Enhancing 3D reconstruction with Android Sensor Fusion data

Krzysztof Wrbel

CollegeOrDepartment

Technishe Universitt Berlin

A thesis submitted for the degree of

Master of Science

December 2014

1. Reviewer: Name

2. Reviewer:

Day of the defense:

Signature from head of PhD committee:

Abstract

Put your abstract or summary here, if your university requires it.

To ...

Acknowledgements

I would like to acknowledge the thousands of individuals who have coded for the LaTeX project for free. It is due to their efforts that we can generate professionally typeset PDFs now.

Contents

List of Figures

List of Tables

Acronyms

BA Bundle Adjustment. ix, 3

Base line The distance between the left and the right camera in a stereo pair.. ix

Epipolar lines TODO. ix

Essential Matrix TODO. ix

Extrasinc??? camera parameters TODO. ix

FoV Field of View. ix

Intersinc camera parameters TODO. ix

Pitch Rotation around the y axis.. ix

Pose Estimation TODO. ix

Quaternion TODO. ix

RANSAC Bundle Adjustment. ix

Roll Rotation around the x axis.. ix

Rotation Matrix TODO. ix

SIFT TODO. ix

SVD Singular Value Decomposition. ix

Yaw Rotation around the z axis.. ix

Chapter 1

Introduction

Mobile and wearable devices are becoming more and more popular. Modern smartphones despite having extremely good camera's also use advanced sensor's, like Accelerometers, Gyroscope, Magnetometer, Barometer etc.. There is also a big need and growing market of Augmented Reality (AR) and Virtual Reality(VR). That's why image analysis and recognition, as well as 3-D reconstruction techniques are really hot topic. Unfortunately algorithms that support these techniques are very time and memory consuming, that's why it's really hard to run them on mobile devices, which have many limitations in terms of CPU speed and RAM memory capacity.

Today many devices are capable of 3D reconstruction. One of them is very popular Kinect(**?**). It has ability of performing real-time 3D cloud point generation, but it has very high accuracy of reconstruction, but in the same time it's very expansive and not exactly mobile.

1.1 Purpose of this thesis

Author of this document will present the reader with an overview of the idea of 3D reconstruction. This thesis also inculdes brief description of related research in this area. After short analysis of efficiency, accuracy and common problems of few chosen algorithms this thesis will propose their enhacment with data acquired with sensors, which can be found in smartphones. At the end author presents evaluation and discuss test results. TODO finish

1.2 Scope

The author researched, how Accelerometer, Gyroscope and Magnetometer can be used in order to improve Fundamental, Essential matrix and also relative Pose Estimation. Unfortunately raw data sensors are really noisy and it's really hard to use them individually to enhance reconstruction. However there is a way to combine these data together in order to compensate error of each individual sensor. The term describing this process is called "Sensor Fusion". This data fusion allows to estimates in real time a relative or global(in term of earth magnetic field) rotation and translation of the device. TODO finish

1.3 Initial assumptions

The general process of 3-D reconstruction is quite broad, that's why the author of the thesis focus only on certain aspects of this topic. That's why author didn't write algorithms from the scratch, but built his algorithms on top of OpenCV library and "Relative Pose Esitmation" Open-source project setuped by ..... . In terms of sensor fusion, currently state of art approach is used by most of big Mobile Operating Systems(Android, iOS, Windows Phone). That's why author used Sensor Fusion API from API and only wrote what's needed it terms of getting rotation and translation of the smartphone camera, when acquiring images for his research. TODO finish

1.4 Thesis Outline

In Chapter 2 something something and so on
In Chapter 3
In Chapter 4
In Chapter 5
In Chapter 6
In Chapter 7
In Chapter 8

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 sensor fusion performed with the use of accelerometer, gyroscope and magnetometer data.

2.1 3-D reconstruction in general

There are many possibilities of performing reconstruction, starting from two-view reconstruction, multiple-view reconstruction and ending with the use of stereo calibrated cameras like the ones used in Kinect[reference]. Reconstruction can also be performed with a single hand-held camera either from a video or a sequence of images. As few as two pictures taken from different angles of a single object are sufficient to perform 3D model generation. Reconstruction process consists of the following steps:

1. **Image Acquistion**, where images are acquired
2. **Feature extraction and corresponding points matching**, where distinctive features are extracted from the images and compared
3. **Fundamental & Essential Matrices**, where matrices meeting the requirements of basic epipolar geometry are calculated
4. **Camera parameters estimation**, where external and internal camera parameters are estimated
5. **Triangulation**, where camera projection matrices are composed and used in order to calculate 3D cloud points

2.1.1 Feature extraction and corresponding points matching

Usually, each image used in reconstruction has to be analysed in order for the dstinctive features to be found. Afterwards all features in images are compared in order to find corresponding matches (2.1). There are multiple features detectors and extractors available for use `$http://en.wikipedia.org/wiki/Feature_detection_%28computer_vision%29$`. Some of them are more suitable for edge detection, while others are best used for corner or blob detection. One of the most popluar and robust feature detection method is scale-invariant feature transform (SIFT) `http://en.wikipedia.org/wiki/Scale-invariant_feature_transform`. The use of these descriptors allows for detecting local features in images and describing them with metrics including scale, rotation and translation invariant.

2.1.2 Fundamental & Essential Matrix estimations

Once proper matches are found, it can be proven that there exsists Fundamental matrix F for which the following equation is satisfied:

$$x'^{T} * F * x = 0 \tag{2.1}$$

where x and $x\prime$ are uncalibrated notions of points correspondence (**?**)TODO chapter. It is known that soultuions of this equation are highly sensitive to the occurence 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 (Bundle Adjustment (RANSAC)) `http://en.wikipedia.org/wiki/RANSAC`. The example of sample fitting can be seen in 2.2 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. The use of F matrix allows for calculating epipolar lines for each point (2.3). These lines cross the exactly same points in both images and can be used for dense feature matching since matches need to be searched for exclusively in the surroundings of these lines. When enough inliers are found, points that do not satisfy the equation can be removed from further processing. Once internal camera parameters K are known, the image points found can be calibrated and expressed in

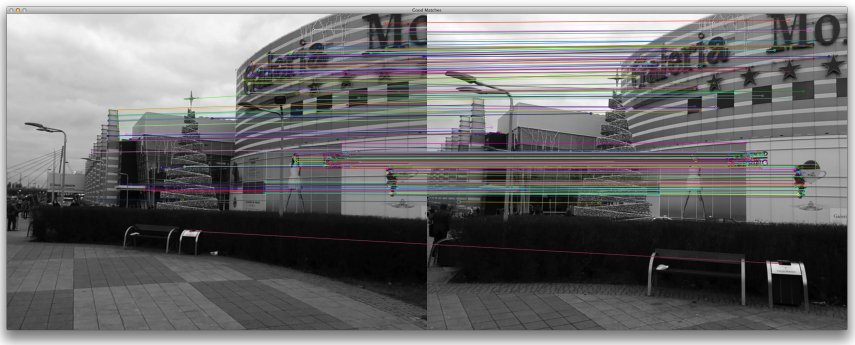

Figure 2.1

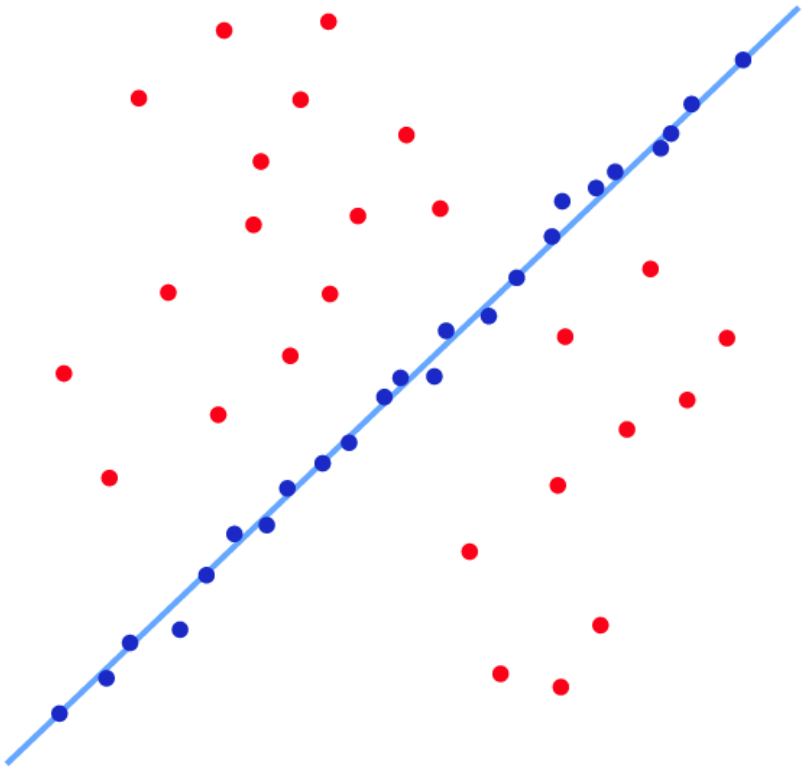

Figure 2.2: RANSAC fitting for 2D image TODO reference

camera reference position system. Such calibrated points satisfy the following essential matrix E equation:

$$x_c^{'T} * E * x_c = 0 \tag{2.2}$$

which is very similar to a fundamental equation 2.1. This results in:

$$E = K^T * F * K \tag{2.3}$$

with K being the internal camera parameters. This equation 2.3 is important in terms of decomposition of F matrix to relative rotation and translation.

2.1.3 Camera parameters estimations

Internal camera parameters are expressed by the following matrix:

$$\begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{2.4}$$

where x = fmx and y = fmy represent the focal length of the camera expressed in pixel dimensions in the x and y direction respectively. Similarly, x 0 = (x0,y0) is the principal point of pixel dimensions. These parameters need to be calculated only once for each camera model. Cameras can be calibrated with special reference boards of the known dimensions and chracteristics. For the proper 3D reconstruction it is essential to properly estimate external camera parameters, such as rotation (orientation angles of the camera) and global postition of the camera. It is often the case in 3D reconstruction that these parameters are not known. However, it is shown in Chapter 9 of Multiple View Geometry in Computer Vision ((**?**)), how essential matrix can be decomposed using Singular Value Decomposition Singular Value Decomposition (SVD) to relative camera positioning system of two projections:

$$P1 = K * \begin{bmatrix} I \mid 0 \end{bmatrix} \tag{2.5}$$

, the second one is equal to

$$P2 = K * \begin{bmatrix} RDiff \mid tDiff \end{bmatrix} \tag{2.6}$$

Unfortunately, there are four possible solutions for such a decomposition and it is not always possible to identify the correct one.

