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.

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 basic theory behind 3D reconstructiona and Structure from Motion. All of this informations can be found in (**?**). Moreover it makes brief overview of Accelerometer, Gyroscope, Gyroscope and their possible fusion[referemces].

2.1 3-D reconstruction in general

Today many devices are capable of 3D reconstruction. Most popular Kinect(**?**) is one of them. It is able to perform real-time 3D cloud point generation. However is very special case, which uses 2 camera: RGB Camera and IR Camera. It has very high accuracy of reconstruction, but in the same time it's very expansive and not exactly mobile. There are possibilities of reconstruction using only single camera either from video or from still image sequence. One need only two images in order to perform 3D model generation. Such reconstraction process consists of few basic steps:

1. Image Acquistion
2. Feature extraction and correspondences matching
3. Fundamental & Essential Matrices
4. Camera parameters estimation
5. Triangulation

2.1.1 Feature extraction and correspondence matching

Each image has to be analysed, interesting features have to be extracted and later used to find correspondences. There are multiple features detector and extractors available to use $\texttt{http://en.wikipedia.org/wiki/Feature_detection_\%28computer_vision\%29}$. Some of them are better for edge detection, where others are better for corner or blob detection. One of the most and robust feature detection method is SIFT`http://en.wikipedia.org/wiki/Scale-invariant_feature_transform`.
Once features are detected and descripted they also have to be matched. Once again there are many approaches to matching process `http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html`. Usually many of this matches are not proper. Such points are called outliers and they highly influence other reconstruction steps.

2.1.2 Fundamental & Essential Matrix estimations

Once proper matches are decided, it can be proven that there exsists Fundamental matrix F for which such equation:

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

$xandx'$ are uncalibrated points in the images. The above relation which defines the fundamental matrix was published in 1992 by both Faugeras and Hartley. It is noted that solving such equation is highly sensetive to outliers. To make estimation more robost to outliers RANdom SAmple Consensus (Bundle Adjustment (RANSAC)) `http://en.wikipedia.org/wiki/RANSAC` can be used. It basic idea relies on choosing randomly subset of However these points can be calibrated, when internal camera parameters, such as focal lengths and principal point are known. These parameters are expressed by internal camera matrix usually noted as K:

$$\begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{2.2}$$

where x = fmx and y = fmy represent the focal length of the camera in terms of pixel dimensions in the x and y direction respectively. Similarly, x 0 = (x0,y0) is the

principal point in terms of pixel dimensions. Using calibrated cameras points, similar to 2.1 equation for essential matrix E can be written:

$$x_c^{'T} * E * x_c = 0 \tag{2.3}$$

This results in another equality:

$$E = K^{'T} * F * K \tag{2.4}$$

$KandK^{'}$ being the internal calibration matrices of the two images involved.

2.1.3 Camera parameters estimations

Acquiring of internal camera parameters can be done with usage of special markers and needs to be done only once per camera model. What's left in terms of propoer 3D reconstruction is external camera parameters, such as rotation(orientation angles of the camera) and global postition. Usually in terms of 3D reconstraction these parametrs are not known. However in Chapter 9 of Multiple View Geometry in Computer Vision((**?**)) autors show, how essential matrix can be decomposed using Singular Value Decomposition Singular Value Decomposition (SVD) to extract proper relative camera positioning. Using SVD and assuming that one of camera projection matrices is equal to:

$$P1 = K * \left[I \mid 0\right] \tag{2.5}$$

second one is equall to

$$P2 = K * \left[RDiff \mid tDiff\right] \tag{2.6}$$

Unfortunately there are 4 possible solutions in such decomposition and it's not always possible to identify correct one. For each one triangulation test can be performed and solution with most points in front of the camera is probably the correct one.

2.1.4 Points Triangulation

Once internal and external(global or relative) parameters triangulation of points can be performed in order to acquire up to affine reconstruction(only scale of model is not known) model. In [TODO hartley chapter 10] this process is described in details, but both camera projection matrices has to be calculated. [TODO paste figures]

2.2 Structure from Motion

Term of Structure from Motion(SfM) refers to structure reconstraction from consecutive sequences of moving camera. It is really popular research topic and two main different approaches called Pose Estimation and Homography Estimation can be used to make it happen.

2.2.1 3D Pose Estimation

Assuming that some 3D cloud point is already known, correspondences between 2D features in new image and 3D point clouds positions can be established. Such 3D-2D correspondences can be used to estimate next camera relative to model position. This allows of reconstraction of new 3D points and merging them smoothly into existing model. Unfortunately this process is also highly sensitive to presence of outliers, so adequate measeures has to be made in order to reduce their influence. One of the main advanteges this method is speed, but on the other hand it's effectiveness strongly relies on exising 3D cloud quality.

2.2.2 Homography estimation

New image can be reconstructed with previous once in a standard way to recive up to scale 3D model. Later freshly acquired model can be merged to existing one using Homography estimation between coresponding 3D points. Such strategy is slowet than previous, but it is not influenced by quality of existing 3D model. It's also higly sensitive to outliers, but once made accurately produces much more new 3D points.

