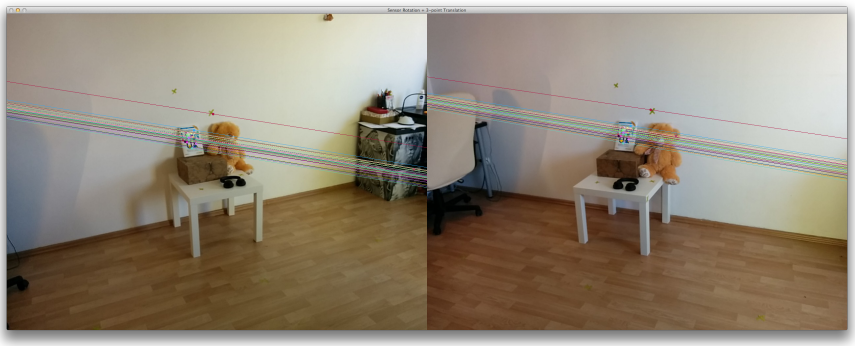

Figure 6.2: Visualisation of epipolar lines for 3-point translation estimation and Sensor Fusion rotation approach for images 0.jpg and 2.jpg in test dataset. It can be seen that in comparison to 5.20 epipolar lines cross or at least are close to corresponding points.

Chapter 7

Enhanced 8-point and 5-point reconstruction

7.1 Concept

As indicated in a similar research (Chapter 3), it is very attractive to use additional data to enhance reconstruction and reduce ambiguity in finding correct solution for 3D reconstruction. Additional camera or model information help to implement faster, more stable and robust algorithms. This thesis will show how the prior knowledge of rotation or translation acquired via mobile sensor fusion can be used to enhance process of 3D reconstruction from a series of images. When it comes to relying on hand-held smartphones, the collected sensor data are very noisy. This thesis shows how even noisy information can be used in the reconstruction processes. Initially, only the camera rotation estimation was supposed to be used within the scope of this thesis. However, the first attempts to perform reconstruction proved it to be not sufficient and additional rotation error matrix estimations were developed. It follows from the analysis of the theory and related works that both epipolar equations and pose estimation techniques can be improved by additional rotation and translation information data. Author of this thesis proposes the use of an environment, where a user can decide what type of strategy to use.

7.1.1 Requirements

The proposed methodology needs an input in the form of a series of images with additional information about the position of the camera - the euclidean rotation and, optionally, translation. The use of smartphone is not necessary; any camera with sensor-fusioned accelerometer and gyroscope (magnetometer and the use thereof is optional, as discussed in [**?**], has both advantages and disadvantages), capable of performing the rotation and translation estimation, can be used.The internal camera parameters need to be calculated before the reconstruction process commences. Additional sensor data need not be fully accurate. Noisy external camera parameters can still be successfully used for enhancing the reconstruction process.

7.1.2 Enhancing epipolar geometry equations with initial rotation matrix

The initial pair reconstruction step is of utmost importance and needs to be accurate in order to let the other images calculate relative position based on the initially reconstructed 3D cloud points.

As already discussed, rotation can be distorted with noise. This can be written down as:

$$R = R_{error} * R_{init} \tag{7.1}$$

where R_{init} is initial rotation matrix constructed from the measured angles and R_{error} is rotation error matrix. Looking at this from a different point of view equation 7.1 one can interpret it as the multiplication of the two rotation matrices: one estimated initial rotation matrix (but close to local optimum) and the second one responsible for correction of noise error. Instead of on the relative rotation matrix calculation, the primary idea of the algorithm proposed in this thesis is based on the entire rotation matrix calculation, which can be acquired from the essential matrix SVD decomposition, where only the rotation R_{error} can be estimated. Eventually, equation 6.4 can be rewritten as:

$$x_r^T * K^{-T} * \left[T\right]_x * R_{error} * R_{init} * K^{-1} * x = 0 \tag{7.2}$$

Having:

$$\begin{aligned} h_r^T &= x_r^T * K^{-T} \\ h &= R_{init} * K^{-1} * x \\ G &= \left[T\right]_x * R_{error} \end{aligned} \tag{7.3}$$

With such notation one can notice that:

$$h\prime^T * G * h = 0 \tag{7.4}$$

which resembles the already known fundamental (Equation 2.2) and essential equations (2.3). Naturally, $h\prime$ and h both are expressed in homogenous coordinates. Tt follows from the analysis that G has 6DOF: 3 due to an unknown translation and another 3 due to an unknown correction angles (created by rotation error matrix decomposition). Theoretically, such matrix can be resolved for instance by both 5 and 8-point algorithms. Therefore the standard fundamental and essential equation solvers can be used in order to retrieve both $\left[T\right]_x$ and R_{error}. The finally estimated R_{error} and calculated R_{init} has to be multiplied in order for the new rotation estimation of R (Equation 7.1) to be retrieved. Pursuant to Appendix 6 section regarding Parametrization of 3D rotations of "Multiple View Geometry in Computer Vision"(A6.9.1 (iii) [**?**]), the use of Rodrigues parametrisation for small angle (and noise in initial rotation matrices estimations can be expressed by small angles) the rotation matrix can be expressed as:

$$R_{error} \cong I + \left[w\right]_x \tag{7.5}$$

and thus in our case R_{error}, equals approximately to:

$$R_{error} \cong \begin{bmatrix} 1 & -w_z & w_y \\ w_z & 1 & -w_x \\ -w_y & w_x & 1 \end{bmatrix} \tag{7.6}$$

Such a criterion with a special matrix design can be used when decomposing G to resolve ambiguity in choosing the proper solution. The standard four solution ambiguity with two possible rotations and translations can be reduced to two possible translation calculations. The concept described constitutes a basis of the implemented, enhanced 8-point and 5-point algorithms.

7.2 Implementation

This chapter describes the chosen implementation environment. It also describes an Android application which was created for the purpose of acquiring image sequences with additional sensor data. The structure of both Android and desktop projects is explained and essential implementation details of the proposed algorithms and strategies are discussed herein.

Most of the necessary algorithms were already implemented in OpenCV. In addition an open-source project with the 5-point algorithm implementation was used. Its source code can be found in [**?**].

7.2.0.1 Enhancing epipolar equations

As could have been observed in 7.1.2, in order to enhance initial pair reconstruction, the algorithms same as the standard ones can be used, but with specially conditioned points matches. The standard 8-point algorithm is already available in OpenCV library. The implemented enhanced 8-point version can be found in 'Multiview.cpp' file; its main difference is that it uses sets of specially modified input points, which are conditioned and the first one is additionally rotated with Initial Rotation Matrix. In OpenCV it can be done with:

```
    ...
    undistortPoints(points1Exp, points1Exp, K, distCoeffs, rotDiffGlobal
        );
    undistortPoints(points2Exp, points2Exp, K, distCoeffs);
    ...
```

where rotDiffGlobal is the relative rotation matrix between two cameras. The estimated fundamental matrix needs to be transformed to essential matrix for further decomposition. In order to choose proper matrix decomposition the following code is used:

```
void chooseProperMatrixFromEnhanced(Mat &dRx, Mat &dR1x, Mat &TdRExp,
    Mat &dR, Mat &T) {
    dR = dRx;
    if (decideProperMatrix(dRx, 0.05)) {
        dR = constraintMatrix(dRx);
    } else if (decideProperMatrix(dR1x, 0.05)) {
        dR = constraintMatrix(dR1x);
    } else if (decideProperMatrix(-dRx, 0.05)) {
        dR = constraintMatrix(-dRx);
    } else if (decideProperMatrix(-dR1x, 0.05)) {
        dR = constraintMatrix(-dR1x);
    }

    Mat skewT = TdRExp * dR.inv();
    cout << "skewT" << skewT << endl;
```

```
    Mat tdecx = Mat(3,1, CV_64FC1);
    tdecx.at<double>(0) = (skewT.at<double>(2,1) - skewT.at<double>(1,2)
        )/2;
    tdecx.at<double>(1) = (skewT.at<double>(0,2) - skewT.at<double>(2,0)
        )/2;
    tdecx.at<double>(2) = (skewT.at<double>(1,0) - skewT.at<double>(0,1)
        )/2;
    T = tdecx;

}

bool decideProperMatrix(Mat dRot, double tolerance){
    double a00 = abs(dRot.at<double>(0,0) - 1);
    double a11 = abs(dRot.at<double>(1,1) - 1);
    double a22 = abs(dRot.at<double>(2,2) - 1);
    if((a00 + a11 + a22)/3< tolerance) {
        return true;
    }else {
        return false;
    }
}
```

These lines help to decide on a properly constrained rotation error matrix equation 7.6 and calculate relative translation between cameras, but generally speaking this codes decides, which of the two standard rotation matrices has all of diagonal elements equal or close to ones.

7.3 Evaluation

In the following section reader would be able to see how proposed enhancements influence camera rotation and epipolar line calculation. There was no reasonable way to comprehend translation accuracy, because this is something, which is relative and depending on a scale, which in normal situation cannot be determined without additional knowledge about photographed environment.

7.3.1 Rotation calculation tests

In figures 7.1 and 7.2 rotation matrix decomposition to euler angles results are presented. However this time results of 5-point method effects as well as 8-point and

5-point enhanced versions are presented. All of them were calculated from automatically matched correspondences, which included as usually some of the outliers. In blue are marked reference values. Most accurate estimations were presented in green and worst ones were presented in red. It's hard time to decide, which of 8-point and 5-point method is better, however in all cases 8-point enhanced version gives slightly more accurate results than standard one version.

	Pitch	Azimuth	Roll	Pitch Difference	Azimuth Difference	Roll Difference	Differences sum
Known points	3,98	-12,96	-1,99	0,00	0,00	0,00	
Known point + Outliers	10,20	-9,64	-5,02	6,22	3,32	3,02	**12,56**
Sensors	0,55	-7,20	0,93	3,43	5,76	2,92	**12,11**
8-point	0,90	-8,85	-7,10	3,08	12,83	11,07	**26,98**
8-point enhanced	0,58	-12,60	-6,94	3,40	0,36	4,95	**8,71**
5-point	2,42	-1,24	-1,97	1,56	11,73	0,03	**13,31**
5-point enhanced	-0,63	-23,40	-4,68	4,61	10,44	2,68	**17,73**

Figure 7.1: Rotation tests comparison for all proposed approaches for 0.jpg and 1.jpg images. In first raw reference angles were calculated for proper point matching. Next raw shows how outliers influence those measurements. 3rd shows calculation using Sensor Fusion approach and 4th one measurements from automatic OpenCV supported matching. 5th row presents angles calculated for 8-point enhanced version. Next two rows presents 5-point algorithm both in standard and enhanced version. All angle differences are referenced to the 1st row. In green the most accurate result is presented. In red the worst one.

7.3.2 Epipolar line calculation

In figures 7.3 - 7.6 epipolar lines calculation effects for both standard 8-point and 5-point algorithms as well as their proposed versions were presented. Very interesting that 8-point enhanced version slightly straightens epipolar lines in comparison to standard one. These are more similar to perfect matching cases of 5.13 and 5.17. In case of 5-point algorithm for 1st tested pair (7.4), enhanced version produced much more readable results both in epipolar lines and euler angles calculation. There was not enough time, but it would be good to research, why in case of 5-point algorithm results are so inconsistent (one time better, other worse). In case of enhanced 8-point algorithm, even in feature sets with many outliers it gives very promising effects. Next chapter

	Pitch	Azimuth	Roll	Pitch Difference	Azimuth Difference	Roll Difference	Differences sum
Known points	-5,98	-37,78	-13,30	0,00	0,00	0,00	
Known point + Outliers	-51,03	-44,27	1,68	45,05	6,49	14,99	**66,52**
Sensors	0,28	-29,41	-0,07	6,26	8,36	13,24	**27,86**
8-point	-1,38	-16,54	-12,13	4,60	10,56	6,15	**21,32**
8-point enhanced	-1,06	-30,05	-13,27	4,92	7,73	0,03	**12,67**
5-point	3,01	-1,30	-70,69	8,99	36,48	57,39	**102,85**
5-point enhanced	-1,38	-2,47	-9,95	4,60	35,31	3,36	**43,27**

Figure 7.2: Rotation tests comparison for all proposed approaches for 0.jpg and 2.jpg images. In first raw reference angles were calculated for proper point matching. Next raw shows how outliers influence those measurements. 3rd shows calculation using Sensor Fusion approach and 4th one measurements from automatic OpenCV supported matching. 5th row presents angles calculated for 8-point enhanced version. Next two rows presents 5-point algorithm both in standard and enhanced version. All angle differences are referenced to the 1st row. In green the most accurate result is presented. In red the worst one.

will describe validation of authors experiments with usage of Sensor Fusion on whole Structure From Motion process.