2.1.4 Points Triangulation

Once internal and external (global or relative) camera parameters are calculated, the triangulation can be performed in order to acquire an up-to-affine reconstruction model (**??**). The only elemnt that cannot be determined in such a relative case is the scale. This process is described in detail in [TODO hartley chapter 10].

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.

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 occurence 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 exising 3D cloud quality.

2.2.2 Homography estimation

A new image can be reconstructed with a previous one in a standard way in order to recive an up-to-scale 3D model. Afterwards, a newly acquired model can be merged into an existing one using homography estimation between the coresponding 3D points. Such a strategy is slower than the previous one, but it is not influenced by the quality of the existing 3D model. Although it is highly sensitive to outliers as well, if used properly it can produce more new 3D points.

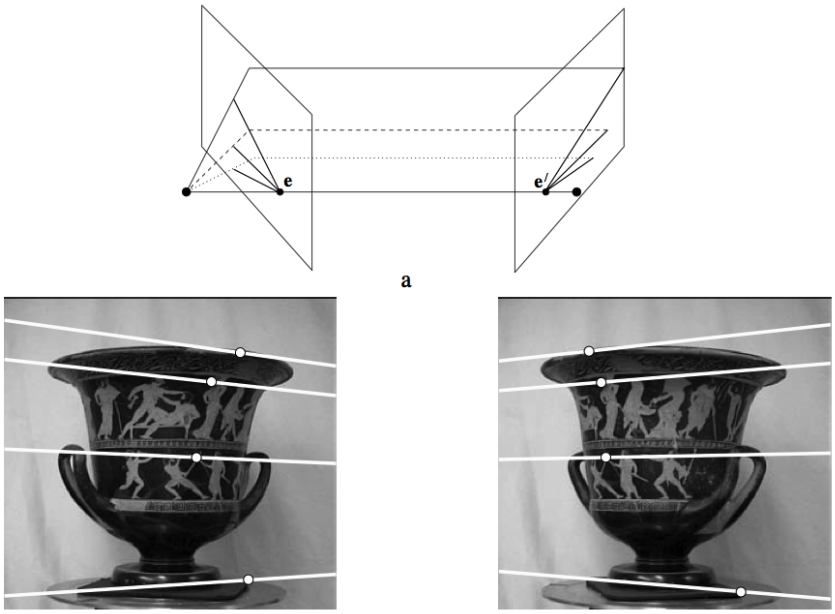

Figure 2.3: Epipolar lines found in an image of a vase TODO reference

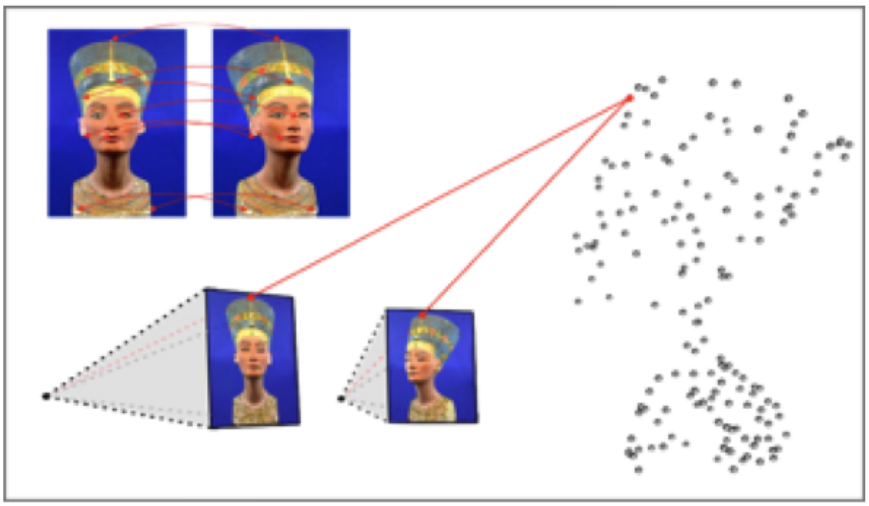

Figure 2.4

2.2.3 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 Bundle Adjustment (BA) `http://en.wikipedia.org/wiki/Bundle_adjustment`. The algorithm, using the information concerning the corresponding matches between multiple sets of images, iteratively modifies either both the camera external paramaters 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 expresed in figure 2.5.

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 disadvanteges, and thus the errors of some might be compensated for with the strengths of the others in otrder to, for example, accuretely compute camera rotation angles.

2.3.1 Accelerometer

An accelerometer is a device that measures acceleration along 3 axes of the device [reference]. Generally, an accelerometer allows for measuring total acceleration by sensing what force is applied to its micro strings. An accelerometer, which lies on a flat surface perpendicular to the Earth's surface will indicate approximately 1G upwards. This gravitation vector can be used to caculate 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. The gravity vector is obtained from the accelerometer data and tracked during unexpected movements with the used of gyroscope sensor.

2.3.2 Gyroscope

A gyroscope is a device for measuring or maintaining orientation, based on the principles of conservation of angular momentum [reference]. 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 [reference]. 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.

2.3.3 Magnetometer

A magnetometer is an instrument used to measure the strength and/or direction of the magnetic field in the surrounding area of the instrument. Its main idea works in the same manner as the conventional compass. With some magentometers magnetic field can be measured in three directions, relative to the spatial orientation of the device [reference]. Using magnetometers allows for estimating relative position in geomagnetic north position system.

2.3.4 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) [reference]. One of the strategies available is Kalman filtering (2.6) This approach is used in sensor fusion available in Android API, where the gravity vector calculation is enhanced with the accelerometer and gyroscope data. The gravity vector can be decomposed in order to estimate the camera's rotation angles. Magnetometer can be used optionally in order to acquire rotation angles referrenced to the Earth's coordinate system. However, magnetometer is sensitive to changes in the electromagnetic fields, therefore its measurement can be noisy in confined spaces filled with electronic devices. Substracting gravity from the actual acceleration measurements allows for estimating linear acceleration. The use of linear acceleration makes it possible to measure the relative change of the device's translation. However, it requires a double integration

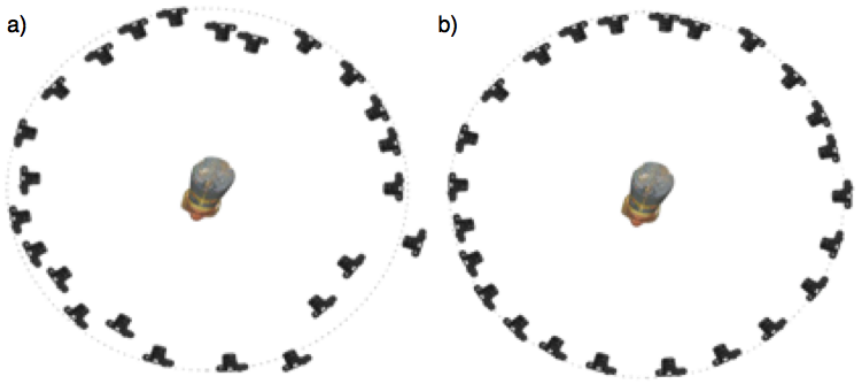

Figure 2.5: TODO reference

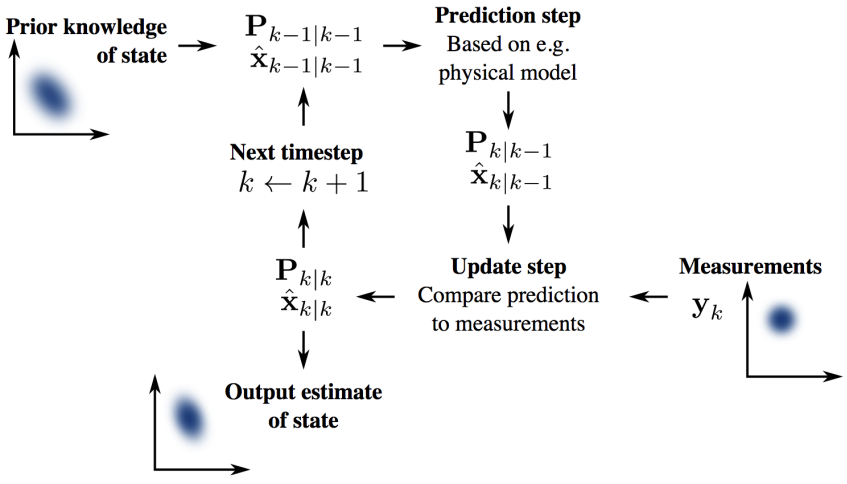

Figure 2.6

over time, which results in increasing the translation's noise. It is important to note that calculating rotation is significantly more accurate than estimating the translation.

Chapter 3

Related Work

There are many approaches to the reconstruction problems, starting with the raw one-by-one pixel analysis for density matching, 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 first two relative camera positions essential matrix must be decomposed. Since the basic epipolar geometry equation discovery, many scientist have introduced various ways to solve this linear problems, wchich 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 to find relative camera positions and that is why it is necessary to calculate internal camera parameters. Next, the 5-point algorithm approaches were introduced (**?**) (**?**), which used internal camera parameters for the proper essential matrix estimation. In his paper M. David Nister shows that the 5-point algorithm outperforms almost all similar algorithms in terms of accuracy and speed. Only the 8-point algorithm can regarded as competitive, when it comes to forward image sequences. 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 occurence of outliers. One of the most common approaches to use is RANSAC modelling(**?**).

A research group from Technishe Univeristt Berlin made an interesting comparison and evaluation of methods, which were discovered by 2008 (**?**). It was discovered that estimation of camera rotation is much more stable than translation calculation. Furthermore, there are a lot of ambiguities regarding the choice of the correct solution of epipolar geometry equation.

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, which uses 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 3 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.

Accuracy and speed constitute significant elements of the process of creating the systems capable of augmenting our reality. One of the first successful tracking and mapping reconstruction-based system was proposed by a research team from Oxford University (**?**). They showed how two simultaneously working threads can be used to both create a model of the environment being scanned and use this knowledge to apply graphical effects to objects presented on a video stream. Some of these concepts have already been applied to robotic vision. The authors showed efficiency of the proposed system for a robot walking in cluttered indoor workspaces(**?**). There were also approaches of porting reconstruction strategies to mobile devices(**?**) (**?**) (**?**) (**?**).

The major conclusions to be drawn include the fact that rotation estimation is more stable than translation estimation and that the reconstruction accuracy is improved by a better described model of camera movement. Some of the researches proposed 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 (**?**).