2.2.3 Structure Adjustment

After some time of SfM reconstruction some techniques, which compensate increasing error of wrong estimates propagating through images special algorithms can be used to refine reconstructed models. One of such techniques is Bundle AdjustmentBundle Adjustment (BA) `http://en.wikipedia.org/wiki/Bundle_adjustment`. Through finding point correspondances between multiple sets of images algorithm iterativly modifies either both of one camera external paramaters as well as 3D points positions. It's main disadvantages is execution times. It takes a lot of time to use it in real-time applications.

2.3 Mobile Sensors overview

2.3.1 Accelerometer

An accelerometer is a device that measures the proper acceleration of the device. This is not necessarily the same as the coordinate acceleration (change of velocity of the device in space), but is rather associated with the phenomenon of weight experienced by a test mass that resides in the frame of reference of the accelerometer device. Generally, accelerometers measures acceleration by sensing how much a mass presses on something when a force acts on it. An accelerometer at rest relative to the Earth's surface will indicate approximately 1 G upwards, because any point on the Earth's surface is accelerating upwards relative to the local inertial frame (the frame of a freely falling object near the surface). To obtain the acceleration due to motion with respect to the Earth, this "gravity offset" must be subtracted [18].

2.3.2 Gyroscope

Generally, a gyroscope is a device for measuring or maintaining orientation, based on the principles of conservation of angular momentum [17]. A conventional (mechanical) gyroscope consists of a spinning wheel mounted on two gimbals which allow it to rotate in all three axes. An effect of the conservation of angular momentum is that the spinning wheel will resist changes in orientation. Hence when a mechanical gyroscope is subjected to a rotation, the wheel will remain at a constant global orientation and the angles between adjacent gimbals will change. A conventional gyroscope measures orientation, in contrast to MEMS (Micro Electro-Mechanical System) types, which measure angular rate, and are therefore called rate-gyros [4]. MEMS gyroscopes contain vibrating elements to measure the Coriolis effect. A single mass is driven to vibrate along a drive axis, when the gyroscope is rotated a secondary vibration is induced along the perpendicular sense axis due to the Coriolis force. The angular velocity can be calculated by measuring this secondary rotation. The Coriolis effect states that in a frame of reference rotating at angular velocity , a mass m moving with velocity v experiences a force [4]:(2-1) An important note to be made is 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. Magnetometers can be divided into two basic types: scalar magnetometers that measure the total strength of the magnetic field to which they are subjected, and vector magnetometers (the type used in this project), which have the capability to measure the component of the magnetic field in a particular direction, relative to the spatial orientation of the device [19].

2.3.4 Sensor Fusion

Sensor fusion is the combining of sensory data derived from sensory data from disparate sources such that the resulting information is in some sense better than would be possible when these sources were used individually. The term better in this case can 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) [11].

Chapter 3

Related Work

There are many approaches to the problems, from raw one by one pixel analysis to high level abstraction of objects, light and shadows estimation[some references to each].However this thesis does not focus on high-level abstraction reconstruction, but focuses on refining relative poses estimation steps. In order to estimates especially first two relative positions of cameras essential matrix must be decomposed. Since basic epipolar geometry equation many scientist introduced a way to solve this linear problems, where the main differences were in terms of accuracy and speed. One of the first was 8-point algorithm[reference], which can be used to compute a fundamental matrix. This is done without any prior knowledge of the scene, as well as camera parameters. Still to find relative position knowledge of internal camera parameters is needed in order to calculate and decompose Essential Matrix. Later on 5-point algorithm approaches were introduced[references], which needed prior knowledge of internal camera parameters. David Nister in his paper shows that 5-point outperforms almost all similar algorithms in terms of accuracy and speed. Only 8-point algorithm can be competitive, when it comes to forward image sequences. One of the state-of-art real-time and roboust approaches is iterative 5-pt Algorithm created by Vincent Lui and Tom Drummond[].

Most of algorithms are very sensitive to presence of outliers. One of the most common approach is to use of RANSAC modelling[refining estimates], where iteratively subset of data is chosen to find a solution and then other points are checked, if they also satisfy equation with calculated solution.

3. RELATED WORK

Research group from Technishe Univeristt Berlin made an comparison and evaluation of methods, which were published at that point. It turned out that estimation of camera rotation is much more stable than translation. Also there are a lot of ambiguities in terms of choosing the correct solution of epipolar geometry equation.

In certain situation 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[reference], which shows how to describe and use model of camera mounted on a car to enhance 3D reconstruction. Introduced in 2013 4- point algorithm, which uses information of rotation angle in certain axis from additional sensor as show in paper[reference] can outperform even some versions 5-point algorithm. Lately scientists are creating more complex models to estimates relative stereometry. For instance group from Zurich proposed a way to enhance reconstruction with additional 6DOF sensor [Robust Real-Time Visual Odometry with a Single Camera and an IMU].

There are also different approaches like [Line-Based Relative Pose Estimation], where it's shown how to estimates the relative pose from 3 lines with two of the lines parallel and orthogonal to the third. Very accurate estimations also can be achieved when there is no camera rotation[Epipole Estimation under Pure Camera Translation*]. All these references shows that enhanced models help to achieve often faster more accurate solution.

Accuracy and speed are very important, when it comes to create systems capable of augmenting our reality. One of first successful systems for such situations were proposed by research group[PTAM]. They showed how two simultaneously working threads can be used to both create model of environment and use this knowledge to apply graphical effects to objects presented on stream camera video. Also some of these concepts where applied already even to robotic vision. Authors showed efficiency of proposed system for robot walking in cluttered indoor workspaces[MonoSLAM: Real-Time Single Camera SLAM].