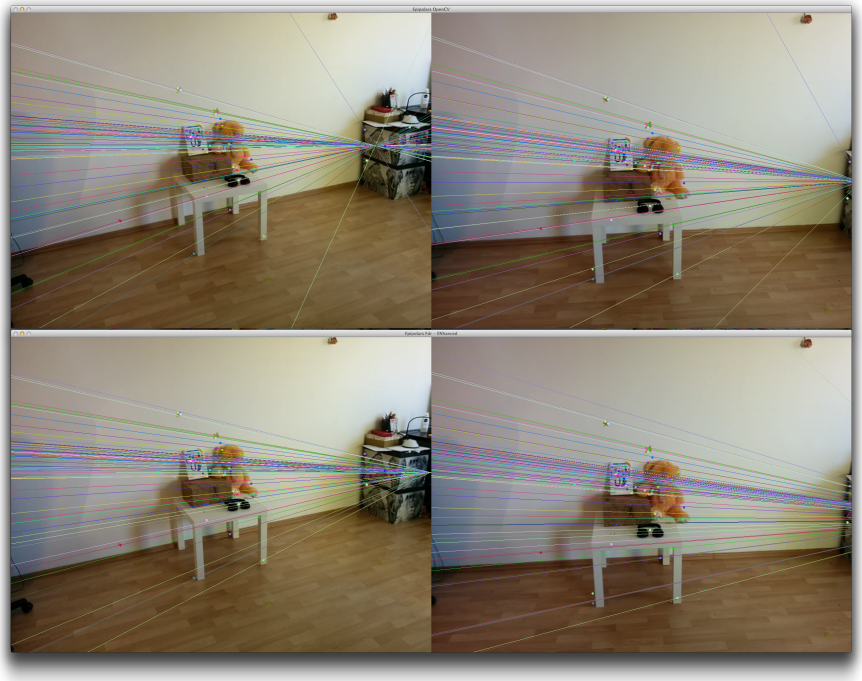

Figure 7.3: Comparison of epipolar lines for 8-point standard and enhanced versions for automatically matched points in images 0.jpg and 1.jpg from test dataset. Some outliers are produced during feature matching process.

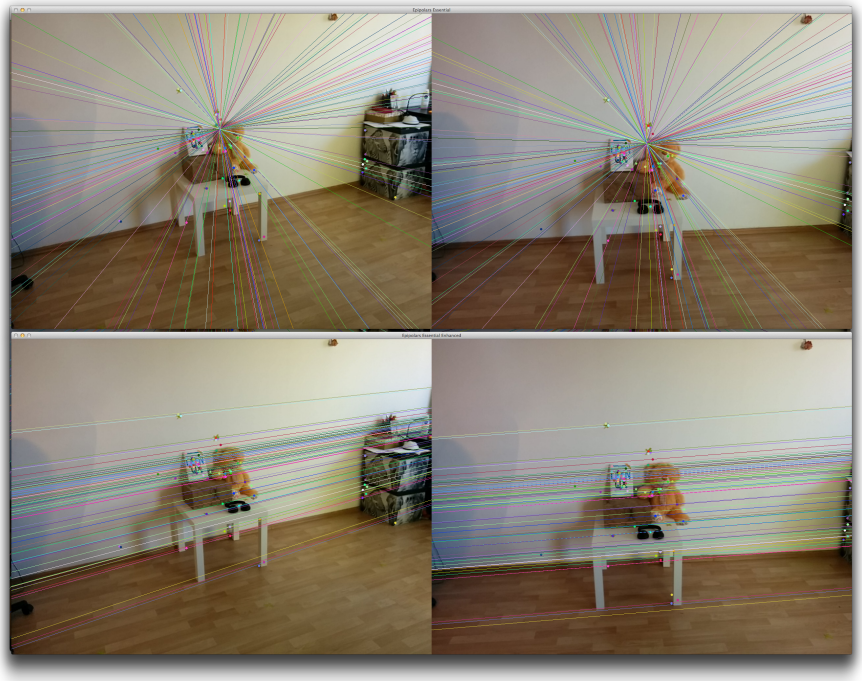

Figure 7.4: Comparison of epipolar lines for 5-point standard and enhanced versions for automatically matched points in images 0.jpg and 1.jpg from test dataset. Some outliers are produced during feature matching process.

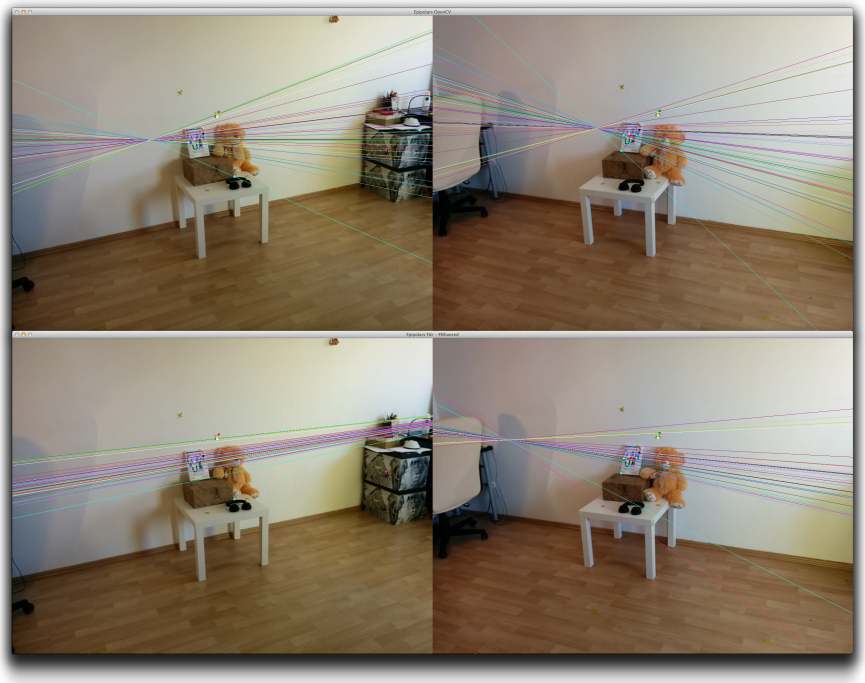

Figure 7.5: Comparison of epipolar lines for 8-point standard and enhanced versions for automatically matched points in images 0.jpg and 2.jpg from test dataset. Some outliers are produced during feature matching process.

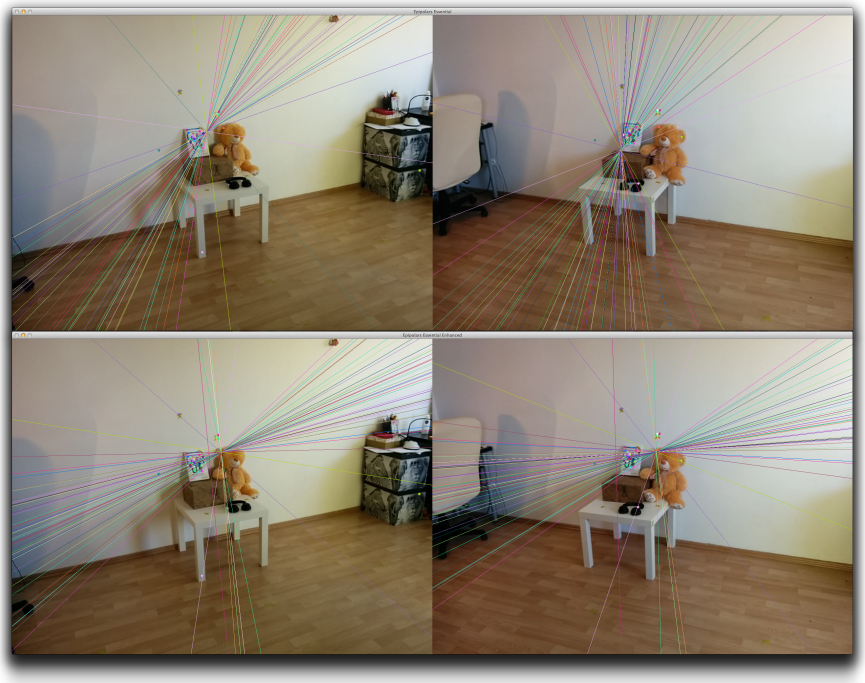

Figure 7.6: Comparison of epipolar lines for 5-point standard and enhanced versions for automatically matched points in images 0.jpg and 2.jpg from test dataset. Some outliers are produced during feature matching process.

Chapter 8

Structure From Motion with Sensor Fusion

All implemented algorithms were tested in terms of speed, accuracy and effectiveness. As regards the speed test, it included the measurement of the reconstruction execution time. In the accuracy testing Sampson Error measurement was used (this variable measures the distance between all points and their corresponding epipolar lines in an image). Effectiveness was tested with the use of the visual comparison of reconstructed 3D cloud points.

8.1 Concept

8.1.1 Reconstruction process strategy

Finally, all methods described herein can be combined in different reconstruction strategies. For the initialisation of 3D cloud point the following strategies can be used:

1. **Known rotations and translations**, which theoretically allows for an metrical reconstruction
2. **Known rotations**, where translation is estimated with an alternative 3-point algorithm
3. **Noisy rotations**, where enhanced 8-point fundamental or 5-point essential algorithms are used in order to calculate rotation error and relative translation

4. **Unknown external camera parameters**, where standard 8-point fundamental or 5-point essential algorithms are used in order to calculate rotation and relative translation

For the pose estimation the following methodologies can be used:

1. **Known rotations and translations**, where no additional calculations are required and an metrical model can be acquired
2. **Initial rotation**, where relative translation needs to be calculated
3. **Noisy rotation and translation**, where both rotation and translation errors need to be calculated
4. **Unknown external camera parameters**, where standard pose estimation has to be used

All proposed strategies were implemented and can be tested in desktop Application.

8.1.2 Pose estimation

One of the main goal was to develop a more accurate and faster version of the reconstruction algorithm. That is why the relative pose estimation techniques were used to enrich models with additional points. In general, this approach produces less outliers than the homography merging techniques. This also allows to maintain the scale of the reconstructed images. For any point of the image, which has a corresponding 3D point the following condition is met:

$$x = P * X \tag{8.1}$$

where x is image point expressed in homogenous coordinates (x , y , 1) and X homogenous 3D point (X , Y , Z , 1). The projection matrix of the camera design is as follows:

$$P = K * \left[R \mid t\right] \tag{8.2}$$

8.1.2.1 Rotation enhancements

Applying the similar approach to initial pair reconstruction enhancements, it can be noted that:

$$\begin{aligned} x &= K * \left[R_{init} + dR \mid t\right] * X \\ x &= K * \left[T_{init} \mid 0\right] * X + K * \left[dR \mid t\right] * X \\ x - K * \left[R_{init} \mid 0\right] &= K * \left[dR \mid t\right] * X \end{aligned} \tag{8.3}$$

Substituting $x_m = x - K * \left[R_{init} \mid 0\right]$ the following is received:

$$x_m = K * \left[dR \mid t\right] * X \tag{8.4}$$

This can be solved using a standard PNP calculating algorithms. Rotation can be used as an initial solution for the pose estimation in order to focus solely on the rotation error and translation estimation.