Chapter 4

Concept

As indicated in similar research it's very attractive to use additional data to enhance reconstruction and reduce ambiguity in finding correct solution of 3D reconstruction. Additional camera or model information help to implement faster, more stable and robust algorithms. This thesis will show how prior knowledge of rotation or translation acquired via mobile sensor fusion can be used to enhance process of 3D reconstruction from series images. When it comes to relaying on hand-held smartphones, collected sensor data are very noisy. This thesis, how even noisy informations can be used in struction reconstraction processes. Initially only camera rotation estimation was supposed to be used in the scope of this mater thesis. However first trials of reconstruction proved to be not sufficient enough and additional rotation error matrix estimations were proposed. From analysis of theory and related works it seems that both epipolar equations and pose estimation techniques can be improved with additional rotation and translation information data. Author of this thesis proposes environment, where user can decide what type of stargy to use.

4.1 Requirements

Proposed methodology needs as the input series of images with additional information about position of the camera - euclidan rotation and optionally translation. Usage of smartphone is actually not necessary. Any camera with SensorFusioned accelerometer and gyroscope(magnotometer is actually optional and as discussed in [TODO] has its advantages and disadvantages) capable of rotation and translation estimation can be

used. Internal camera parameters need to be calculated before reconstruction process is started. Additional sensor data doesn't have to be very accurate. Noisy external camera parameters stil can be successfully used in order to enhance reconstruction process.

4.2 Enhancing epipolar geometry equations with initial rotation matrix

Initial pair reconstruction step is very important and needs to be accurate to let other images calculate relative position on basis of initially reconstructed 3D cloud points. Taking standard fundamental geometry equation and relative camera based system ($P = \left[I|0\right], P' = \left[R|t\right]$) we can note that:

$$x_r^T * K^{-T} * \left[T\right]_x * R * K^{-1} * x = 0 \tag{4.1}$$

It is can also be written that:

$$\left[T\right]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} whereT = \left[t_x, t_y, t_z\right] \tag{4.2}$$

As we were discussing rotation can be distorted with noise. This can be written as:

$$R = R_{error} * R_{init} \tag{4.3}$$

where R_{init} is initial rotation matrix constructed from measured angles and R_{error} is rotation error matrix. Looking at this from different point of view 4.3 can be interpreted as multipling two rotations matrix: One estimated, but close to local optimum initial rotation matrix and second one, which is responsible for correction of noise error. Basic idea of algorithm proposed in this thesis is instead of relative whole rotation matrix calculation, which can be acquired from Essential matrix SVD decomposition, only rotation R_{error} can be estimated. In the end 4.1 can be rewritten in form:

$$x_r^T * K^{-T} * \left[T\right]_x * R_{error} * R_{init} * K^{-1} * x = 0 \tag{4.4}$$

Having:

$$\begin{aligned} h_r^T &= x_r^T * K^{-T} \\ h &= R_{init} * K^{-1} * x \\ G &= \left[T\right]_x * R_{error} \end{aligned} \tag{4.5}$$

With such notation one can notice that:

$$h_\prime^T * G * h = 0 \tag{4.6}$$

which resembles already known fundamental(**??**eq:fundamntalEquation) and essential equations (2.2). Of cource $h\prime$ and h both are expressed in homogenous coordinates in such situation. From analysis it is known that G has 6DOF: 3 due to unknown translation and another 3 due to unknown correction angles(created by rotation error matrx decomposition). From theory such matrix can be resolved for instance by both 5 and 8-point algorithms. So basicly standard fundamental and essential equation solvers can be used in order to retrieve both $\left[T\right]_x$ and R_{error}. In the end estimated R_{error} and calculated R_{init} has to be multiplied to retrieve new rotation estimation of R (4.3). From Appendix 6 "Multiple View Geometry in Computer Vision"(A6.9.1 (**?**)) using Rodigues parametrization, when angles are small(and noise in initial rotation matrices estimations can be expresed by small angles) rotation matrix and thus R_{error} as well is more or less equals to:

$$R_{error} \cong \begin{bmatrix} 1 & -w_z & w_y \\ w_z & 1 & -w_x \\ -w_y & w_x & 1 \end{bmatrix} \tag{4.7}$$

Such criterium with special matrix design can be used when decomposing G to resolve some ambiguity in choosing proper solution. From standard 4 solution ambiguity with 2 possible rotations and translations, it can bre reduced to 2 possible translation calculations. Described concept is basis of implemented enhanced 8-point and 5-point algorithms

4.2.1 Alternative 3-point algorithm for translation finding

Starting 4.6 it can be written that:

$$x_\prime^T * K^{-T} * \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} * R * K^{-1} * x = 0 \tag{4.8}$$

Having:

$$\begin{array}{rcl} h_\prime^T & = & x_\prime^T * K^{-T} \\ h & = & K^{-1} * x \end{array} \tag{4.9}$$

and

$$\begin{bmatrix} h\prime_1 & h\prime_2 & h\prime_3 \end{bmatrix} * \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} * \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0 \tag{4.10}$$

and multiplying it we will end up with

$$h_1 * h_{\prime 2} * t_z - h_1 * h_{\prime 3} * t_y - h_2 * h_{\prime 1} * t_z + h_2 * h_{\prime 3} * t_x + h_3 * h_{\prime 1} * t_y - h_3 * h_{\prime 2} * t_x = 0 \quad (4.11)$$

what can be grouped:

$$t_x * (h_2 * h_{\prime 3} - h_3 * h_{\prime 2}) + t_y * (h_3 * h_{\prime 1} - h_1 * h_{\prime 3}) + t_z * (h_1 * h_{\prime 2} - h_2 * h_{\prime 1}) = 0 \quad (4.12)$$

rewritten as:

$$\begin{bmatrix} t_x & t_y & t_z \end{bmatrix} * \begin{bmatrix} (h_2 * h_{\prime 3} - h_3 * h_{\prime 2}) \\ (h_3 * h_{\prime 1} - h_1 * h_{\prime 3}) \\ (h_1 * h_{\prime 2} - h_2 * h_{\prime 1}) \end{bmatrix} = 0 \quad (4.13)$$

and solved for instance with SVD with only 3 points. This is very fast way of translation estimation in reconstracted images. However in such situation overall accuracy strictly depends on precise measuerments of camera orientation.

4.3 Pose estimation

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

$$x = P * X \quad (4.14)$$

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

$$P = K * \begin{bmatrix} R \mid t \end{bmatrix} \quad (4.15)$$

4.3.1 Rotation enhancements

Using similar thinking to initial pair reconstruction enhancements it can be noted that:

$$\begin{aligned} x &= K * \begin{bmatrix} Rinit + dR \mid t \end{bmatrix} * X \\ x &= K * \begin{bmatrix} Rinit \mid 0 \end{bmatrix} * X + K * \begin{bmatrix} dR \mid t \end{bmatrix} * X \\ x - K * \begin{bmatrix} Rinit \mid 0 \end{bmatrix} &= K * \begin{bmatrix} dR \mid t \end{bmatrix} * X \end{aligned} \quad (4.16)$$

Substituting $x_m = x - K * \left[Rinit \mid 0\right]$ we get:

$$x_m = K * \left[dR \mid t\right] * X \tag{4.17}$$

This can be solved using normal PNP calculating algorithms. Using rotation as initial solution for pose estimation can be used in order to focus only on rotation error and translation estimation.

4.3.2 Rotation & translation enhancements

Similar to 4.16:

$$\begin{aligned} x &= K * \left[Rinit + dR \mid Tinit + dt\right] * X \\ x &= K * \left[Rinit \mid Tinit\right] * X + K * \left[dR \mid dt\right] * X \\ x - K * \left[Rinit \mid Tinit\right] &= K * \left[dR \mid dt\right] * X \end{aligned} \tag{4.18}$$

end in the end by Substituting $x_n = x - K * \left[Rinit \mid Tinit\right]$ we get:

$$x_n = K * \left[dR \mid dt\right] * X \tag{4.19}$$

This situation can also be solved using normal PNP[TODO reference] calculation algorithm. Using rotation and translation as initial solutions for pose estimation can be used in order to focus only on rotation error and translation error estimation.

4.4 Known rotations & translations

In situation were accurate rotations and translations of cameras are known no aditional pose calculations are needed. Such situation is interesting, because everything needed for corresponding points triangulation is already known. However in mobile sensor data, such measurements are very noisy. However Bundle Adjustment can be used in order to refine measured calculated camera positions and final 3D cloud reconstruction.

4.5 Reconstruction process strategy

Finally all methods described in this section can be combined in different reconstruction strategies. In terms of initialization of 3D cloud point can be made using:

1. **Known rotatations and translations**, which theoritcally allows on up-to-metrical reconstruction

2. **Known rotations**, where translation is estimated with 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

In terms of Pose Estimation following methodologies can be used:

1. **Known rotatation and translations**, where no additional calculations are required and up-to-metrical model can be acquired
2. **Initial rotation**, where relative translation needs to be calculated
3. **Nosy rotation and translation**, where both rotation and translation error needs to be calculated
4. **Unknown external camera parameters**, where standard pose estimation has to be used

Chapter 5

Implementation

This chapter describes choosen implementation environment. This chapter also describes Android application, which was created in order to acquire image sequences with additional sensor data. Sturctures of both Android and Desktop projects are explained and most important implementation details of proposed algorithms and strategies are discussed.

5.1 Choosing Environment

Currently Android has one of the best API, which allows programmers to acquire Sensor fusioned data of rotation and linear acceleration. In order to avoid coding from scratch all epipolar geometry algorithms C++ version of OpenCV library was used. Image acquiering process was seperated from model reconstruction. This allowed to saved a lot of time in debbuging, which is extremly hard in native C++ development on Android. In order to enrich images with sensor data Android app was created and standalone C++ desktop app to perform processing and evaluation.

5.2 Project Structure

Both applications source codes can be found on attached CD and under the Github webpage: TODO `https://github.com/KrzysztofWrobel/MasterThesisSource.git`. In general project was seperated to different subprojects: Android and Desktop. In addition some of sample datasets were added to Dataset folder in order to perform similar to described in next chapter evaluations.

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

In order to capture images and associate them with Sensor data custom photo capture app called "Sensor Enhanced Images Camera" was created. Using this application camera orientation angles can be tracked in real-time in reference to Earth's coordinate system. Also linear acceleration, current velocity and relative translation from last taken picture location can be tracked. When taking picture current camera rotation, relative translation are stored in custom JSON file. Corresponding image is saved along this JSON file in folder, which is created on every app start-up. It allows for taking different dataset captures without mixing them up.