The most important things, which can be concluded are rotation estimation is more stable than translation estimation and the more well described model of setup.

Chapter 4

Concept

Here general concept will be described without details about descriptors used and so on... 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. It also helps to achive faster, more stable and robust algorithms. This thesis will show how prior knowledge of rotation or translation can be used to faster process 3D reconstruction of series of images. However there are many algorithms, which relay on accuracy of additional rotation and translation data. In reality especially, when it comes to hand-held smartphones, collected data are very noisy. That's why this thesis also proposes enhancements of most popular algorytmic approaches, when noisy data are used.

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 optional and as discussed in 2.5??? has its up and downsides) capable of storing pictures and sensor data can be used. During algorithm runtime either both rotation and translation informations are used or just rotation, which as indicated in ....??? is less noisy than translation estimation. Internal camera parameters need to be calculated before reconstruction process is began. Additional sensor data can be unaccurate and noisy.

4.2 Reconstruction process strategy

Both Epipolar equations and Pose Estimation improvments can be used in different configurations. It's expected that some of them are more accurate and some are faster. It was necessery to evaluate this approaches both in term of speed and accuracy. Author of this thesis proposes environment, where user can decide what type of stargy she/he wants to use. In terms of initialization of 3D cloud point reconstraction can be made using:

1. Known rotatations and translations - relative poses are calculated from sensor data - euclidan reconstruction
2. Known rotations - Alternative 3-point algorithm for translation finding or enhanced fundamental/essential
3. Noisy rotation - enhanced fundamental/essential for dR and translation
4. No known extrisinc parameters - standard fundamental/essential for R and Translation

Later one new images are analized in comparison to previous one to enrich initial cloud point with new points. It's only necessery to keep track of each 3D cloud point corresponding 2D positions in images. In terms of Pose Estimation following methodologies can be used:

1. Known rotatation and translations - no additional calculations, just triangulation - euclidan reconstruction
2. Known rotation - looking for relative translation by backProjection optimisations
3. Noisy rotation - looking for dR and translation by backProjection optimisations
4. No known extrisinc parameters - standard pose estimation

Of cource in all of this steps it's really important to have as minimum outliers as possible. All of this methods are pretty good in terms of removing outliers, espiecially when connected with RANSAC algorithm.

4.3 Enhancing epipolar geometry equation

4.3.1 Rotation enhancements

Taking standard fundamental geometry equation and relative camera based system ($P = \left[I|0\right], P' = \left[R|t\right]$) we can note that:

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

It's also good to note 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 both rotation and translation can be distorted with noise. This can be written as:

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

where R_{init} is rotation matrix from measured angles and R_{error} is rotation matrix of angles errors. 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 correcting noise error rotation matrix. Basic idea of algorithm proposed in this thesis is instead of R calculation, which as described in [reference] can be acquired from Essential matrix SVD decomposition, R_{error} will be estimated. In the end 4.1 can be rewritten in form:

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

Having:

$$\begin{array}{lcl} h_{\prime}^{T} & = & x_{\prime}^{T} * K^{-T} \\ h & = & R_{init} * K^{-1} * x \\ G & = & \left[T\right]_{x} * R_{error} \end{array} \tag{4.5}$$

With such notation one can notice that

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

quite resembles both standard fundamental and essential equation [reference]. Of cource $h\prime$ and h are both homogenous. From analysis it's known that G has 6DOF: 3 due to unknown translation and 3 due to unknown correction angles. 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 order to resolve some common problems in retrieving correct solution and improve accuracy. In the end estimated R_{error} and calculated R_{init} can be multiplied to retrieve new rotation R (4.3). Also to reduce error estimation using Rodigues parametrization, when the angles are small(and they indeed are because only small error is present in sensor data) we can note that R_{error} in fact is more or less equals to:

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

[Refernece to Hartley]. It can be used when decomposing G to ensure proper decomposition. What's more in normal situation in case of both these algorithms we have to decompose recovered matrix to both relative rotation and translation. However we normally have to deal with small ambiguity and check for instance by initial triangulation, which R and T use. Using this Rodriguez representation, we can reduce 4 solution test cases to 2 possibly solutions test-case.

4.3.2 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:

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

$$\begin{bmatrix} h_{\prime 1} & h_{\prime 2} & h_{\prime 3} \end{bmatrix} * \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} * \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0 \tag{4.10}$$

and multiplying it we will end up with

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

what can be grouped:

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

rewritten as:

$$\begin{bmatrix} t_x & t_y & t_z \end{bmatrix} * \begin{bmatrix} (h_2 * h\prime_3 - h_3 * h\prime_2) \\ (h_3 * h\prime_1 - h_1 * h\prime_3) \\ (h_1 * h\prime_2 - h_2 * h\prime_1) \end{bmatrix} = 0 \tag{4.13}$$

and solved for instance with SVD with only 3 points. This is very attractive way of reconstracting images from only 3 points especially in terms of speed. Overall accuracy strictly relies on precise measuerments of camera orientation.