8.1.2.2 Rotation & translation enhancements

Similar to equation 8.3:

$$\begin{aligned} x &= K * \left[R_{init} + dR \mid T_{init} + dt\right] * X \\ x &= K * \left[R_{init} \mid T_{init}\right] * X + K * \left[dR \mid dt\right] * X \\ x - K * \left[R_{init} \mid T_{init}\right] &= K * \left[dR \mid dt\right] * X \end{aligned} \tag{8.5}$$

end in the end by substituting $x_n = x - K * \left[R_{init} \mid T_{init}\right]$ the following is received:

$$x_n = K * \left[dR \mid dt\right] * X \tag{8.6}$$

This can also be solved using a standard PNP[**?**] calculation algorithm. Rotation and translation can be used as an initial solution for the pose estimation in order to focus solely on the rotation error and translation error estimation.

8.2 Implementation

This chapter describes the chosen implementation environment. It also describes an Android application which was created for the purpose of acquiring image sequences with additional sensor data. The structure of both Android and desktop projects is explained and essential implementation details of the proposed algorithms and strategies are discussed herein.

8.2.1 Enhancing pose estimation

OpenCV already has the rotation and translation enhanced pose estimation implemented. The only thing that has to be done is the conversion of euclidean rotation matrix to its Rodrigues representation. The initial rotation and translation need to be passed as input variables. Pose estimation methods implementations are as follows:

```
Mutiview::FindPoseEstimation(
        cv::Mat &rvec,
        cv::Mat &t,
        cv::Mat &R,
        cv::Mat &K,
        cv::Mat &distCoeffs,
        std::vector<cv::Point3d> ppcloud,
        std::vector<cv::Point2d> imgPoints,
        vector<int> inliers)
Mutiview::FindPoseEstimationEnhanced(
        cv::Mat &rvec,
        cv::Mat &t,
        cv::Mat &R,
        cv::Mat &RInit,
        cv::Mat &TInit,
        cv::Mat &K,
        cv::Mat &distCoeffs,
        std::vector<cv::Point3d> ppcloud,
        std::vector<cv::Point2d> imgPoints,
        vector<int> inliers)
```

can be found in 'Mutiview.cpp' file.

8.3 Evaluation

8.3.1 Acquiring datasets

For the purpose of analysis the following datasets were captured with the implemented "Sensor Enhanced Images Camera" application and Nexus 5 camera:

1. Warsaw University of technology main building
2. Advertisement Pole
3. Warsaw Business School Gate and Entrance
4. Warsaw Shopping Center Back
5. Warsaw Shopping Center Front

Since most of the algorithms proposed in the Chapter 4 require internal camera parameters to be known, the camera used in the course of conducting this study was calibrated and its parameters were stored in *"outcameradata.yml"* file. All of these datasets can be found on attached CD or in Github repository.

8.3.2 Test Environment

All tests were performed on MacBook Air with 1.7GHz dual-core Intel Core i7 processor and 8GB 1600MHz DDR3 RAM using "Enhanced 3D Reconstructer" implemented as described in the Implementation chapter. Numerical tests which allowed for measuring total errors and execution time were run on the "Warsaw University of Technology" dataset. Initial pair reconstruction ability of each method proposed was measured as well as various reconstruction strategies. Finally, for each dataset the most effective methods were used to reconstruct sparse models.

8.3.3 Testing initial pair reconstruction methods

The key question to be answered at this point was whether the proposed sensor enhancement gave better results than the standard algorithms. The following methods were tested:

1. **Standard 8-point** - based on the 8-point fundamental matrix decomposition, implemented in OpenCV
2. **Enhanced 8-point** - the proposed camera rotation enhanced version of the above 8-point algorithm
3. **Alternative 3-point** - the proposed 3-point algorithm for translation estimation.
4. **Known rotations and translations** - calculation from known cameras rotations and translations
5. **Standard essential 5-point** - based on the 5-point essential matrix decomposition, implemented in [**?**]
6. **Enhanced essential 5-point** - similar to 2. rotation-enhanced version of the above 5-point algorithm

8.3.3.1 Accuracy - Epipolar lines correspondence

In the case of initial pair images one of the most important factors include the epipolar constraint. With the use of a properly estimated fundamental matrix, drawing corresponding epipolar lines in both images is possible. Furthermore, points lying on two matching epipolars lines in different images can be easily matched. In other words, epipolar lines cross exactly the same points in both images. The more accurate they are the more corresponding pairs can be found, e.g. for the purposes of performing

dense reconstruction. Sampson Error is one of the metrics that can be used in order to estimate the accuracy of epipolars lines; the smaller the error's value the more accurate the lines. Its primary function is to measure the total distance between all points and their corresponding epipolar lines. However, each of the proposed methods has different outliers removal capabilities, therefore in order to compare their efficiency, the average of Samson Error per a corresponding pair was calculated. The following

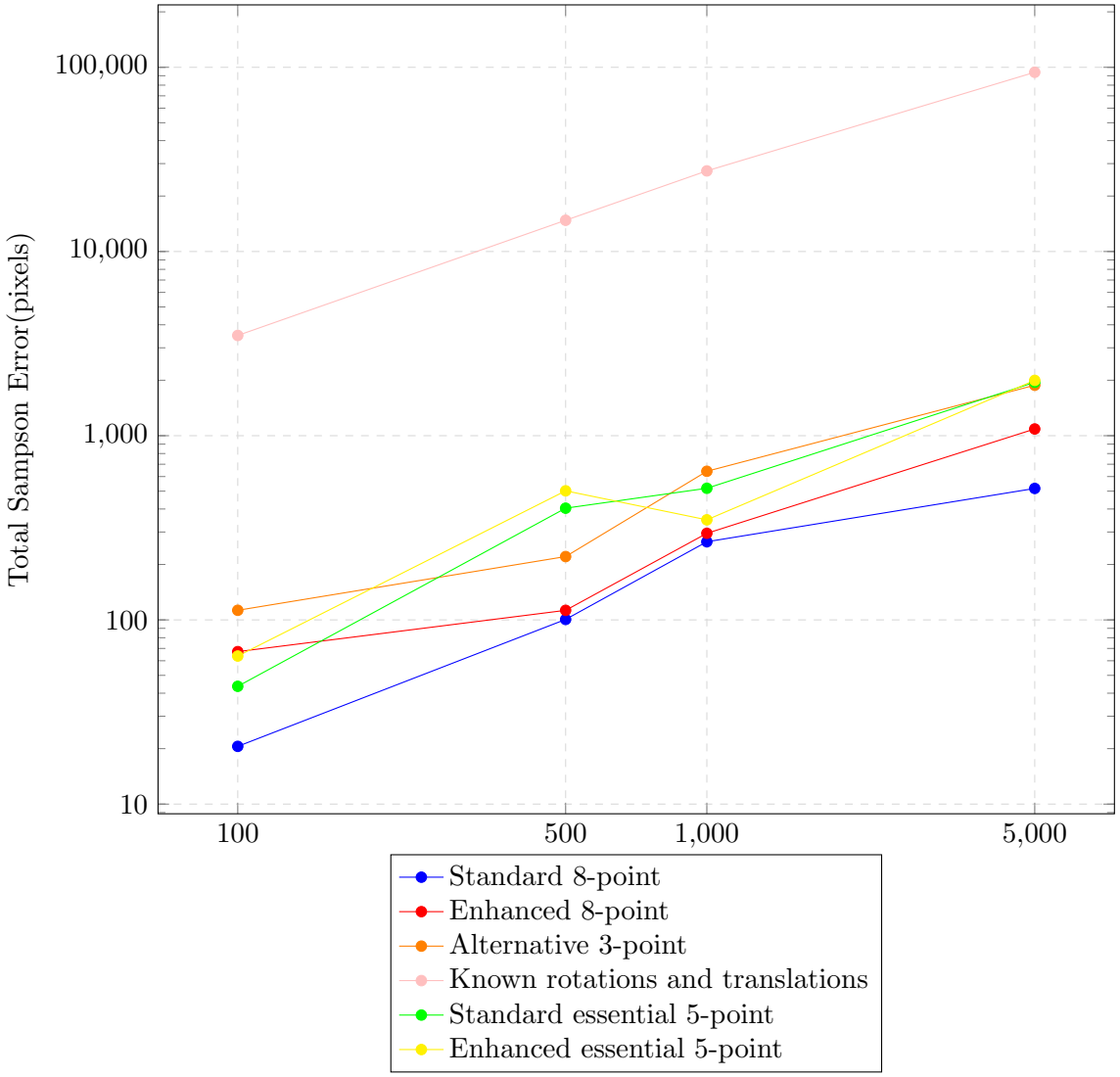

Figure 8.1: Chart showing the sum of Sampson Errors in a picture(1024x768pixels) per various initial Sift features sets sizes

chart 8.1 shows the total Sampson Errors for different sets of initial SIFT features. The major observation made based on these results is that most of the algorithms remove outliers properly. As could have been expected the only one that is not efficient in this regard is the method involving drawing epipolar lines from heuristically estimated movement and noisy rotation, which produces significantly larger error than the other algorithms. Analysing the per pair Sampson Error results 8.2 it can be noted that

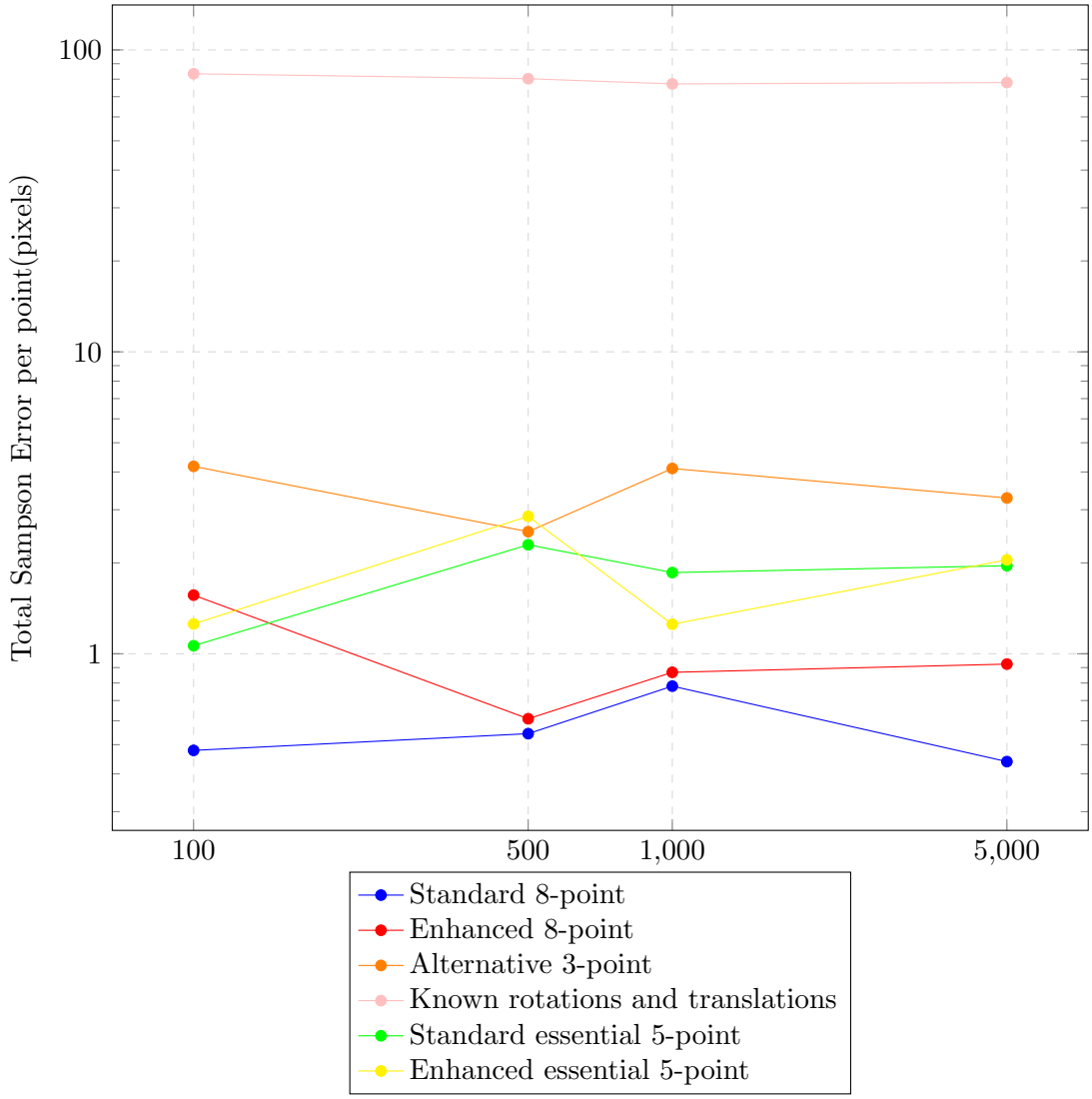

Figure 8.2: Chart showing per point Sampson Error in a picture(1024x768pixels) per various initial Sift features sets sizes