5.3.1 Installation

In order to compile and distribute Android application Gradle based built system was used. This is currently recomended way to maintain Android based projects[TODO ref. google]. Currently this app supports only devices with Android 4.0 and above. In order to compile this project it's recomended to install Android Studio, which already contains necessery SDK and also has built-in Gradle support. More informations about configuration, compilation and install steps can be found in README.md in main catalog of attached source code.

5.3.2 User Interface

As mentioned earlier apps allows to track all necessery external camera parameters in real-time. This is how first version of user interface look like:

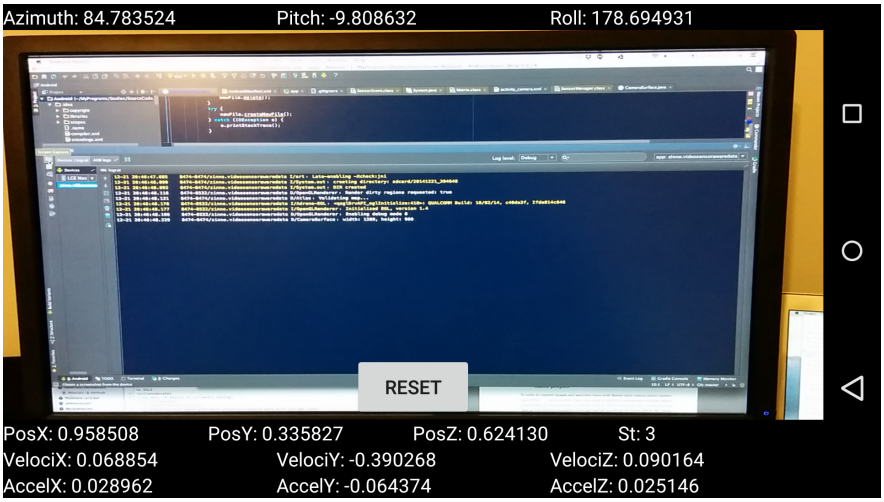
Azimuth: 84.783524
Pitch: -9.808632
Roll: 178.694931
RESET
PosX: 0.958508
PosY: 0.335827
PosZ: 0.624130
St: 3
VelociX: 0.068854
VelociY: -0.390268
VelociZ: 0.090164
AccelX: 0.028962
AccelY: -0.064374
AccelZ: 0.025146

Figure 5.1

In order to capture photo, user only has to click in the screen center. No additional configuration is required. To use use acquired datasets in other program smartphone has to be connected to the computer. Folders are created with actual time suffixes in main catalog of internal SD card.

5.3.3 Important Implementation Aspects

There were two important aspects of Andoid application implementation:

1. Camera rotation calculation
2. Heuristcal movement estimation

5.3.3.1 Rotation calculation

Android SDK has already built-in API, which can be used to access sensor fusioned data. In particular for rotation measurment acquisition $Sensor.TYPE\_ROTATION\_VECTOR$ was used. It returns 9-degree quaternion in reference to geomagnetic north. In order to decompose it to euler angles helper method was used:

```
private float[] euclidanAnglesFromQuaternion(float[] quaternion) {
        ...
        //Converts quaternion to 4x4 rotation matrix
        SensorManager.getRotationMatrixFromVector(rotMat, quaternion);
        ...
        //Extracts euclidan Angles in radians
        SensorManager.getOrientation(rotMat, orientation);
        ...
        return orientation;
}
```

5.3.3.2 Custom heouristic for move estimation

Using currently available Android sensor data it is hard to estimate relative translation of the device. Linear accelaration measurments, which can be accessed with $Sensor.TYPE\_LINEAR\_ACCELERATION$ constant, are very noisy. Such noisy data with double intergration over time results in quite big errors after some time. First integration is required to acquire current velocity change. Second integration over velocity is required to calculate position change in particular moment. However

using human-walking model description it can be improved with following heuristical constraints over calculated velocity:

1. When new linear acceleration sensor data is ready, first apply lowPass filtering in order to reduce some high frequency noise.
2. Change sensor datda vector from local camera coordinates to global reference system(multiply with inverted rotation matrix)
3. Decide depending on current state, if deviceMovmentState 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
   (b) Check current speed with walk constraint(People walk with avarage speed $1.5\frac{m}{s}$) and eventually scale it down to maximum value
   (c) Update device postion

Code snippet of described movement estimation heuristic was derived from Android application and can be found in listing 5.1. Maximum walking speed can be adjusted but by default it should be around $1.5\frac{m}{s}$ TODO `http://en.wikipedia.org/wiki/Preferred_walking_speed`. The most important factor of accurate reconstractin are correletions between movements in X,Y and Z axes. Noise in sensors is equally distributed on each axis, so it estimated transaltion should properly keep these movements correlations.

5.3.3.3 Custom Sensor Data File format

As mentiond earlier each photo sensor information are stored in separe specific file. By default following informations are stored for each captured image:

- **ID**, which indicates order of the photos
- **Path**, which relative path to corresponding image file

```
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
                    avarage 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

- **Euler Angles - pitch, roll, azimuth**

- **Relative position coordinate**

All of this information are stored in a list and when the user is done with taking pictures all informations are saved into JSON file named sensor.txt. Sample file can be found in Additional Materials[TODO].

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

In order to evaluate algorithms there was a need to prepare comfortable project environment, which would allow multiple datasets numerical and visual comparisons. CMake was choosen in order to make simpler build and compilation process. Whole code architecture was developed and tested on Apple MacbookAir with OSX 10.9 Mavericks. In order to compile this project following things need to be installed first:

- CMake 2.8
- OpenCV 2.4.10
- Point Cloud Library (PCL) 1.7
- Boost 1.55
- cvsba (http://www.uco.es/investiga/grupos/ava/node/39) [TODO reference move]

5.4.1 User Interface

There two targets defined in CMake file:

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

5.4.1.1 Test Efficiency

There is no graphical interface for reconstruction parameters configuration. All needed parameters are configured from command-line interface. Following parameters has to be configured for this target:

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

Calculated execution times and execution time are printed to the console output.

5.4.1.2 Test reconstruction

There is no graphical interface for reconstruction parameters configuration either. All neccessary parameters have to be configured from command-line interface. Following variables needs to be configured:

1. **Enhanced Photo Data folder path**
2. **Initial SIFT features number**
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 Bundla Adjustment or not**

As output user gets reconstructed models in *.asc extension. Reconstracted model is also visible in PCL Visualizer integrated into the code. User can navigate through model with mouse and 'F' key, which centers camera view on currently selected point.

5.4.2 Important Implementation Aspects

Most of necessery algorithms were already implemented in OpenCV. Also opensource project with 5-point algorithm implementation was used. It source code can be found here: [TODO reference].

5.4.2.1 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 proper rotation matrix following code inspired on MathWorld Wolfram [TODO http://mathworld.wolfram.com/EulerAngles.html]

```
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;
}
```

5.4.2.2 Enhancing epipolar equations

As we could observe in **??** in order to enhance initial pair reconstruction we can use the same algorithms as standard ones, but with specially conditioned point correspondences. Especially standard 8-point algorithm is already available in OpenCV library. Implemented enhanced 8-point version can be found in 'Multiview.cpp' file, but it main difference from standard implementation is that uses specially modified input points sets, which are conditioned and first one is additionaly 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 relative rotation matrix between 2 cameras. Estimated Fundamental matrix needs to be transformed to essential matrix for further decomposition. In order to choose proper matrix decomposition followin 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 helps to decide properly constrainted rotation error matrix 4.7 and calculate relative translation between cameras.

5.4.2.3 Alternative 3-point translation estimation

When it known or assumed that acquired rotation data are accurate translation can be calculated with 3-point algorithm. Proposed in 4.11 and 4.13 were implemented. As similar to other Epipolar estimations methods implemented in OpenCV, this one also has ability to properly filter out outliers, when RANSAC algorithm is used. Following listing contains this implementation:

```
...
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 * (rotDiff * Kinv).t();
    hg2 = hg2 * (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) * hg1.at<double>(i,
            1) - hg2.at<double>(i, 1) * hg1.at<double>(i, 2));
        A.at<double>(i, 1) = (hg2.at<double>(i, 0) * hg1.at<double>(i,
            2) - hg2.at<double>(i, 2) * hg1.at<double>(i, 0));
        A.at<double>(i, 2) = (hg2.at<double>(i, 1) * hg1.at<double>(i,
            0) - hg2.at<double>(i, 0) * 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;
```

5.4.2.4 Enhancing pose estimation

OpenCv has already rotation and translation enhanced pose estimation implemented. The only thing that has to be done is to perform conversion of euclidan rotation matrix to its Rodrigues representation. Initial rotation and translation needs to be passed as input variables. Pose estimation methods implementations:

```
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.

Chapter 6

Evaluation

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 using visual comparison of reconstructed 3D cloud points.

6.1 Acquiring datasets

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

1. Warsaw University of technology main building(4 images 1024x768pixels)
2. Advertisment Pole ???
3. Warsaw Buisness School Gate and Entrance ??
4. Warsaw Shopping Center Back???
5. Warsaw Shopping Center Front???

Since most of the algorithms proposed in the (TODOreference to Concept) Concept section require internal camera parametrs to be known, the camera used in the course of conducting this study was calibrated and its parameters were stored in "$out_camera_data.yml$" file. All of these datasets can be found in attached CD or Github repository.

6.2 Test Environment

All tests were perfomed 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 Implementation section. Numerical tests which allowed for measuring total errors and execution time were run on "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.