4.3.3 Translation enhancements

It's known that without any additional informations about photographed scene or exact translation of camera scale cannot be retrived and only affine reconstruction can be done. As descriped in [ref] using SensorFusion linear acceleration of the capable camera can be calculated. Combining it with certain robust heuristic for movement estimation and low-pass filter relative/global translation of the camera can also be estimated. This information can be used to perform Euclidan reconstruction. Calculated T from G decomposition(**??**) can be combined with T_{global} acquired by double integration of linear acceleration. Due to double integration error can be big, but in general it's almost the same in every direction. It may not be perfect, but still it can help to estimate range of euclidian reconstruction.

4.4 Pose estimation

Different approaches to continous multiview 3D reconstruction were discussed. This research main goal was to make it more accurate and faster. That is way it focuses on pose estimation instead of homographies merging. This is also good in terms of keeping scale in between next image analysis. For any point in image following condition is kept:

$$x = P * X \tag{4.14}$$

where x is homogenous image point (x , y , 1) and X homogenous 3D point (X , Y , Z , 1). Of course

$$P = K * \begin{bmatrix} R \mid t \end{bmatrix} \tag{4.15}$$

4.4.1 Known rotations & translations

In situation were rotations and translations of cameras are known no aditional calculations are needed. Both rotation and translation of the camera can be estimated from sensors. Such situation is interesting, because there is already everything needed for triangulation of points. To resolve inaccuracy of especially translation measurements standard Bundle Adjustment methods can be used in order to refine reconstruction results.

4.4.2 Rotation enhancements

Using similar thinking as in previous section[ref] we can note that:

$$\begin{aligned} x &= K * \left[Rinit + dR \mid t\right] * X \\ x &= K * \left[Rinit \mid 0\right] * X + K * \left[dR \mid t\right] * X \\ x - K * \left[Rinit \mid 0\right] &= K * \left[dR \mid t\right] * X \end{aligned} \tag{4.16}$$

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. It's known that orientation estimation is more accurate than translation estimation using SensorFusion and this approach allows very accurate calculation of rotation error matrix dR and translation t, which keeps the scale of initially reconstructed points cloud.

4.4.3 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:

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

We can note that when initial solution is already known, the problems begins to focus on refining pose estimation instead of searching for it.

Chapter 5

Implementation

In order to perform some tests with proposed methodology there was a need to prepare enviornment for it.

5.1 Choosing Environment

One thing was to prepare application, which would be capable of taking both pictures with additional SensorFusioned data. Currently Android has one of the best APIs which allows user to get both raw and fusioned sensor data. In order to avoid coding from scratch all sterometry algorithms there was need to choose library, which is fast and has most of discussed algorithms implemented. This is where OpenCV performs really good. However it's really hard to get it running on Android, due to long time compiling, problematic error debbuging and speed performance. This is why author decided to seperate process of acquiring images from processing them. 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 can be found on attached CD and under the Github webpage: `https://github.com/KrzysztofWrobel/MasterThesisSource.git`. In general project was seperated to different subprojects: Android and Desktop. In addition some sample datasets were added to Dataset folder.

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 "..." were created. User can track in real time parameters euclidan angles of smartphone in global earth coordinate system. When taking picture current roation data as well relative translation from last capture are stored in custom JSON file, which is later saved along taken picture in seperate folder. All of captured pictures and sensor data can be compressed and send though internet to the user. By default they are saved on internal SD card in folder, which is automaticly created timestamp folder.

5.3.1 Installation

Gradle based built system was used, which is currently recomended way to handle Android based projectsref. google. Currently this app supports devices with Android 4.0 and above. In order to compile this project it's recomended to install Android Studio. It already contains Android 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.

5.3.2 User Interface

ScreenShot. Apps allows to track camera angles in real time, so user can precisly decide angles of capture. Also heuristical movement and translation estimation can be seen. In order to capture photo, user only has to click in the screen center.

5.3.3 Important Implementation Aspects

Only master thesis specific code is described in detail.

5.3.3.1 Rotation calculation

Android SDK has already API, which can be used to access noth raw and "sensor fusioned" data. In particular for rotation estimation $Sensor.TYPE\_ROTATION\_VECTOR$ was used. It returns 9DOF quaternion in reference to geomagnetic north. In order to decompose it to euler angles helper method were 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

Unfortunately using only accessible sensor data it's hard to estimate relative translation of the device. Accelerometer, even compensated Android $Sensor.TYPE\_LINEAR\_ACCELERATIO$ measures only linear acceleration of the device. These measures, which are very noisy itself, not only due to for instance hands shaking, need to be double integrated in time in order to get move distance. As proposed system is supposed to be used for hand-held cameras on order to minimise noise influence following enhanced heuristic for people walk model was proposed:

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

Described algorithm implementation snippet can be found in listing 5.1. Human walking model was introduced in order to constarint the velocity. Without it integrating small noise over time, would eventually results in high unrealistic movements. Maximum walking speed can be adjusted but be default it should be around $1.5\frac{m}{s}$ `http://en.wikipedia.org/wiki/Preferred_walking_speed`. In fact the most important factor of good position estimation is vector angle, fortunately noise equally influence each axes, so it should be statisticly correct.