the proposed sensor enhanced methods did not improve either the 8-point or 5-point algorithms, but the 3-point algorithm developed for the purposes of this thesis proved to be much faster than the 8-point algorithm and also quite accurate. To present the discussed errors, estimated epipolar lines for 300 initial SIFT features set were drawn on the Figures 8.3 - 8.4. It can be seen that both the standard 8-point algorithm and the proposed rotation-enhanced version return very good results in terms of epipolar lines accuracy. The alternative 3-point algorithm gives slightly worse results due to the

Figure 8.3: The results of drawing estimated epipolar lines for the Warsaw University Dataset with 300 Sift points. 1) Standard Fundamental 8-point algorithm (upper pair), 2) Rotation Enhanced Fundamental 8-point algorithm (middle pair), 3) Alternative 3-point algorithm (bottom pair)

Figure 8.4: The results of drawing estimated epipolar lines for the Warsaw University Dataset with 300 Sift points. 1) Fundamental matrix created from rotation and translation (upper pair), 2) Standard Essential matrix 5-point algorithm (middle pair), 3) Rotation Enhanced Essential 5-point algorithm (bottom pair)

uncompensated mobile sensors' noise. In this particular dataset the essential 5-point method was inefficient in producing proper essential matrix estimation, contrary to the proposed sensor enhanced 5-point algorithm, which found satisfactory correspondency in epipolar lines.

To verify whether the epipolar lines calculation is influenced by the number of SIFT features, the same process was applied to 1000 SIFT features set (Figures 8.5 - 8.6). In general, the proposed enhanced algorithms are not better than standard versions in

terms of accuracy of epipolar lines estimation. It is mostly because during calculation some of the noise in sensor data is propagated on epipolar geometry estimation. However, the enhanced versions are always close to optimal solutions, while the standard versions can often fail in terms of fundamental matrix estimation.

Figure 8.5: The results of drawing estimated epipolar lines for the Warsaw University Dataset with 1000 Sift points. 1) Standard Fundamental 8-point algorithm (upper pair), 2) Rotation Enhanced Fundamental 8-point algorithm (middle pair), 3) Alternative 3-point algorithm (bottom pair)

Figure 8.6: The results of drawing estimated epipolar lines for the Warsaw University Dataset with 1000 Sift points. 1) Fundamental matrix created from rotation and translation (upper pair), 2) Standard Essential matrix 5-point algorithm (middle pair), 3)Rotation Enhanced Essential 5-point algorithm (bottom pair)

8.3.3.2 Time comparison

The chart 8.7 below shows the averaged execution time for 100 estimation attempts. Execution time of the 8-point algorithms are very similar, which is understandable,

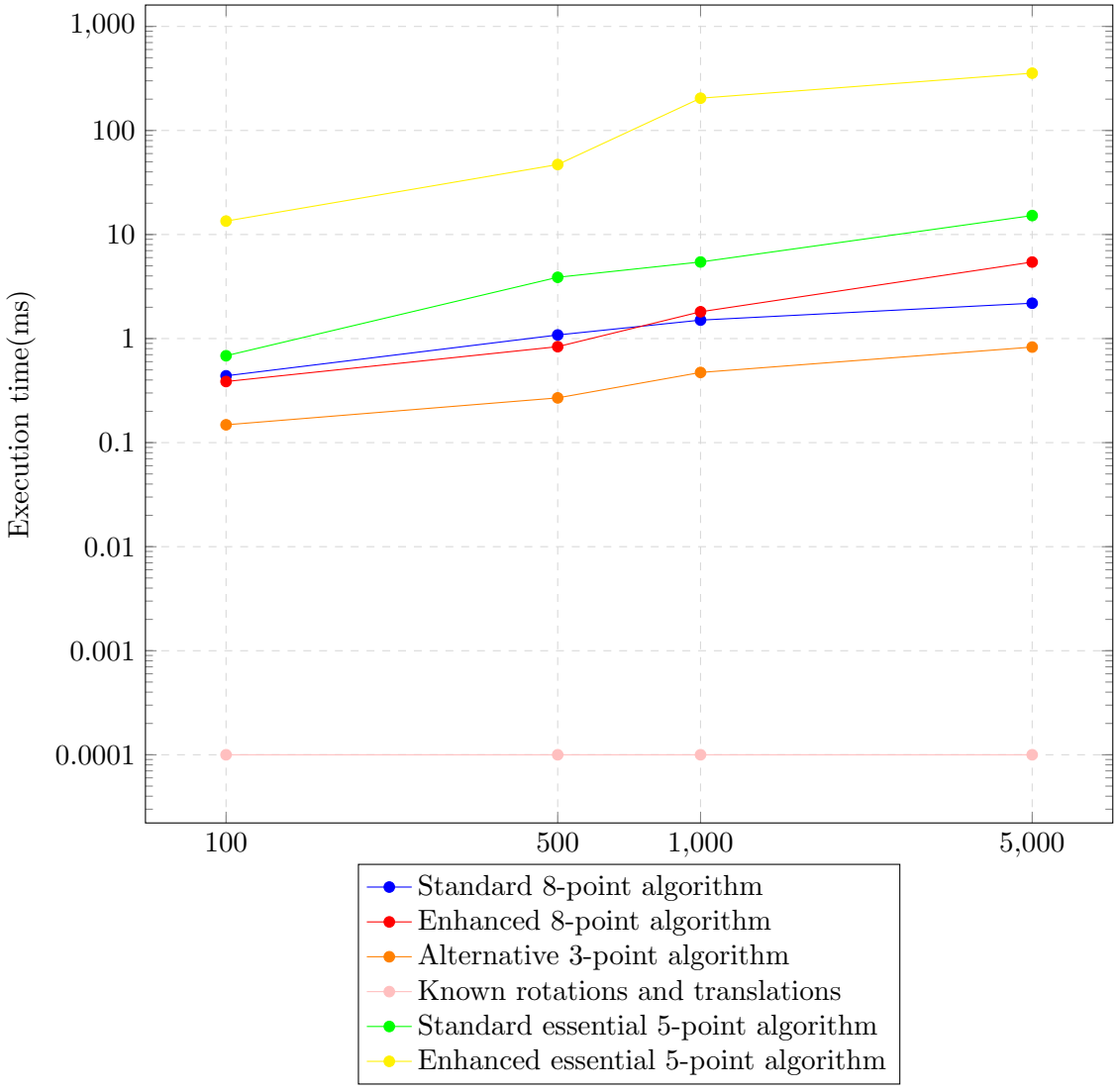

Figure 8.7: Execution time of the proposed algorithms(1024x768pixels) per initial SIFT feature set size

since their implementation is nearly identical and they differ only in the characteristics of the analysed dataset. It can be seen, however, that execution time values of the 5-point epipolar estimations differ significantly depending on the version. In this

situation either finding optimal solution with initial rotation is much more difficult or implementation of these methods is significantly different in terms of memory allocation. Execution time of the 3-point algorithm is few times faster than the one of the 8-point algorithms. The reconstruction with the sole use of the known rotations and translations has a hundred times shorter execution time than the standard algorithms, but at the same time it does not produce properly correlated epipolar lines.

8.3.4 Testing reconstruction strategies

As was shown in 8.3.3 not every initial reconstruction method gives good results. Only the most effective techniques were used to prepare a number of completely different strategies:

1. Standard 8-point + OpenCV Pose Estimation
2. Enhanced 8-point + OpenCV Pose Estimation
3. Enhanced 8-point + Initial Rotation and Translation OpenCV Pose Estimation
4. Alternative 3-point + OpenCV Pose Estimation
5. Alternative 3-point + Initial Rotation and Translation OpenCV Pose Estimation

The first method gives the basic reference to standard 3D reconstruction strategy. The second one was chosen for its consistency in 3D reconstruction. It has never failed to produce solution close to optimal. The third one was prepared in order to determine how the enhanced pose estimation influences the final reconstruction outcomes. The strategies number four and five were used to see if the models can be produced faster without impacting the final accuracy too heavily. Finally, the fifth method was proposed in order to verify whether the entire reconstruction process can be performed using the sensor data only.

8.3.4.1 Accuracy

Accuracy of the 3D reconstruction was measured using Bundle Adjustment algorithm from SBA library[ref]. It allows for calculating the error based on the projective constraint both before and after Bundle Adjustment. The tests performed with different strategies for the "Warsaw University" dataset are presented on 8.8. The significant differences in the initial error can be explained by different numbers of reconstructed points after initial phase and inconsistent and unknown scale of the finally reconstructed

models. The enhanced initial pair reconstruction and pose estimation methods have bigger impact on Bundle Adjustment error reduction.