6.3 Testing initial pair reconstrucion methods

The key question to be answered at this point was whether the proposed sensor enhancement gave better results than standard algorithms. The following methods were tested: TODO methods should be already described in Concept and implementation

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

6.3.1 Accuracy - Epipolar lines correspondence

In the case of initial pair images one of the most imporant factors include the epipolar constraint. Using 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 th epurposes of performing dense reconstruction. Sampson Error is one of 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 sum of distance between all points and their corresponding epipolar lines. However, each of proposed methods has different outliers removal capabilities, therefore in order to compare their efficiency the average

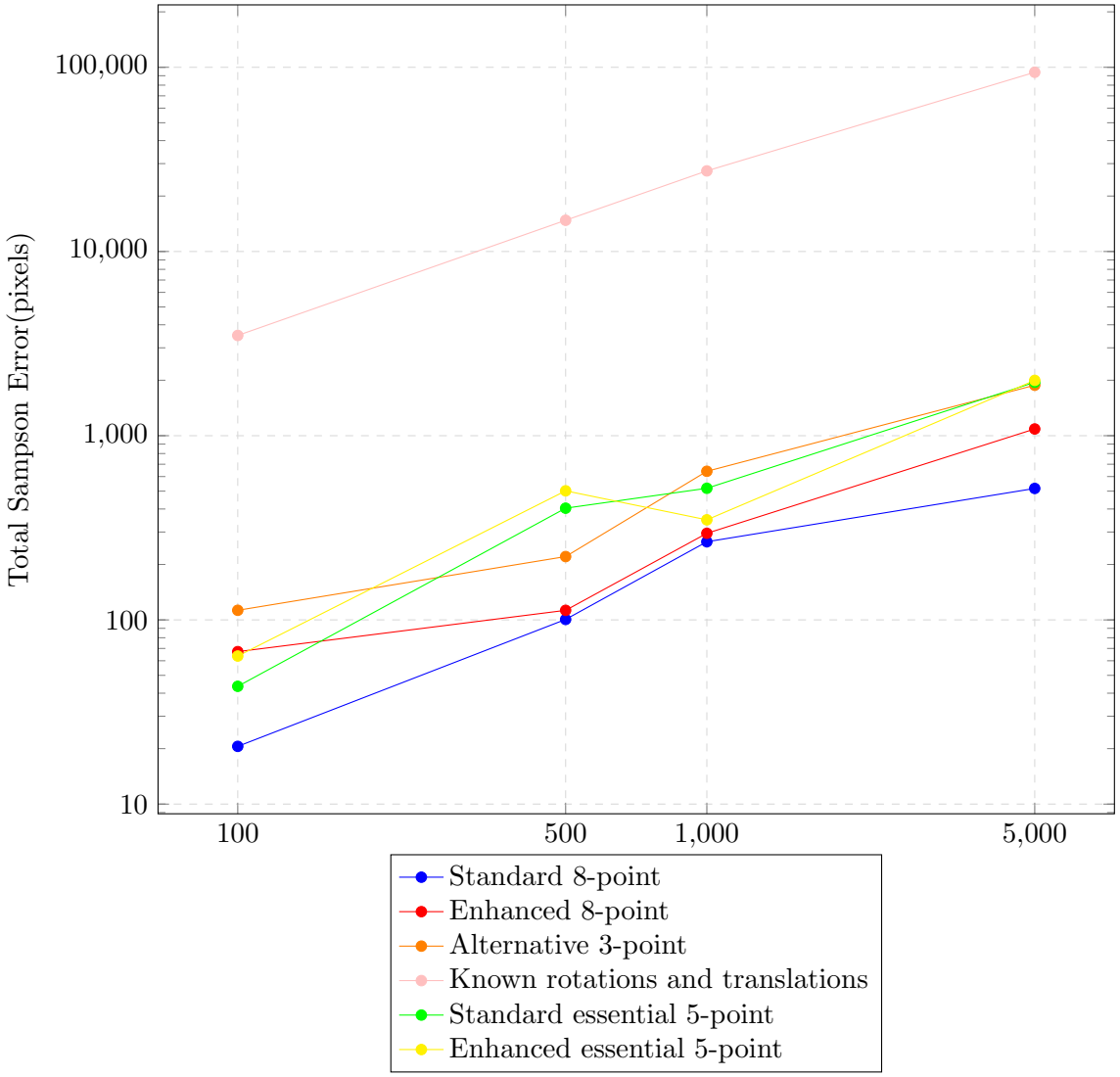

Figure 6.1: Chart showing the sum of Sampson Errors in a picture(1024x768pixels) per various initial Sift features sets sizes

of Samson Error per a corresponding pair was calculated. The following chart 6.1 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 6.2 it can be noted that 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 6.3 - 6.4. It can be seen that both 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 uncompansated 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(6.5 - 6.6). 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 estiamion. However, enhanced versions are always close to optimal solutions, while the standard versions can often fail in terms of fundamental matrix estimation.

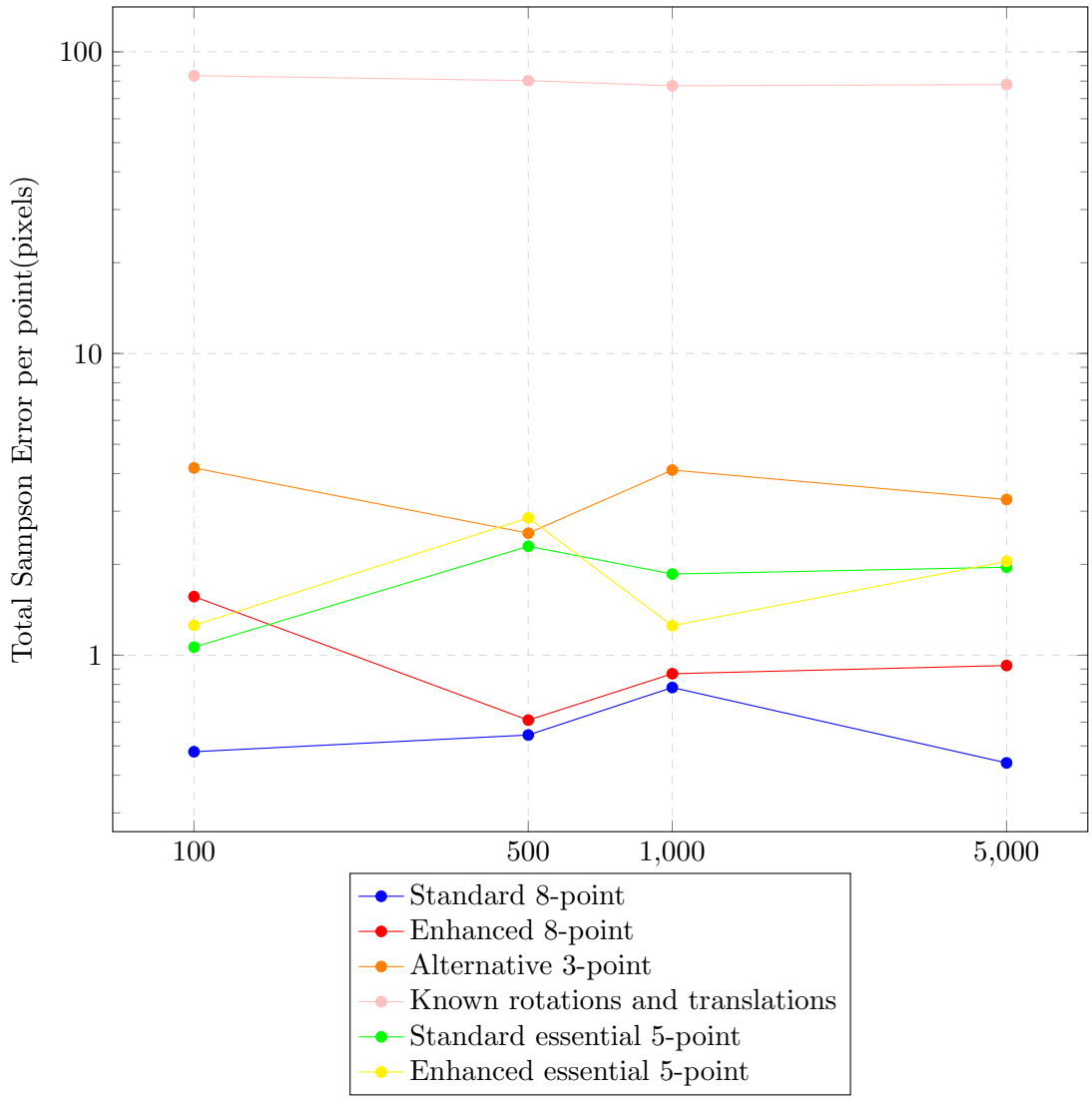

Figure 6.2: Chart showing per point Sampson Error in a picture(1024x768pixels) per various initial Sift features sets sizes

Figure 6.3: The results of drawing estimated epipolar lines on Warsaw Univeristy 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 6.4: The results of drawing estimated epipolar lines on Warsaw Univeristy 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)

Figure 6.5: The results of drawing estimated epipolar lines on Warsaw Univeristy 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 6.6: The results of drawing estimated epipolar lines on Warsaw Univeristy 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)

6.3.2 Time comparison

The chart 6.7 shows the avaraged execution time for 100 estimation attempts. Execution

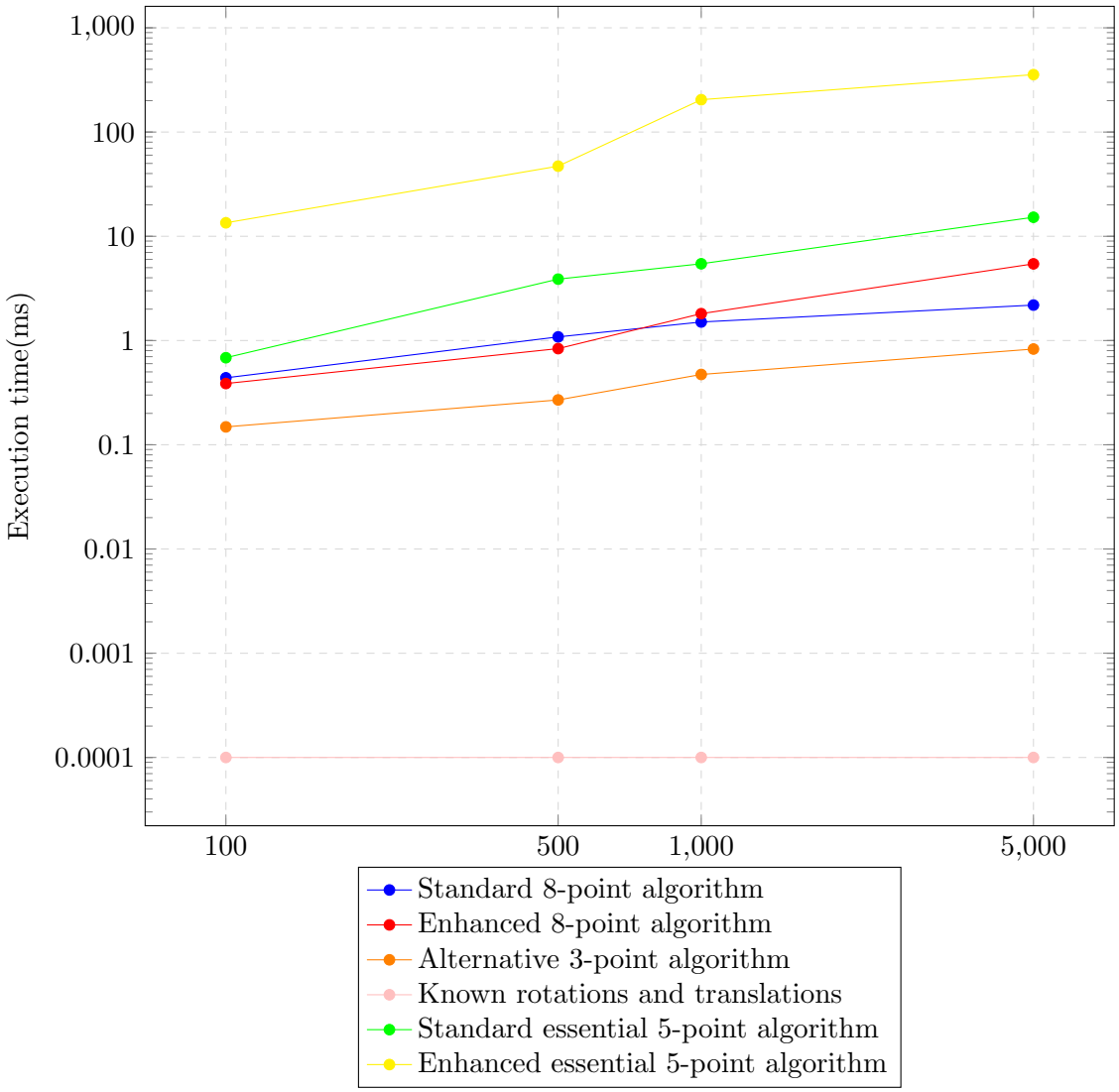

Figure 6.7: Execution time of the proposed algorithms(1024x768pixels) per initial SIFT feature set size