5.3.3.3 Custom Sensor Data File format

As mentiond each photo capture stores additional sensor information. These informations are:

- id - indicates order of photos
- relative path to image file
- Euler Angles - pitch, roll, azimuth
- Position coordinates in geomagnetic north system(relative to previous image)
- Android rotation quaternion

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 atachments.

```
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

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

In order to evaluate algorithms there was need to prepare project environment suitable for this task. Discussion of choosing motivations, where done in 5.2. Author choosed CMake in order to make simpler build and compilation process. Whole project, where developed and tested on 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)

5.4.1 User Interface

There two targets defined in CMake file:

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

5.4.1.1 Test Efficiency

There is no graphical interface for reconstruction parameters configuration. However at the beginning there are few command-line inputs required to continue reconstruction. User needs to configure:

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

Calculated parameters are printed to Console Window. TODO What can be seen there? Screenshot from console output.

5.4.1.2 Test reconstruction

There is no graphical interface for reconstruction parameters configuration. However at the beginning there are few command-line inputs required to continue reconstruction. User needs to configure:

1. Enhanced Photo Data folder path
2. SIFT features number
3. Init pair reconstruct method:
    - Standard Fundamental equation based
    - Enhanced Fundamental equation based
    - Standard essential equation based
    - Enhanced essential equation based
    - 3-point Translation estimation
    - None - use existing rotations and translations
4. Pose estimation method:
    - Standard 3d-2d perspective estimation
    - Enhanced 3d-2d perspective estimation with noisy rotation
    - Enhanced 3d-2d perspective estimation with noisy rotation and translation
    - None - use existing rotations and translations
5. Whether drop outliers or not
6. Whether use Bundla Adjustment or not

As output user gets reconstructed *.asc files with suffix depending on used methods and choosed features. TODO write more about it

5.4.2 Important Implementation Aspects

Write about how you modified different libraries and how you tricked OpenCV to do what you need to get

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 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, for example 8-point and 5-point algorithms. The only thing to do is to change input data from local image coordinates into camera coordinates. In OpenCV it can be done with:

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

where rotDiffGlobal is rotation between 2 cameras. These lines autmaticaly allow as to calculate $h\prime and h$ from 4.4. In order to choose proper matrix decomposition we can do as follows:

```
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;
    }
}
```

5.4.2.3 3-point translation estimation, between cameras

When we do have rotation data, which are more or less accuratre, we can focus on estimating only translation between images. 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 filter out outliers with RANSAC schema.

```
...
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

There was nothing special to implement regarding enhanced pose estimation. OpenCv has option to use initially estimated rotation and translation values(reference)