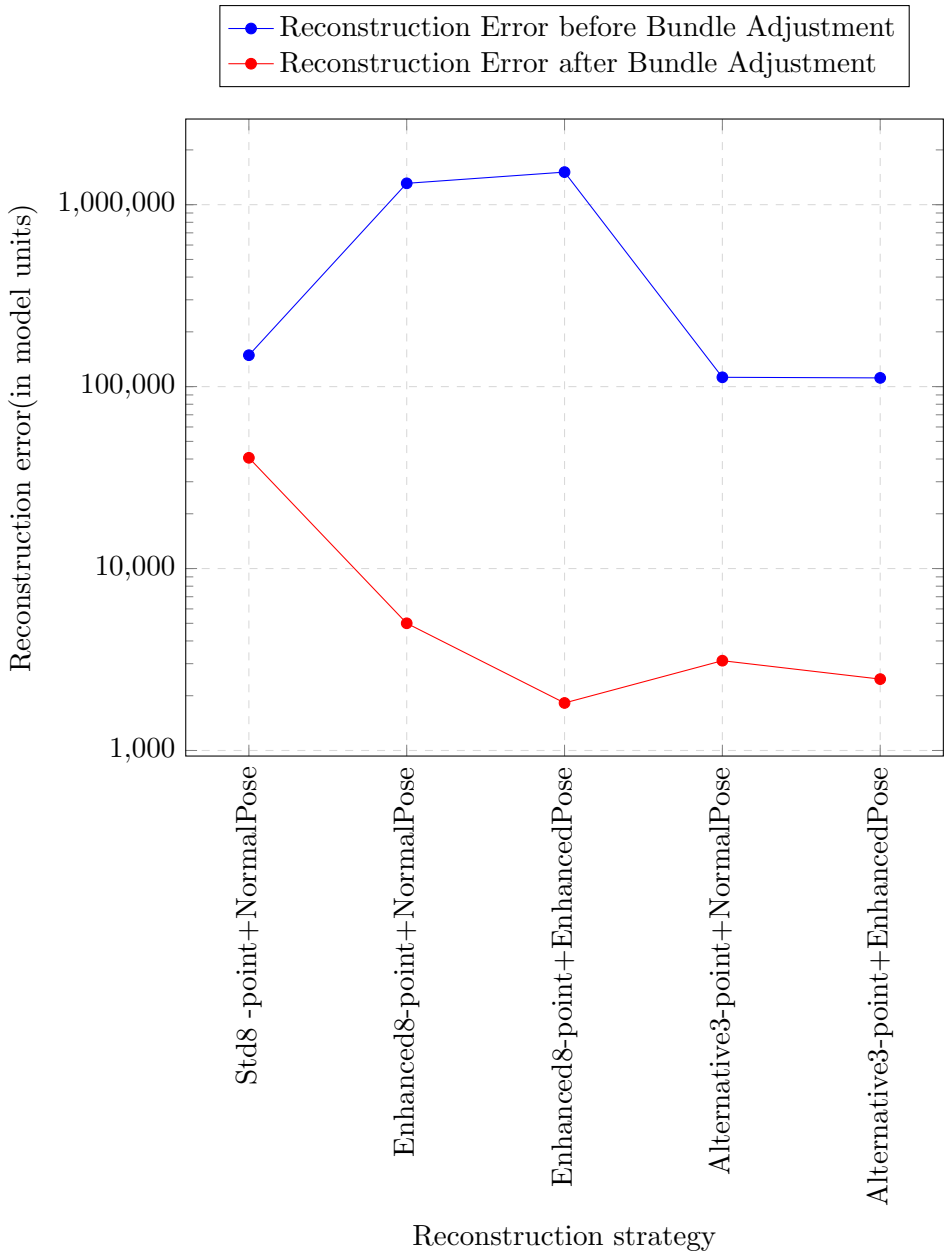

Figure 8.8: Influence of Bundle Adjustment on the models produced with different reconstruction strategies

In general, Bundle Adjustment process allows to rearrange and modify 3D points positions and change external parameters of estimated cameras. However, in the case of the enhanced algorithms the estimated camera positions are already very close to their optimal orientations. This allows Bundle Adjustment method to focus mainly on 3D points modifications. It can be observed that the enhanced pose estimation results in further reduction of BA error when compared to the standard strategies.

8.3.4.2 Execution time

The figure 8.9 presents the execution time without Bundle Adjustment. Comparing it to the execution times in 8.7 one can notice that most of the time is allocated for SIFT correspondences matching, which constitutes a bottleneck in the proposed reconstruction methodology. Furthermore, the reconstruction time was also measured with Bundle Adjustment process at the end(Figure 8.10).
It was found that Bundle Adjustment works significantly better with enhanced initial pair reconstruction and pose estimation. In that case the cloud points are better organised than in the standard reconstruction, which results in faster convergence of BA. Sample difference between the cloud points before and after BA can be seen in Figure 8.11.

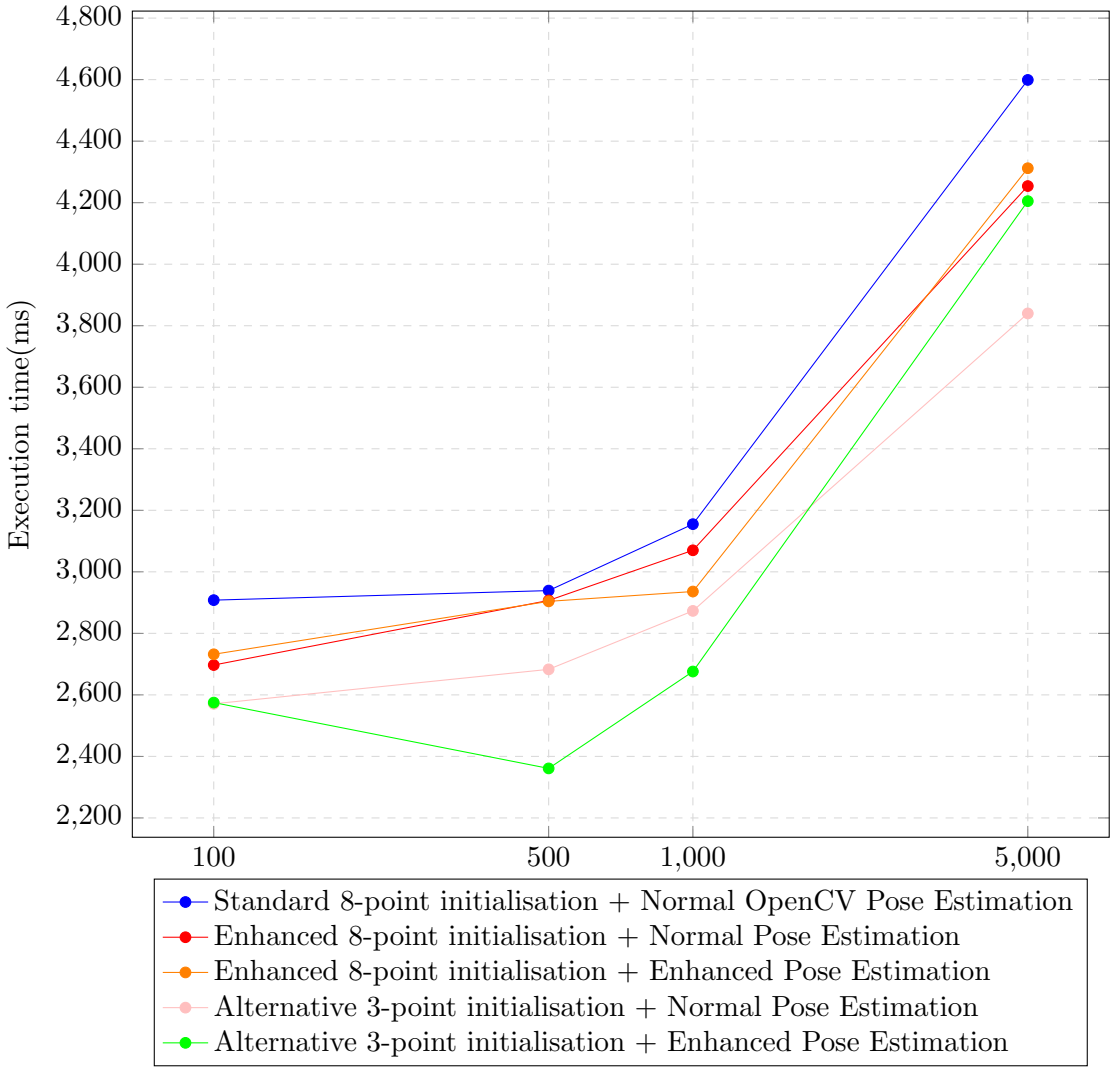

Figure 8.9: Total reconstruction execution time (4 images with resolution 1024x768pixels) per SIFT features set size

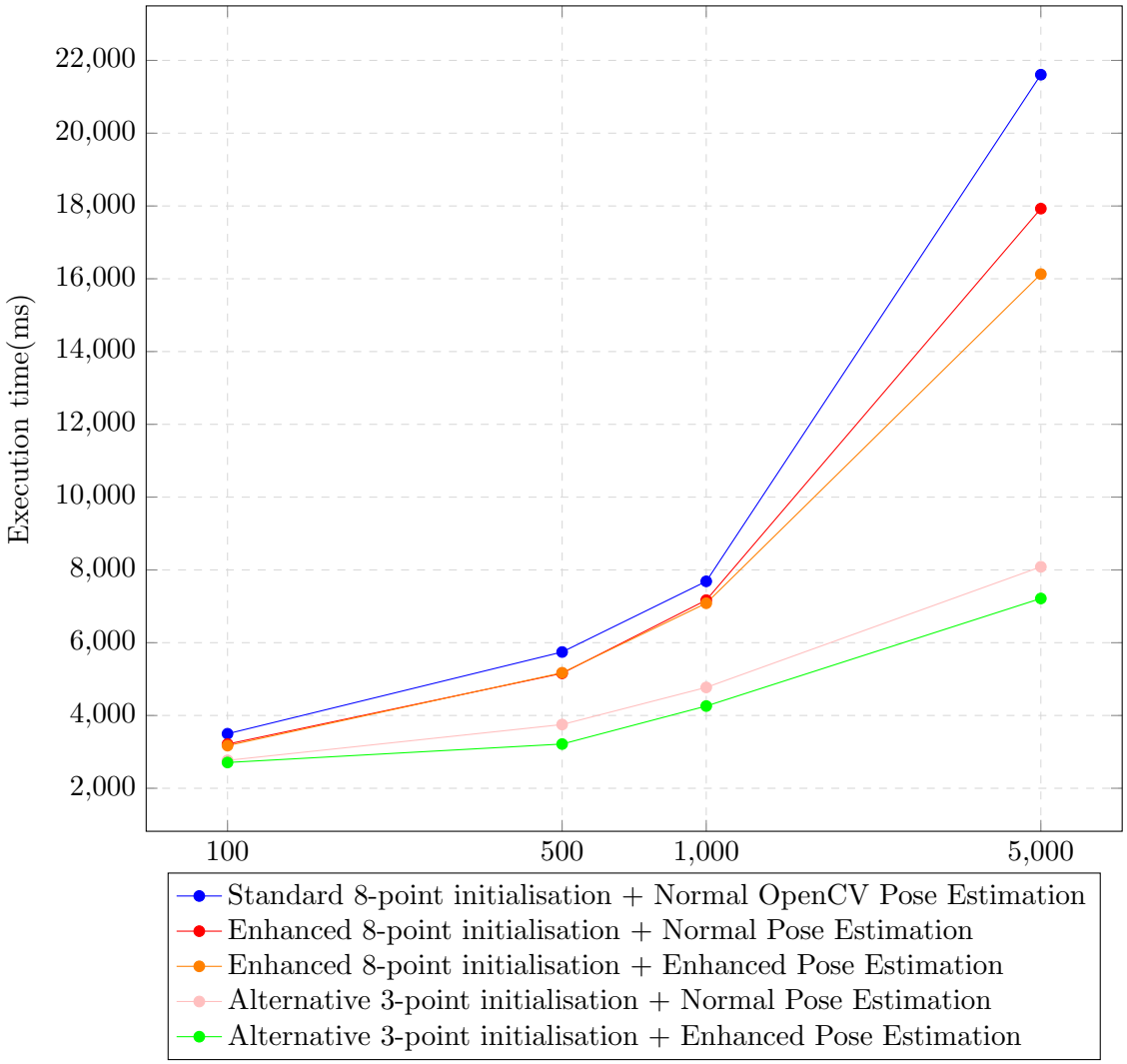

Figure 8.10: Total execution time of reconstruction with Bundle Adjustment (4 images with resolution 1024x768pixels) per SIFT features set size