time of 8-point versions are very similar, which is understandable, since their implementation is nearly identical and they differ only in the characteristics of analysed dataset. It can be seen, however, that execution time of 5-point epipolar estimations differs much depending on the version. In this situation either finding optimal solution

with initial rotation is much harder or implementation of these methods is significantly different in terms of memory allocation. Execution time of 3-points algorithm is few times faster than the one of 8-point algorithms. Reconstruction with the sole use of known rotations and translations has a hundred times lower execution time than the standard algorithms, but at the same time it does not produce properly correlated epipolar lines.

6.4 Testing reconstruction strategies

As was shown in 6.3 not every initial reconstruction methods 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 choosen for its consistency in 3D reconstraction. It has never failed to produce solution close to optimal. The third one was prepared in order to check 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 final accuracy too heavily. Finally, the sixth one was proposed in order to check wheter the entire reconstruction process can be performed using solely sensor data.

6.4.1 Accuracy

Accuracy of 3D reconstraction was measured using Bundle Adjustment algorithm from SBA library[ref]. It allows to calculate error based on projective constratin both before and after Bundle Adjustment. The tests performed with different strategies for the "Warsaw Univeristy" dataset are presented on 6.8. The significant differences in initial error can be explained by different numbers of reconstructed points after initial phase

and inconsistent and unknown scale of the finally reconstructed models. Enhanced initial pair reconstruction and pose estimation methods have bigger impact on Bundle Adjustment error reduction.

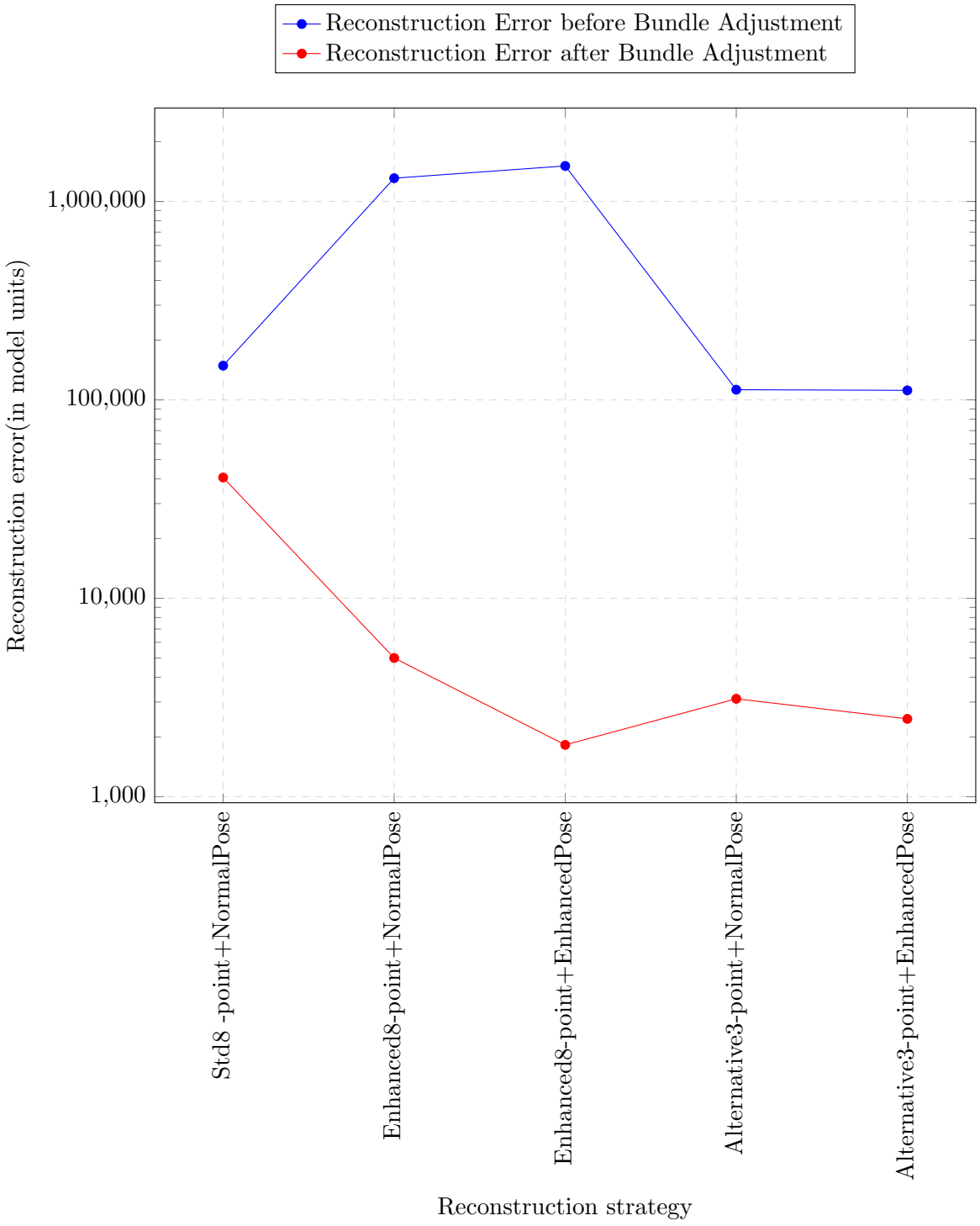

Figure 6.8: Influence of Bundle Adjustment on the models produced with different reconstruction strategies

In general, Bundle Adjustmet process allows to rearrange and modify 3D points positions and change external parameters of estimated cameras. However, in the case of enhanced algorithms 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 enhanced Pose Estimation results in further reduction of BA error when compared to standard the standard strategy.

6.4.2 Execution time

The chart 6.10 presentes the execution time without Bundle Adjustment. Comparing it to execution times in 6.7 one can see that most of the time is allocated for SIFT correspondences matching, which constitutes a bottleneck in the proposed reconstruction methodology [reference to concept pose esitmation choosing]. Furthermore, the reconstruction time was also measured with Bundle Adjustment process at the end(6.11). It was found that Bundle Adjustment works significantly better with enhanced initial pair reconstruction and pose estimation. Cloud points in that case are better organised than the standard reconstructed ones, which results in faster convergence of BA. Sample difference between cloud points before and after BA can be seen in 6.9

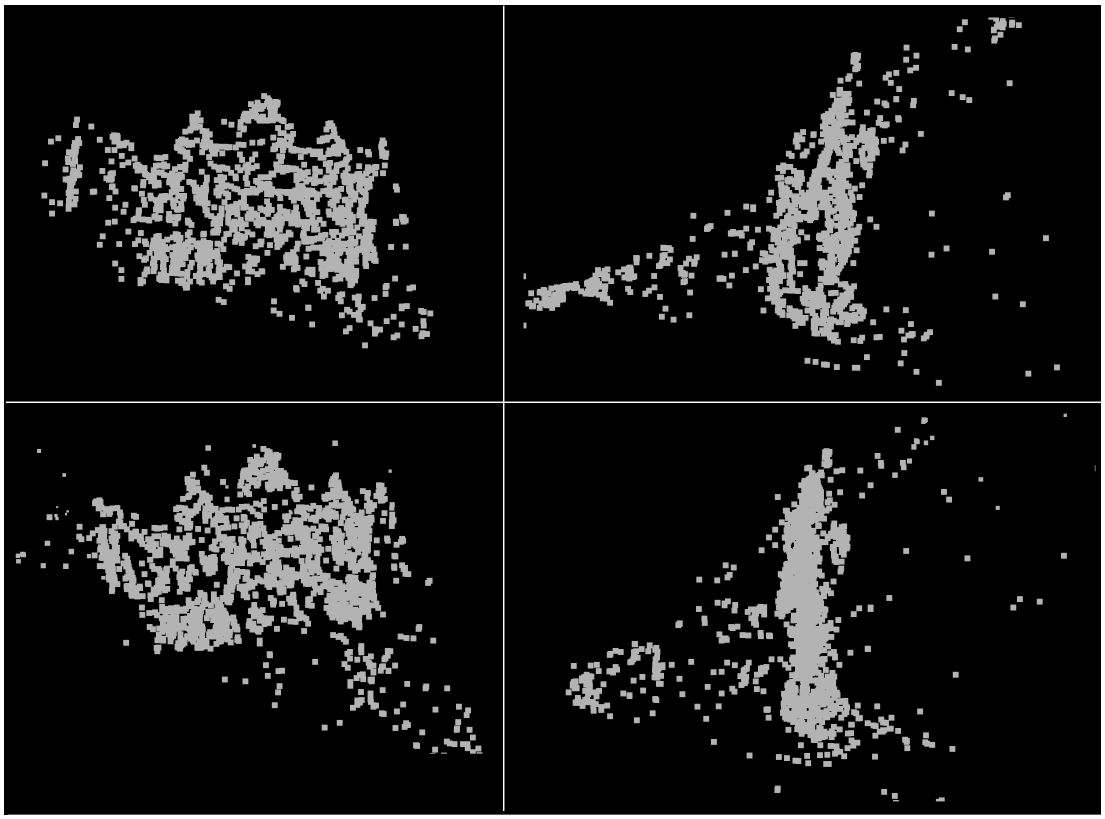

Figure 6.9: 3D point clouds before Bundle Adjustment (upper) and after (bottom) for enhanced 8-point with enhanced pose estimation. Warsaw University of Technology dataset with 1000 SIFT corresponding features (left - front, right - side)