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

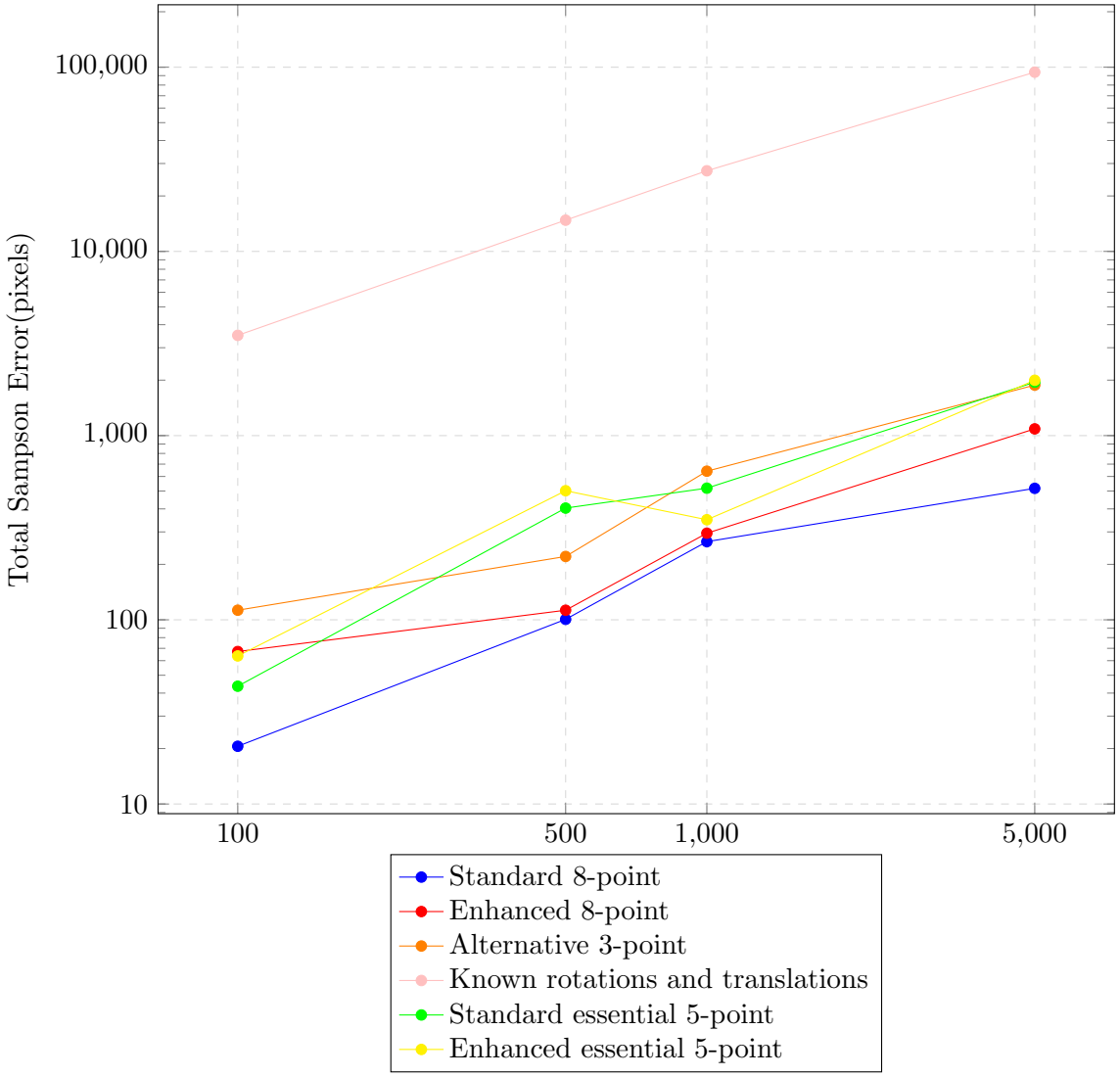

Figure 6.1: Chart showing the sum of Sampson Errors in a picture(1024x768pixels) per various initial Sift features sets sizes

their corresponding epipolar lines. However, each of 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 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. 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. Standard versions for instance can fail sometimes in terms of Fundamental matrix estimation.

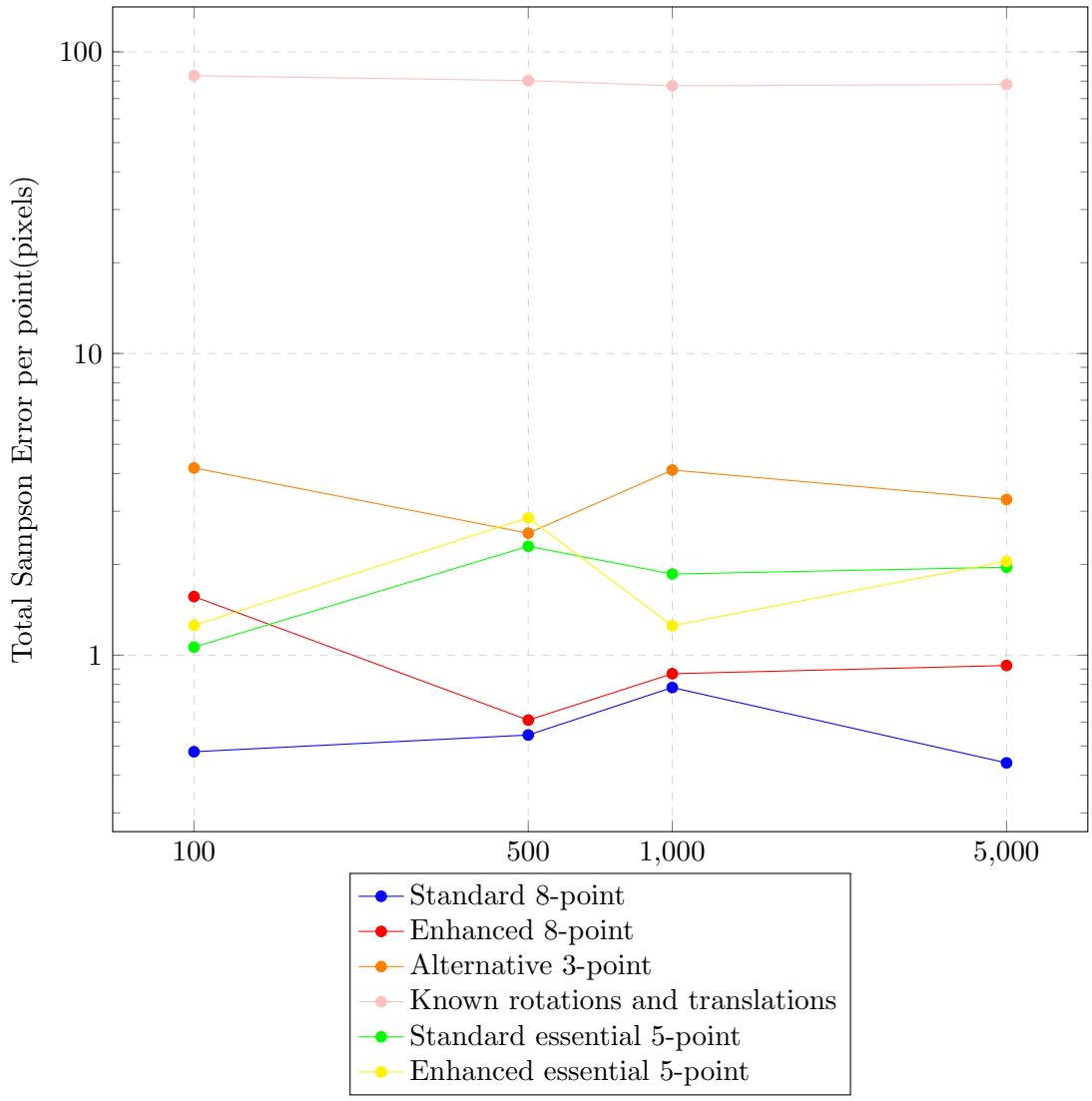

Figure 6.2: Chart showing per point Sampson Error in a picture(1024x768pixels) per various initial Sift features sets sizes