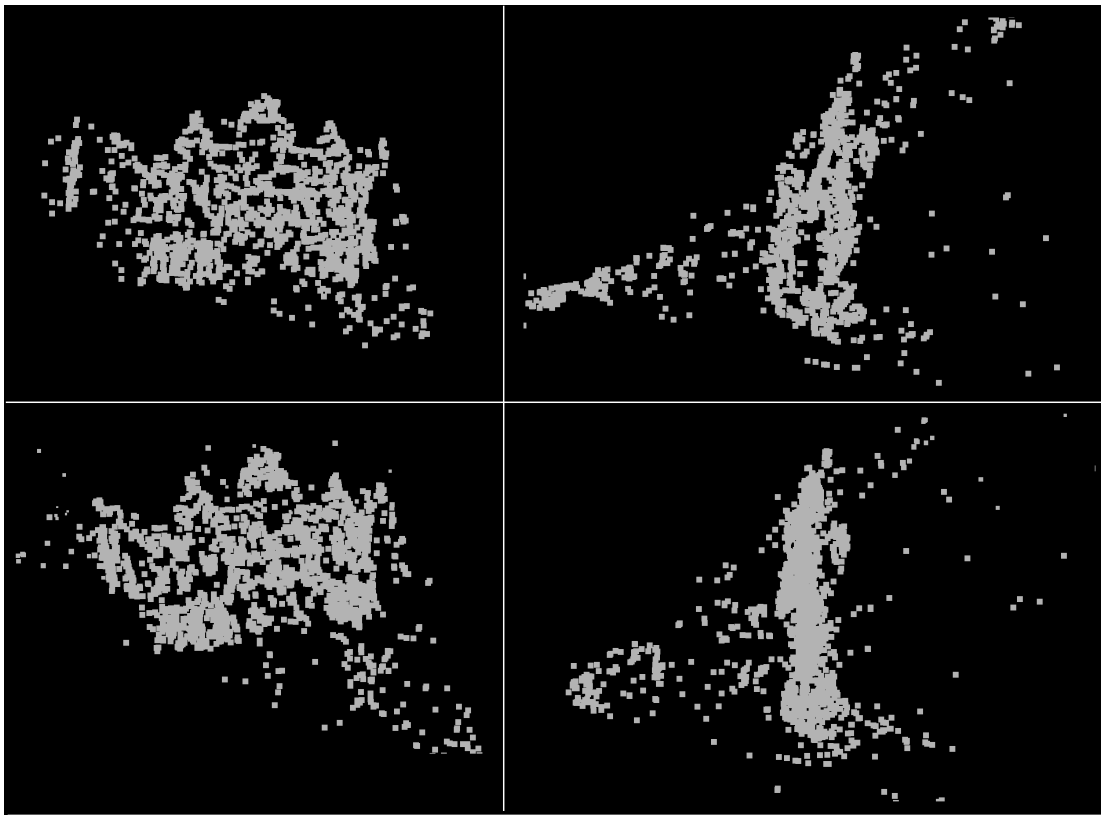

Figure 8.11: 3D point clouds before Bundle Adjustment (upper) and after (bottom) for enhanced 8-point with enhanced pose estimation. Warsaw University of Technology dataset with 1000 SIFT corresponding features (left - front, right - side)

8.3.5 Effectiveness

The numerical measurements used do not always correspond to the proper 3D model reconstructions. The following pages present the reconstruction effects of the proposed strategies. Figure 8.12 shows the reconstructed effects for different initialisation pair methods and 4000 SIFT features set. It can be observed that both standard and enhanced 8-point methods produces very good results, which differ only in terms of final reconstruction scale. Alternative 3-point algorithm produces slightly worse and distorted models due to uncompensated noise in the rotations of the camera used. A model reconstructed from the known rotations and translations is very distorted, but nonetheless still recognisable. It could prove useful should a very fast reconstruction be needed.

The matter is slightly different when 400 SIFT feature points are used. In the first case all algorithms were able to find solutions close to optimal. However, in the second attempt the triangulation test, which is used to identify proper decomposition in standard 8-point approach, failed and produced an unrecognisable model (Figure 8.14).

It can be seen that pose estimations enhancements result primarily in the reduction of outliers (Figure 8.15).

Figure 8.16 shows that reconstruction from the known rotations and translations produces many outliers. More reconstructed models can be found in Additional Materials.

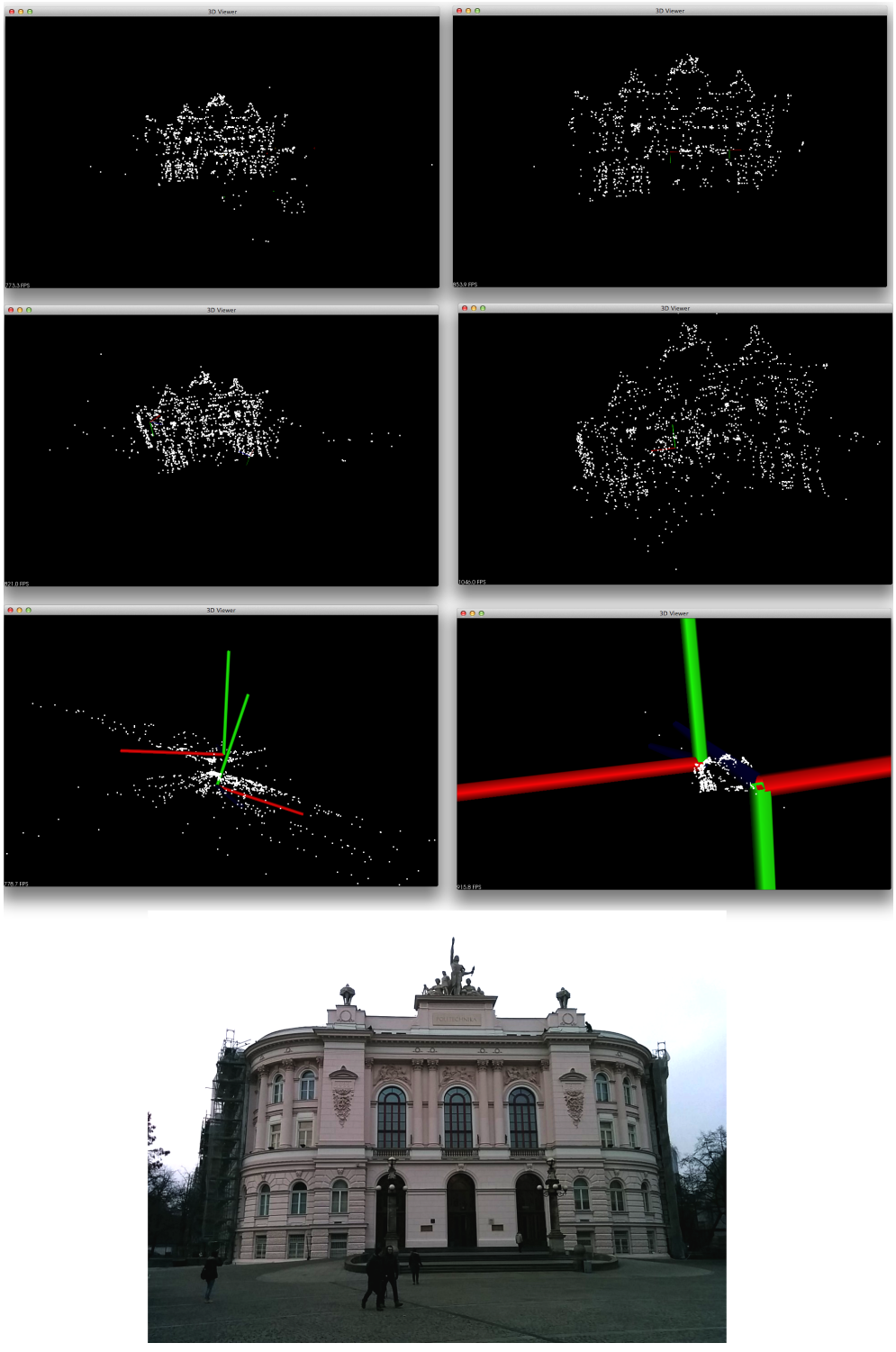

Figure 8.12: Reconstructed models for the proposed initial reconstruction methods and 4000 SIFT features. From upper left to bottom right: 1) standard 8-point, 2) enhanced 8-point, 3) alternative 3-point, 4) known rotations and translations, 5) standard 5-point, 6) enhanced 5-point

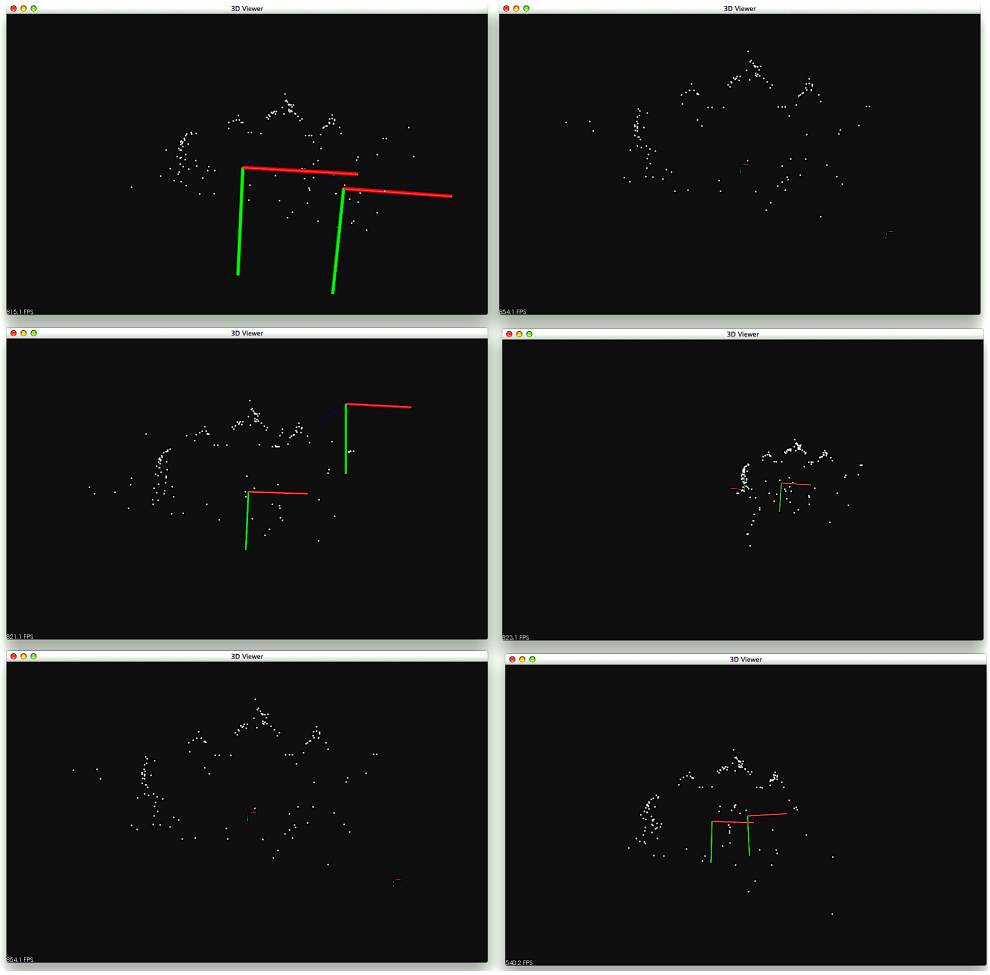

Figure 8.13: Reconstructed models for the proposed initial reconstruction methods and 400 SIFT features. From upper left to bottom right: 1) standard 8-point, 2) enhanced 8-point, 3) alternative 3-point, 4) known rotations and translations, 5) standard 5-point, 6) enhanced 5-point

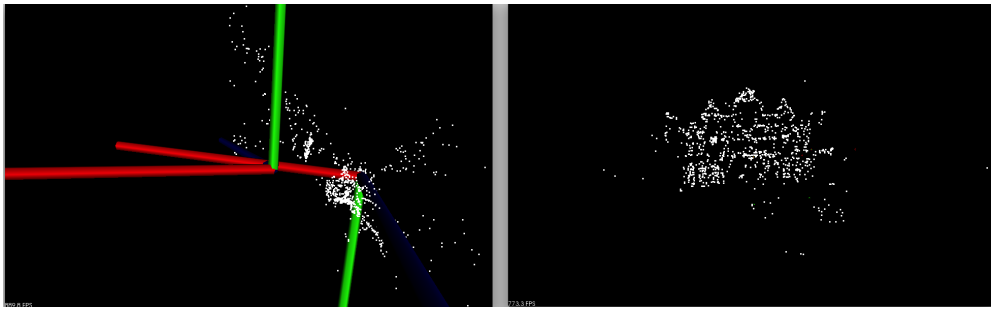

Figure 8.14: Fail test case of Standard 8-point triangulation(left) in comparison to fortunate reconstruction(right)

Figure 8.15: Pose estimation methods comparison (Views from front and side). Left: Normal Pose Estimation, right: Enhanced Rotation and Translation Pose Estimation. Less outliers appear in the reconstruction if enhancement is applied.