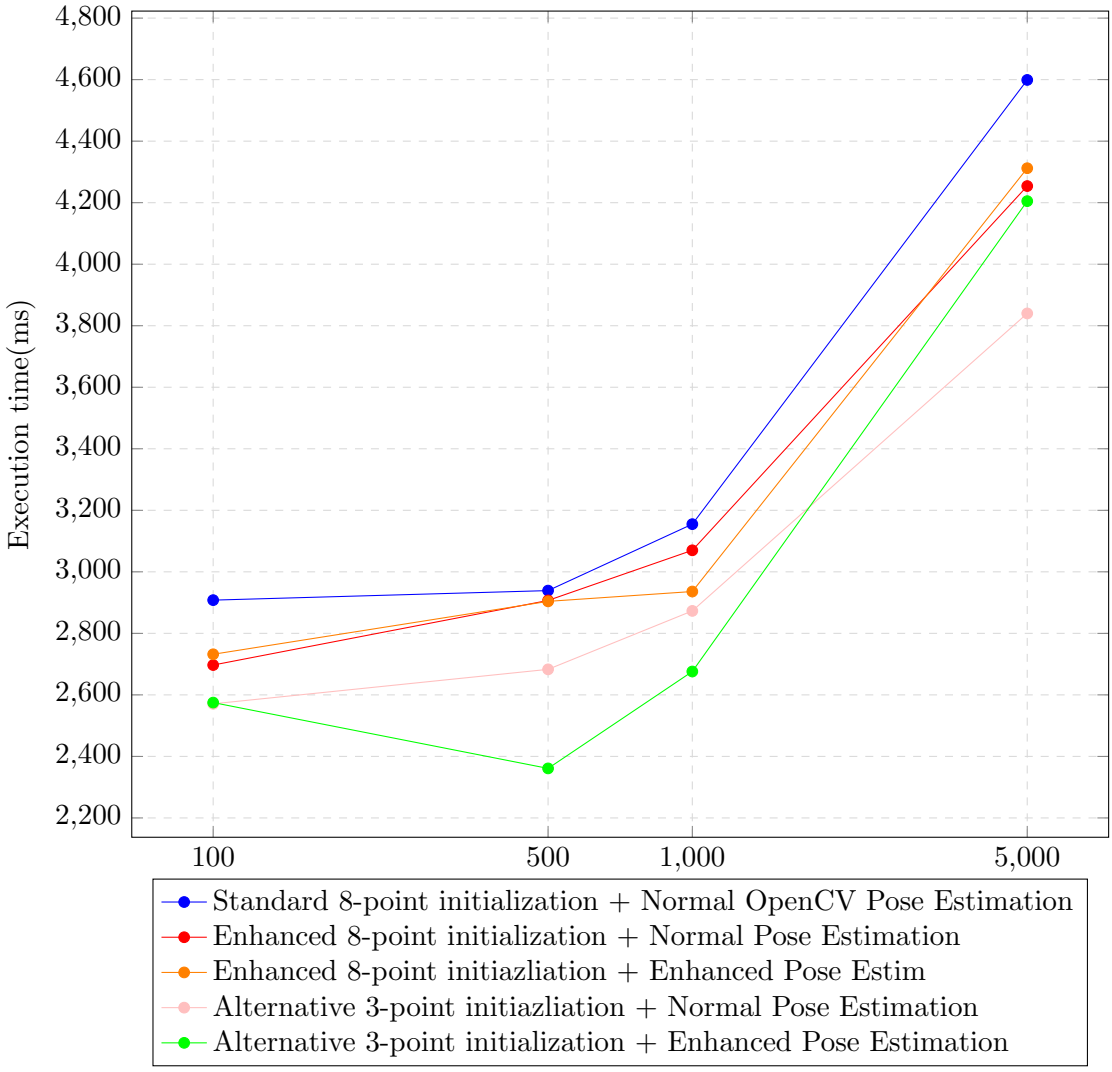

Figure 6.10: Total reconstruction execution time (4 images with resolution 1024x768pixels) per SIFT features set size

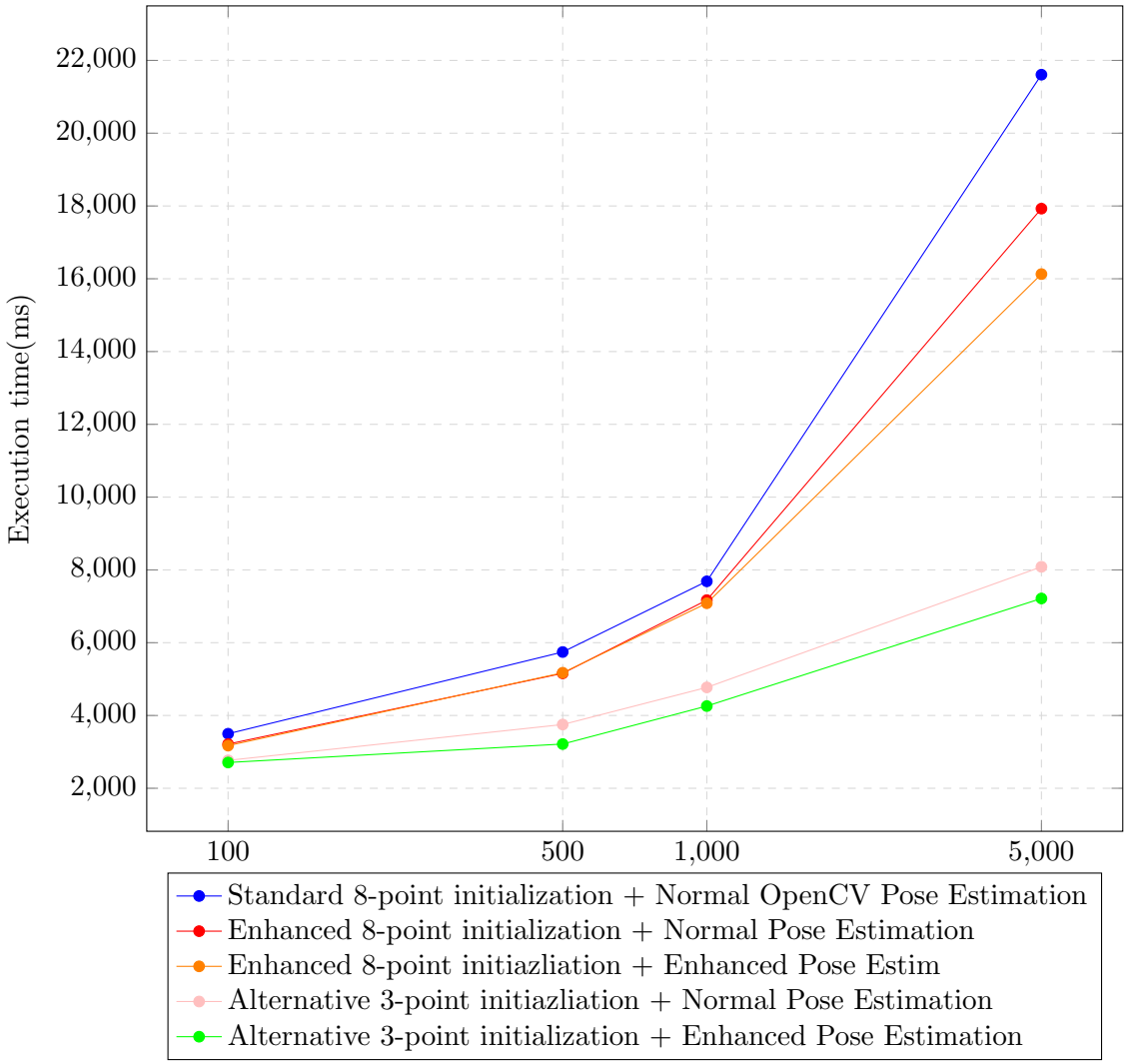

Figure 6.11: Total execution time of reconstruction with Bundle Adjustment (4 images with resolution 1024x768pixels) per SIFT features set size

6.5 Effectivness

The numerical measuerments used do not always correspond to proper 3D model reconstructions. The following four pages present the reconstruction effects of the proposed strategies. Figure 6.12 shows reconstructed effects for different initialization pair methods and 4000 SIFT features set. It can be observed that both standard and enhaced 8-point methods produces very good results, which differ only in terms of final reconstruction scale. Alternative 3-point algorithm produces sligthly worse and distorted models due to uncompenstaed noise in the rotations of the camera used. A model reconstructed from the known rotations and translations is very distorted, but nonetheless stil recognizable. It could prove usefull should a very fast reconstruction be needed.
The matter is slightly different when 400 SIFT feature points are used. [reference] In the first case all algorithms were able to find solutions close to optimal. However, in the second attempt the traingulation test, which is used to identify proper deconmposition in standard 8-point approach, failed and produced an unrecognizable model (6.14).
It can be seen that pose estimations enhancments result primarily in the reduction of outliers (6.15).
Figure 6.16 shows that reconstruction from known rotations and translations produces many outliers.
More reconstructed models can be found in [TODO Materials]