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: 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: 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

On chart 6.7 avaraged execution time for 100 estimation attempts was presented. Execution time of 8-point versions are very similar, which is understandable, because

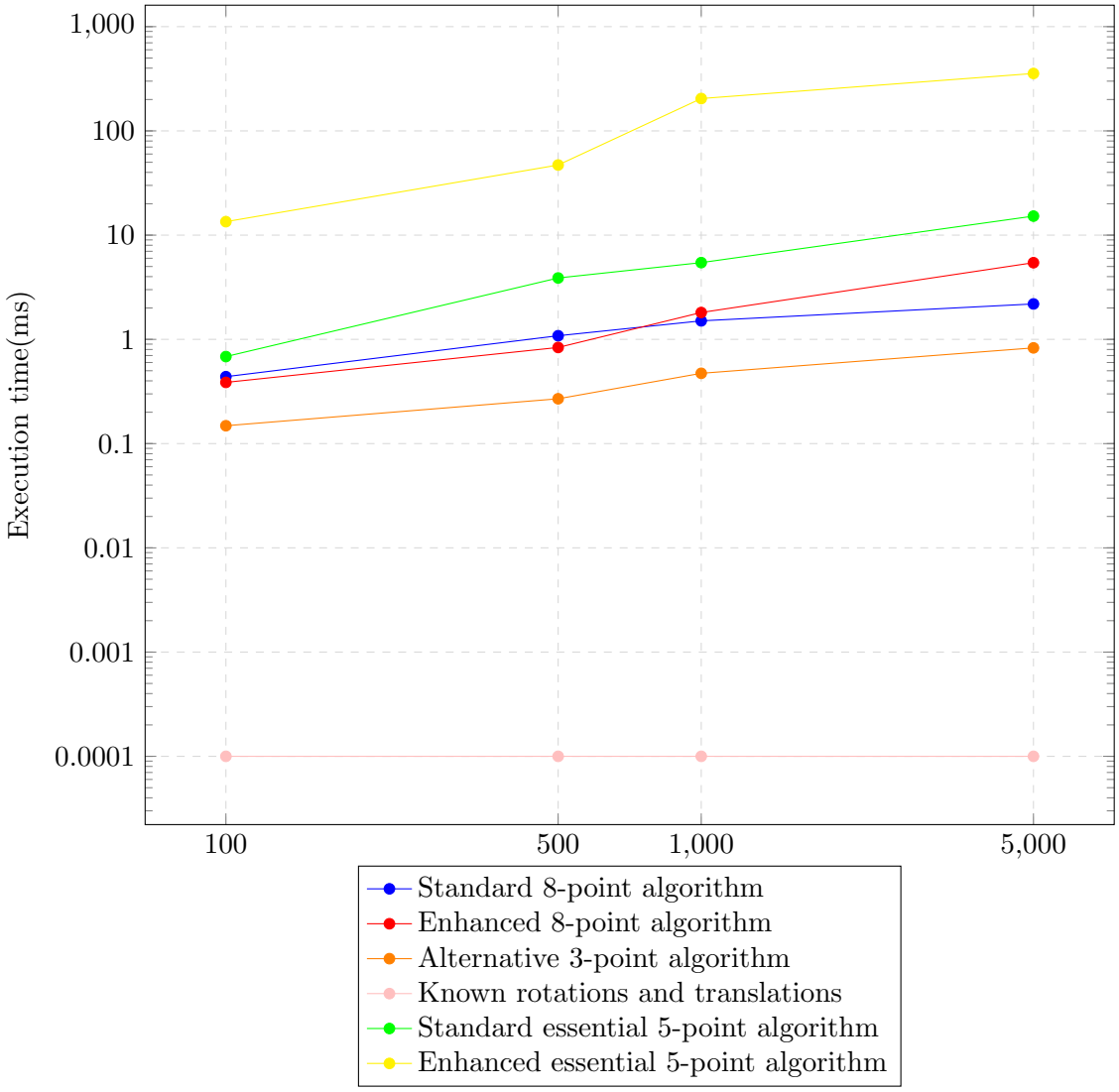

Figure 6.7: Execution time of proposed algorithms(1024x768pixels) in comparison to initial SIFT feature set

they implementation is pretty much the same and they differ only characteristic of analysed dataset. It can be seen that execution time of 5-point epipolar estimations versions differ much. In this situation either finding optimal solution with initial ro-

tation is much harder or implementation of these methods are very different in terms of memory allocation. Execution time of 3-points algorithm is few times faster than 8-point algorithms. Estimation using only known rotations and translations has few times smaller order execution time than standard algorithms, but as we know from previous section gives uncorreleted 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 few 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

First method gives us basic reference to normal 3D reconstruction pipeline. 2nd and 3rd one were used in order to check, how Pose Estimation enhancment influence final results. 4th and 5th one were compared with 2nd-3rd strategies. Finally 6th one was proposed in order to check, how well reconstraction can be performed with only sensor data.