Figure 8.16: Reconstruction results from known translations and rotations from different angles. The upper one shows the front face of building, the others present views from side angles. In the reconstructed model many outliers are present.

Chapter 9

Conclusions

This chapter describes the work performed in the course of investigating the master thesis subject, discusses the problems encountered and proposes ideas for the future development.

9.1 Summary

The idea of enhancing standard 3D reconstruction algorithms with Android sensor fusion data was conceptualised, implemented and verified in several different versions. The main invention of this thesis was a conception and implementation of improved Essential matrix decomposition by combining initial camera rotation estimation from sensor fusion with specifics of Rodrigues parametrisation for small error correcting angles (See equation 7.6).

At the beginning, only the alternative 3-point version was planned to be implemented for the purposes of this thesis. However, it did not produce results of enough accuracy. After few iterations of implementation and testing of the 3-point algorithm it was concluded that it is not enough to base the reconstruction solely on the rotation data from Android sensor fusion. That was when the enhanced 8-point and 5-point versions were designed, which focused on finding error-correcting rotation matrix of the camera orientation and thus limiting decomposition ambiguities. Decomposition, which is hard to do especially, when there are many outliers established during correspondence matching.

The proposed enhancements to the standard 8-point algorithm allow to unambiguously

calculate the proper rotation and translation (with eventually reversed baseline, which does not influence shape of reconstructed model). Application of the enhanced 5-point algorithm resulted in better accuracy than in the case of the standard 5-point algorithm in terms of Sampson Error. Execution time of the enhanced reconstruction methods is generally longer than of the standard ones. The proposed 3-point algorithm for translation estimation allows only for faster and but not very accurate recreation of the structure. Heuristic movement estimation is not entirely accurate and does not have significant impact on the reconstruction process.
In addition, the author proposed different reconstruction strategies for SfM process, which can be used depending on accuracy and speed trade-offs preferred. They were built on personal observations of the reconstructions performed and inspired by related works in the field of 3D reconstruction.
In order to acquire datasets for algorithms' evaluation, the Android application named "Sensor Enhanced Images Camera" was developed. Upon capturing the picture it automatically stores the current device's rotation information and the proposed heuristically estimated translation. To evaluate the proposed methods and collected datasets a desktop application named "Enhanced 3D reconstructor" was implemented. It can be used in two different modes:

1. Efficiency testing - for comparing Samson Error and visual estimation of calculated epipolar lines.
2. Reconstruction testing - for different reconstruction strategy testing and BA influence

What is more, the use of initial rotation and translation in pose estimation results in a greater reconstruction accuracy, particularly in terms of outliers removal, and an increase in Bundle Adjustment convergence speed.
In general, applying Bundle Adjustment to model reconstructed with sensor enhancement results in greater error reduction and shorter execution time in comparison to the models acquired through the standard process. The major problem, a bottleneck of some sort, of the proposed reconstruction process was the time needed for matching corresponding items.
To sum it up, in author's research and this document it was determined that usage of

Sensor Fusion data for initial rotation computation and estimation of rotation error matrix (R_{error}) gives more accurate results and is useful for generating reliable 3D models. The proposed Rodrigues decomposition with its rounding helps resolve ambiguities in decomposition of essential matrix to relative rotation and translation, especially when there are many mismatched corresponding points produced during feature matching phase.

9.2 Dissemination

So far no one has used the implemented applications. Nonetheless the Android application can be useful for further 3D reconstruction research and it is planned to be published to Google Play Store once most needed improvements are made and it is properly tested. "Enhanced 3D reconstructer" has been already published to GitHub as open-source project distributed on LGPL license and can be found here: [**?**].

9.3 Problems encountered

The majority of the problems were related to bugs which appeared during implementation of the proposed algorithms and adaptation of the third party libraries. Android API allows a user to obtain rotation quaternion and offers a way to decompose it, but it does not explain how it is calculated. Decomposition of rotation matrix to euclidean angles and their composition needs to be done in the same order. Some tests were conducted in order to establish its proper rotation matrix composition. All of these problems were successfully resolved. It turned out that using the pose estimation instead of homography merging was not the optimal solution. Relaying on the pose estimation produced few points and sometimes reconstruction was force-stopped after analysing merely a few images.

9.4 Future work

Firstly, it would be valuable to establish how the homography merging approach would influence the accuracy of 3D reconstruction. Secondly, other correspondence matching approaches should be tested. An optical flow estimation using video sequences could constitute one example thereof. This would both allow for a very quick relative pose

estimation and could be used for dense model reconstruction afterwards. Author of this research would like to implement such dense reconstruction techniques and produce nice-looking and textured 3D models with bigger resolution.
All of the libraries used are available or can be ported to Android, therefore it might be valuable to determine whether it is possible to achieve a real-time tracking and mapping on this platform (similar to [**?**]).

Chapter 10

Additional materials

```
[
    {
        "photoPath":"20141210_145643/0.jpg",
        "rotationMatrix":[],
        "azimuth":121.88075,
        "posX":-1.7521392107009888,
        "posY":-1.4345977306365967,
        "posZ":0.9248641133308411,
        "photoId":1,
        "pitch":13.867888,
        "roll":178.16968
    },
    {
        "photoPath":"20141210_145643/1.jpg",
        "rotationMatrix":[],
        ],
        "azimuth":110.66925,
        "posX":-4.244707942008972,
        "posY":-1.1443554759025574,
        "posZ":0.9647054933011532,
        "photoId":2,
        "pitch":11.625216,
        "roll":179.73383
    }
    ...
]
```

100 SIFT Features	Total Sampson Error	Sampson Error per Point	Points left	Execution time(ms)
8-point OpenCV	20.5793	0.478588	43	0.4387
Alternative 3-point	112.749	4.17588	27	0.1484
8-point enhanced	67.2559	1.56409	43	0.3867
Known rot and trans	3501.23	83.3625	42	0.0001
Essential 5-point	43.5866	1.06309	41	0.684
5-point enhanced	1863.66	45.4552	41	13.4643

Table 10.1: Efficiency table of proposed methods for 100 SIFT features in Warsaw University of technology dataset. Columns: Total Sampson Error, Average Sampson error per point, Amount of points left after outliers removal, Execution time

500 SIFT features	Total Sampson Error	Sampson Error per Point	Points left & Execution time(ms)	
8-point OpenCV	100.584	0.543697	185	1.0833
Alternative 3-point	220.722	2.53704	87	0.2692
8-point enhanced	112.7	0.609189	185	0.8362
Known rot and trans	14770.7	80.2756	184	0.0001
Essential 5-point	404.098	2.29601	176	3.8827
5-point enhanced	501.987	2.8522	176	47.0683

Table 10.2: Efficiency table of proposed methods for 500 SIFT features in Warsaw University of technology dataset. Columns: Total Sampson Error, Average Sampson error per point, Amount of points left after outliers removal, Execution time

1000 SIFT features	Total Sampson Error	Sampson Error per Point	Points left	Execution time(ms)
8-point OpenCV	265.637	0.781287	340	1.5055
Alternative 3-point	640.895	4.1083	156	0.4725
8-point enhanced	295.152	0.868093	340	1.8099
Known rot and trans	27394.4	77.1673	355	0.0001
Essential 5-point	518.293	1.85768	279	5.4482
5-point enhanced	349.393	1.2523	279	204.2998

Table 10.3: Efficiency table of proposed methods for 1000 SIFT features in Warsaw University of technology dataset. Columns: Total Sampson Error, Average Sampson error per point, Amount of points left after outliers removal, Execution time

5000 SIFT features	Total Sampson Error	Sampson Error per Point	Points left	Execution time(ms)
8-point OpenCV	517.189	0.439413	1177	2.187
Alternative 3-point	1879.98	3.28094	573	0.8286
8-point enhanced	1087.78	0.924199	1177	5.4395
Known rot and trans	93951.1	77.9677	1205	0.0001
Essential 5-point	1949.53	1.95736	996	15.2223
5-point enhanced	19966.6	20.0468	996	355.464

Table 10.4: Efficiency table of proposed methods for 5000 SIFT features in Warsaw University of technology dataset. Columns: Total Sampson Error, Average Sampson error per point, Amount of points left after outliers removal, Execution time

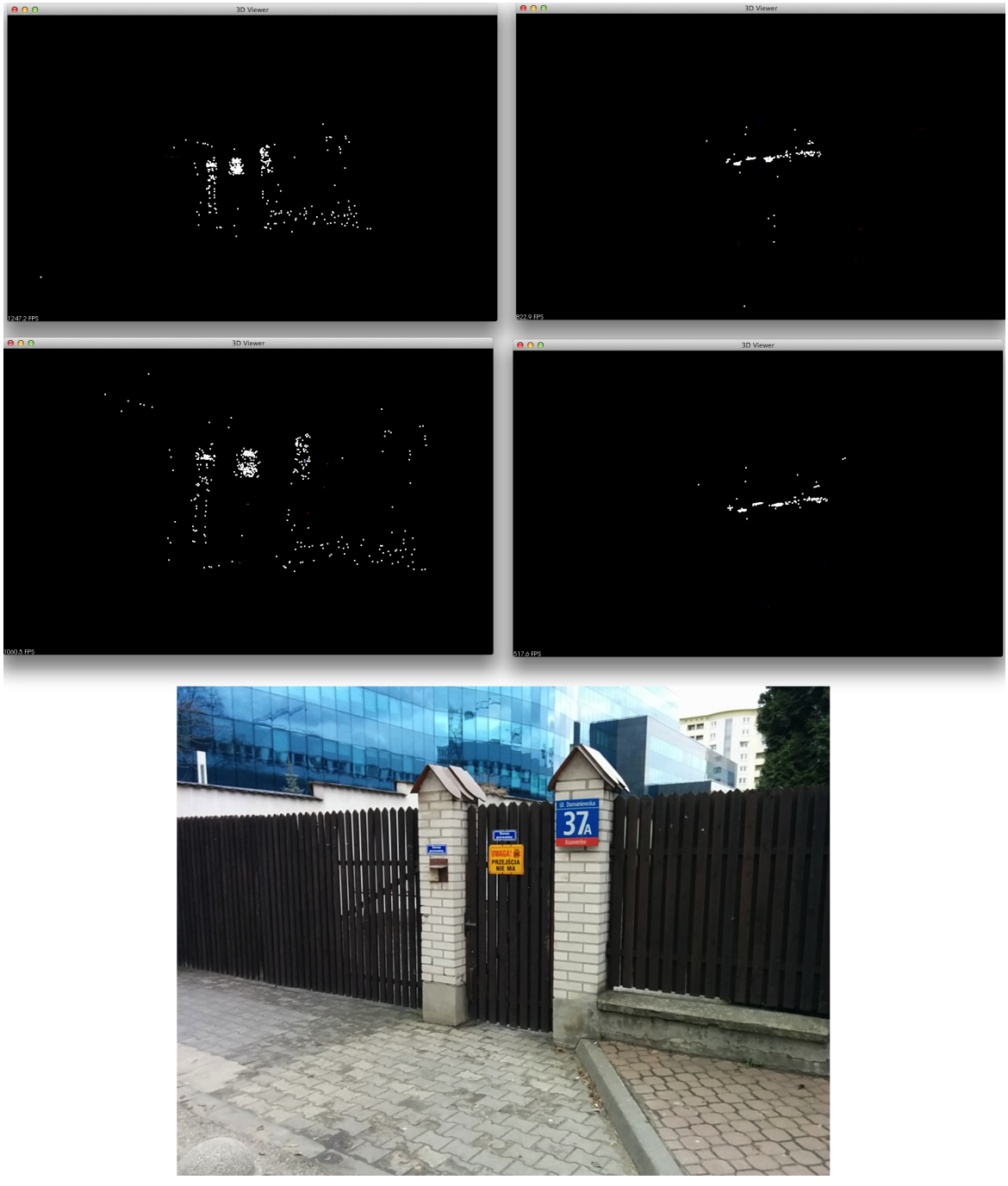

Figure 10.1: Reconstruction results of "Gate" dataset. Upper pair - standard OpenCV 8-point algorithm (left - front view, right - above view). Bottom pair - rotation enhanced 8-point algorithm.