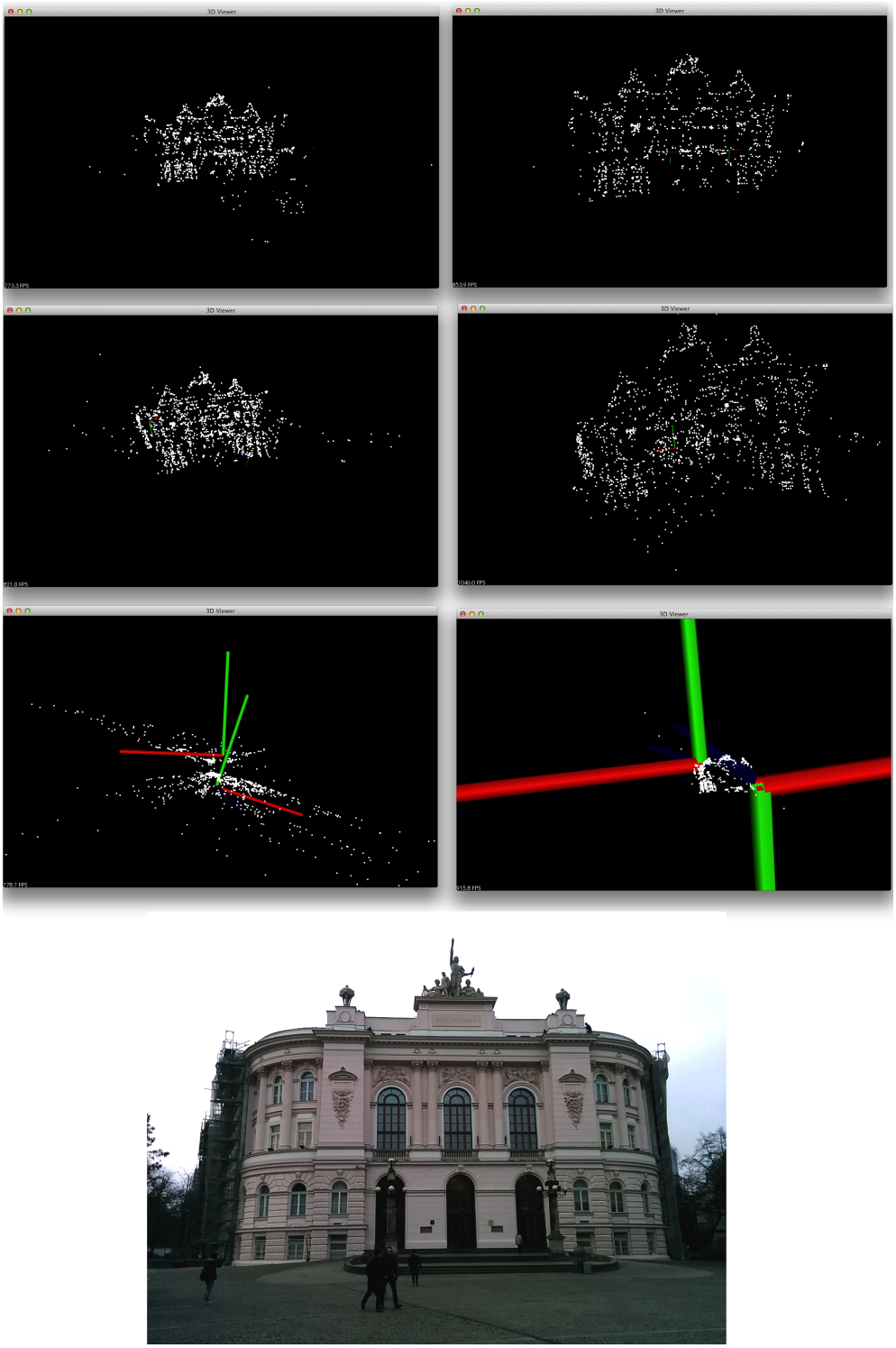

Figure 6.12: Reconstructed models for the proposed initial reconstruction methods and 4000 SIFT features. From upper left to bottom right: 1) standard 8-point, 2) enhanced 8-point, 3) alternative 3-point, 4) known rotations and translations, 5) standard 5-point, 6) enhanced 5-point

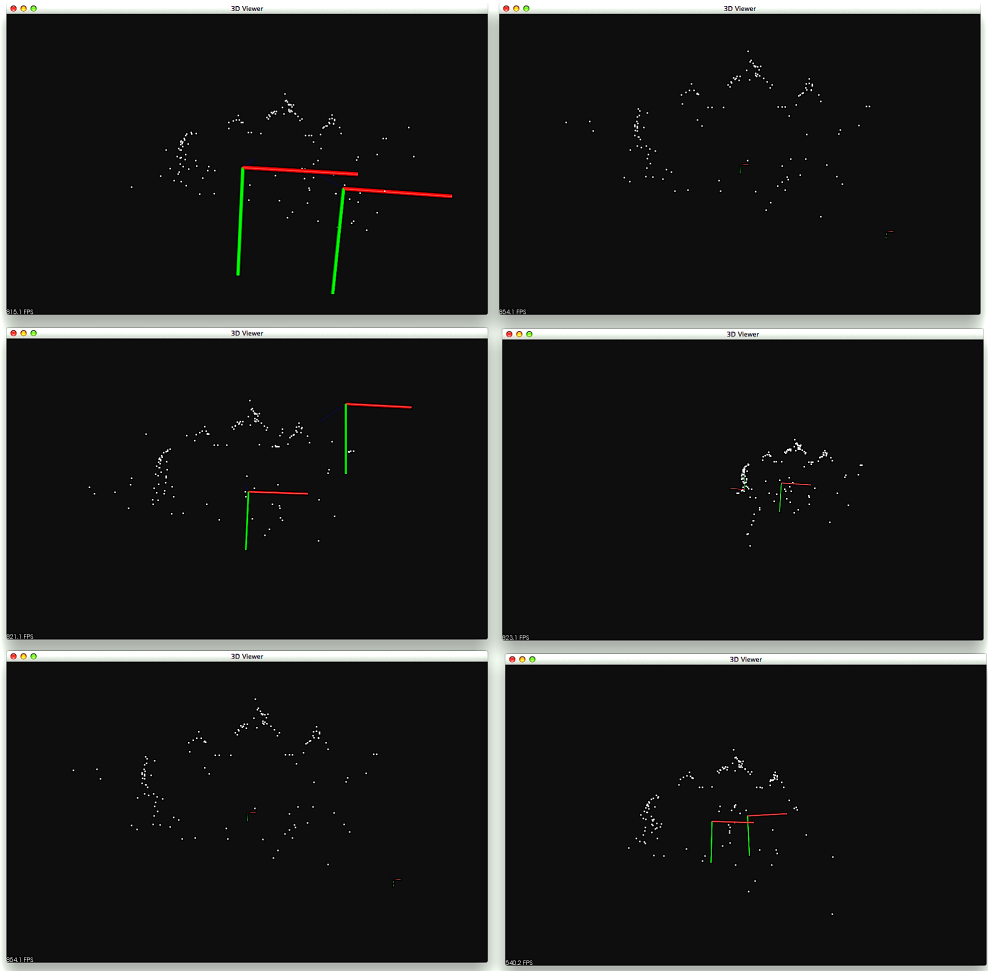

Figure 6.13: Reconstructed models for the proposed initial reconstruction methods and 400 SIFT features. From upper left to bottom right: 1) standard 8-point, 2) enhanced 8-point, 3) alternative 3-point, 4) known rotations and translations, 5) standard 5-point, 6) enhanced 5-point

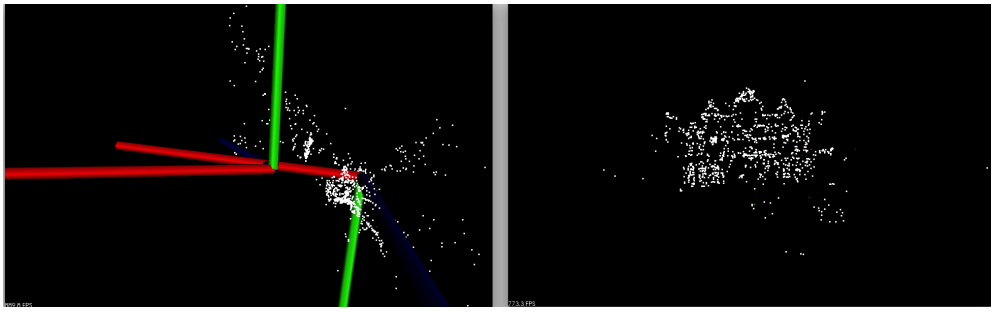

Figure 6.14: Fail test case of Standard 8-point triangulation(left) in comparison to fortunate reconstruction(right)

Figure 6.15: 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 enhancment is applied.

Figure 6.16: 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 7

Conclusion

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

7.1 Summary

The idea of enhancing standard 3D reconstruction algorithms with Android sensor fusion data was conceptualised, implemented and verified in a few different versions. At the beginning, only the alternative 3-point version was planned to be implemented for the purpses 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 data of the rotation from Android sensor fusion. That was when the enhanced 8-point and 5-point versions were designed. In addition, the author proposed different reconstruction strategies, 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. In order to acquire datasets for algorithm, an evaluation Android application named "Sensor Enhanced Images Camera" was developed. Upon capturing the picture it automaticaly stores the current device's rotation information and proposed heuristcally 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. The proposed enhancements to the standard 8-point algorithm allow to unambiguously calculate the proper rotation and translation. 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 the standard ones'. The use of initial rotation and translation estimation in pose estimation results in a greater reconstruction accuracy, particularly in terms of outliers removal and Bundle Adjustment convergence speed. In general, applying Bundle Adjustment of sensor enhanced reconstruction results in greater error reduction and shorter execution time in comparison to the refining standard ones. The major problem, a bottleneck of some sort, of the proposed reconstruction process was the time needed for matching corresponding items. The study showed that using the sensor data only is not enough to create accurate 3D models. However, it is possible to find recognizable model among the reconstructed 3D cloud of points. The only problem is how to distuinguish a recognizable model from the one reconstructed from uncorrect 3D points. It turned out that estimating rotation error matrix (R_{error}) is quite accurate and useful for that purpose and the proposed Rodrigues decomposition and its rounding, such as diagonal of R_{error} consisting of a diagonal identity matrix, returns better accuracy of 3D reconstruction and unambigously defines the proper camera decomposition. Heuristical movement estimation is not entirely accurate and does not have significant impact on the reconstruction process. Finally, the proposed 3-point algorithm for translation estimation allows for faster and quite accurate recreation of the structure.

7.2 Dissemination

So far no one has used the implemented applications. Nonetheless an Android app 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 properly tested. "Enhanced 3D reconstructer" has been already published to GitHub as open-source project distrubuted on LGPL license [reference].

7.3 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 get rotation quaternion and a way to decompose it, but it does not explain how it is calculated. Decomposition of rotation matrix to euclidan 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 insted of homography merging was not the optimal solution. Relaying on the pose estimation produced very small amount of points and sometimes reconstruction stoped only after analysing merely a few images.

7.4 Future work

Firstly, it would be useful to establish how the homography merging approach would influence an 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. 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 (similar to (**?**)).

Chapter 8

Materials & methods

```
[
    {
        "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 8.1: Efficeincy table of proposed methods for 100 SIFT features in Warsaw Univeristy of technology dataset. Colums: Total Sampson Error, Avarage 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 8.2: Efficeincy table of proposed methods for 500 SIFT features in Warsaw Univeristy of technology dataset. Colums: Total Sampson Error, Avarage 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 8.3: Efficeincy table of proposed methods for 1000 SIFT features in Warsaw Univeristy of technology dataset. Colums: Total Sampson Error, Avarage 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 8.4: Efficeincy table of proposed methods for 5000 SIFT features in Warsaw Univeristy of technology dataset. Colums: Total Sampson Error, Avarage Sampson error per point, Amount of points left after outliers removal, Execution time

References

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted from XXX to YYY under the supervision of PI at ZZZ.

CITY,