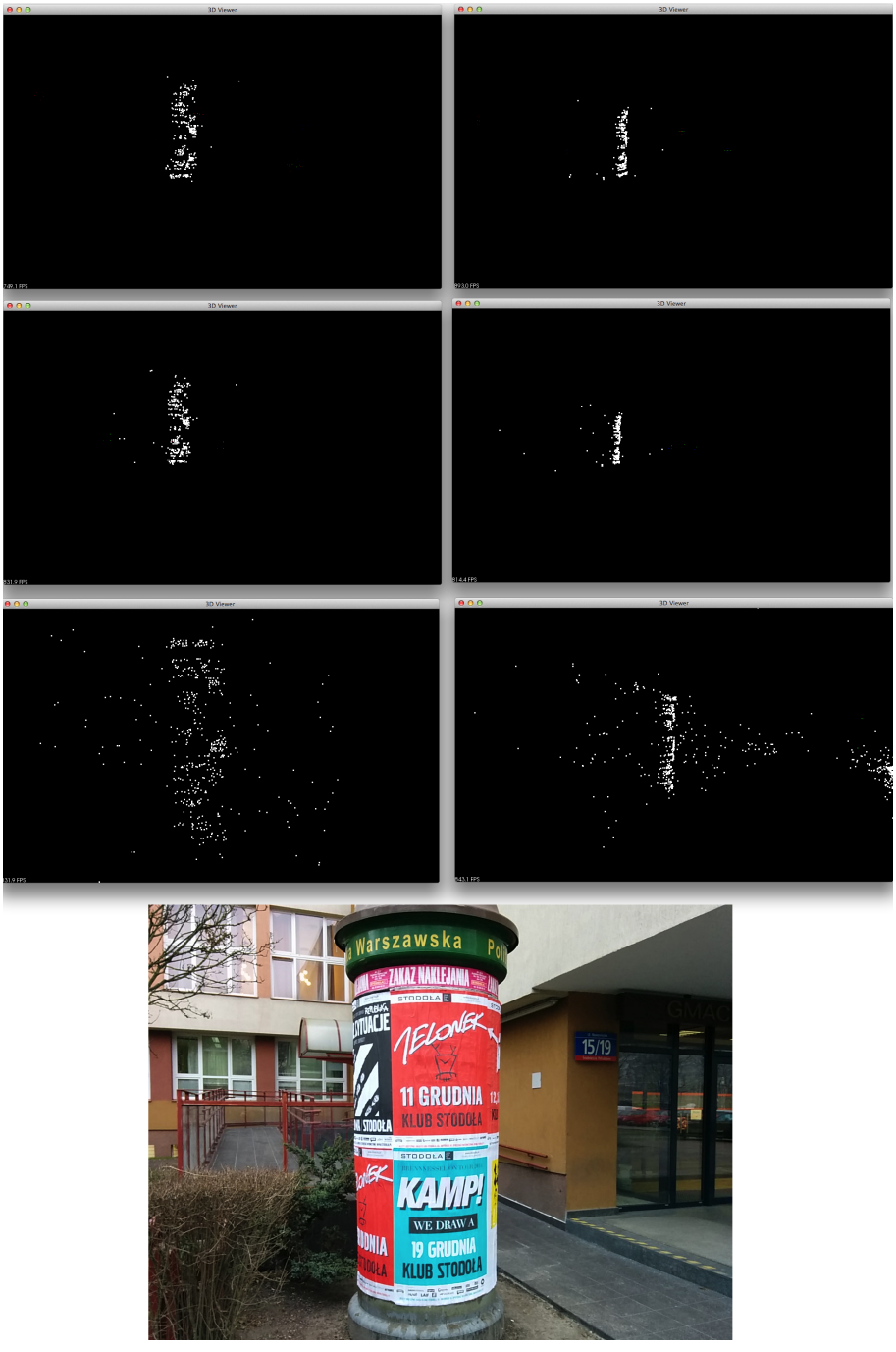

Figure 10.2: Reconstruction results of "Pole" dataset. Upper pair - standard OpenCV 8-point algorithm (left - front view, right - above view). Middle pair - rotation enhanced 8-point algorithm. Bottom pair - reconstruction from known rotations and translations.

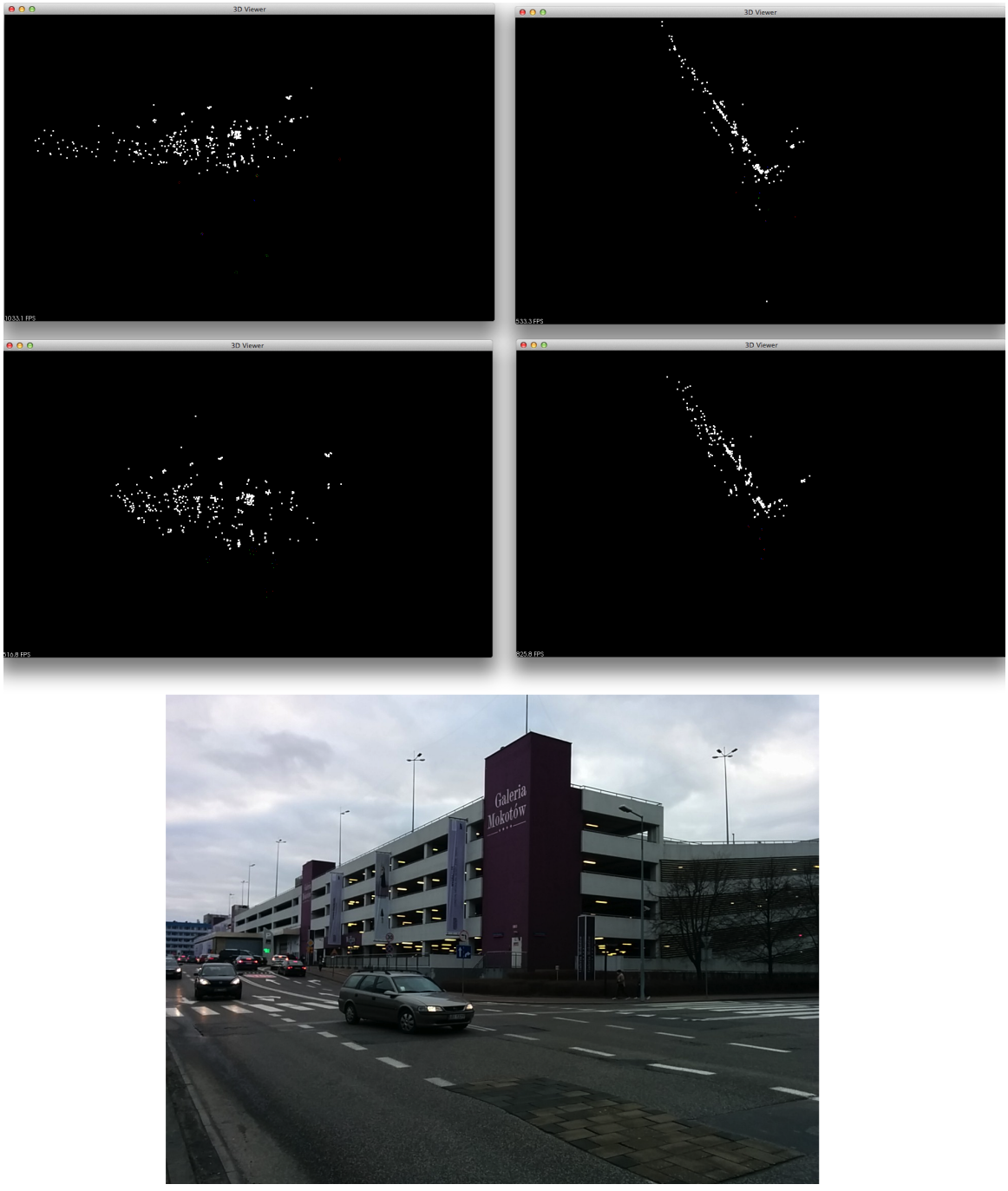

Figure 10.3: Reconstruction results of "Gallery Back" dataset. Upper pair - standard OpenCV 8-point algorithm (left - front view, right - above view). Bottom pair - rotation enhanced 8-point algorithm.

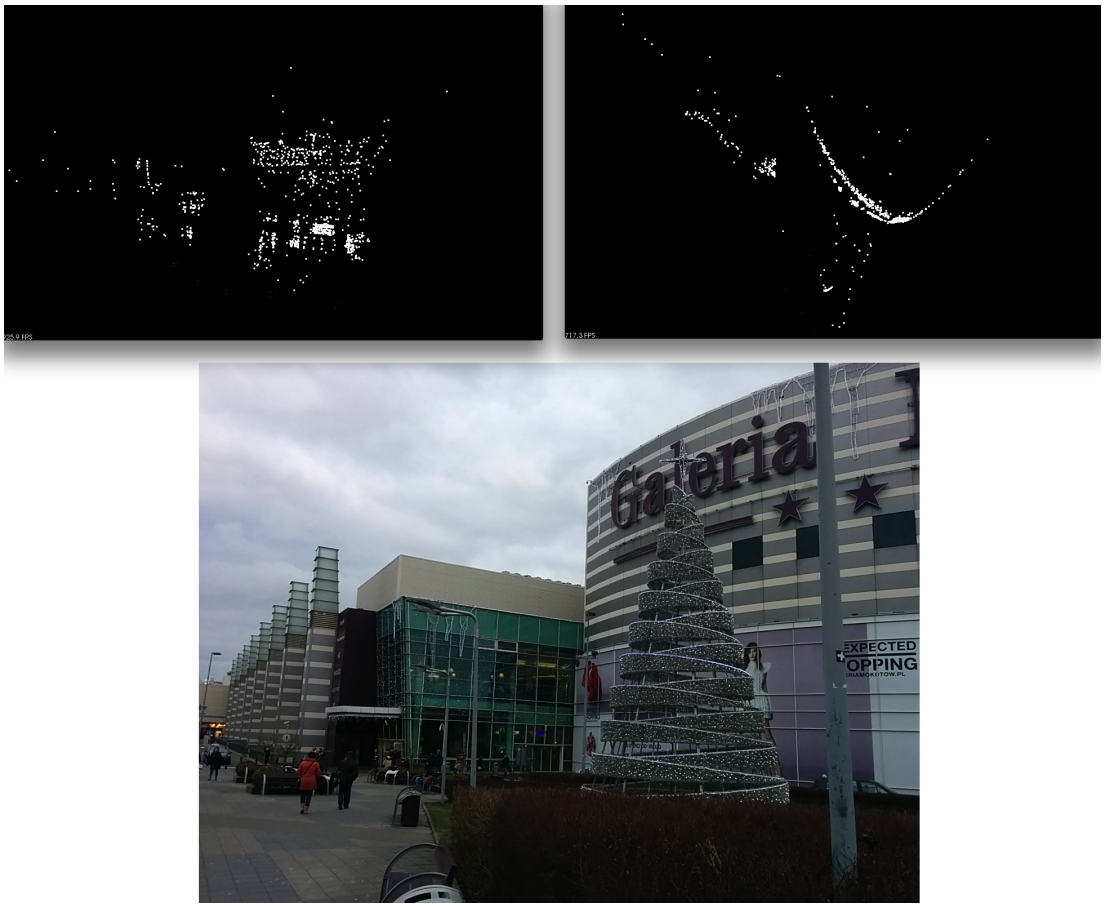

Figure 10.4: Reconstruction results of "Gallery Front" dataset using rotation enhanced 8-point algorithm (left - front view, right - above view).

References