6.4.1 Accuracy

Accuracy of 3D reconstraction were measured using Bundle Adjustment process from SBA library[ref]. At the beginning initial error of whole point cloud is calculated. What's more to better understand, how well models were reconstructed it's intresting to take a look at both model error reduction and BA execution time. For "Warsaw Univeristy" dataset performed performence tests were plotted on 6.8. Big differences in Initial Error mainly lay in number of reconstructed points and scale of models, which turned out to be more very different due to differences in outlier removals. What's interesting is impact of enhancing reconstruction with initial rotations and translation on Bundle Adjustments.

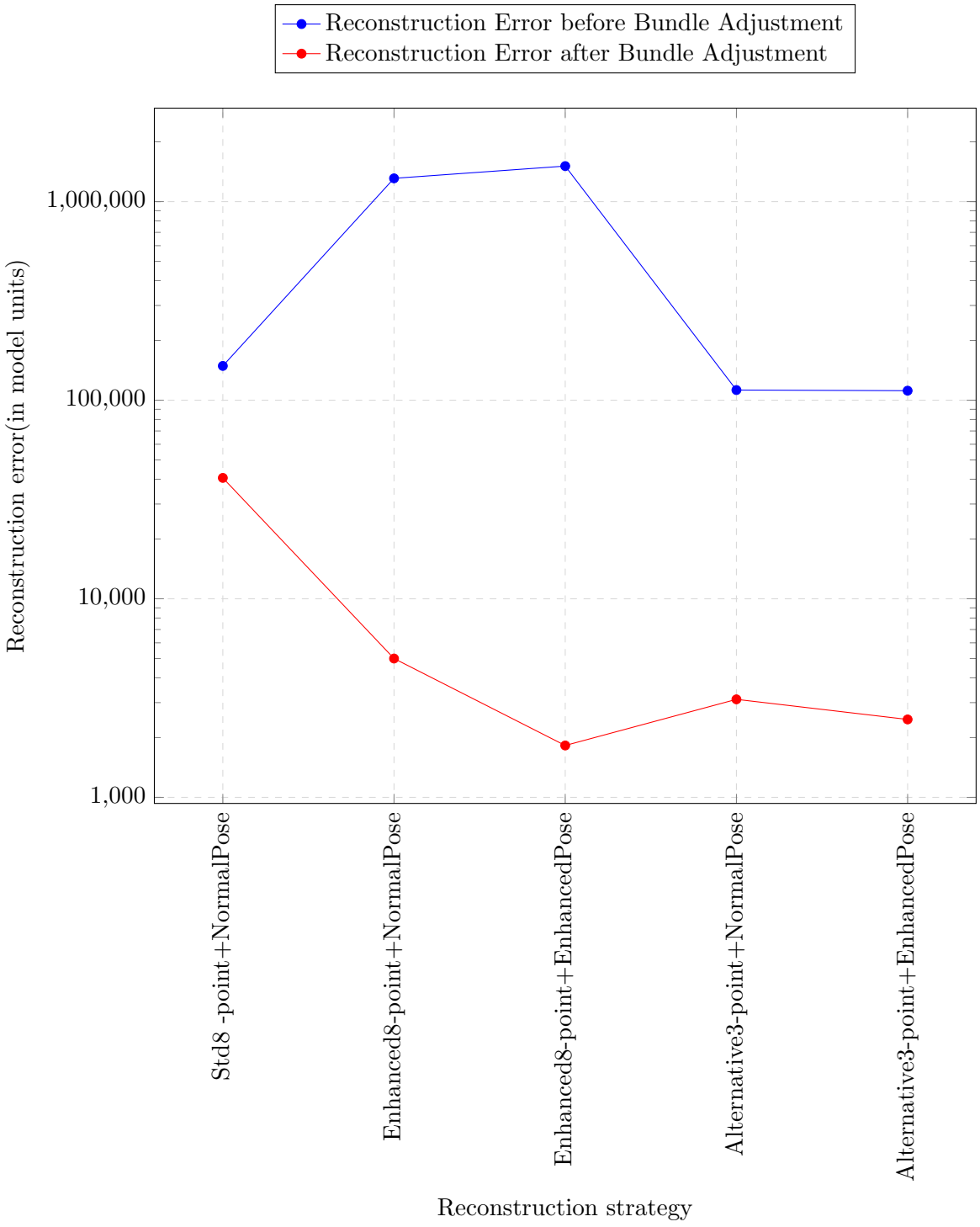

Figure 6.8: Comparison of different reconstruction strategies influence on Bundle Adjustment

Bundle Adjustmet process allows to rearrange both estimated 3D points positions and positions of the cameras. When camera positions are close to their true,optimal positions it allows algorithms to focus on 3D points movement refinments. What's more Rotation and Translation enhanced position estimation has further influence in reduction of final BA error in comparison to Normal Pose Estimation.

6.4.2 Execution time

To see how proposed algorithms improve reconstruction process execution time without Bundle Adjustment was measured in 6.10. What can be noted that in comparison to Epipolar lines estimation overall reconstruction process time was slightly improved especially with 3-point algorithms. Comparing it to Execution times in 6.7 one can see that most of this time is used for SIFT correspondences matching, which is bottle-neck in proposed reconstruction methodology[reference to concept pose esitmation choosing]. Also Reconstruction time was also measered with Bundle Adjustment process at the end(6.11). It turns out that Bundle Adjustment works much better on with enhanced Initial Pait reconstruction and Pose Estimation. Such cloud points are much better organised than 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

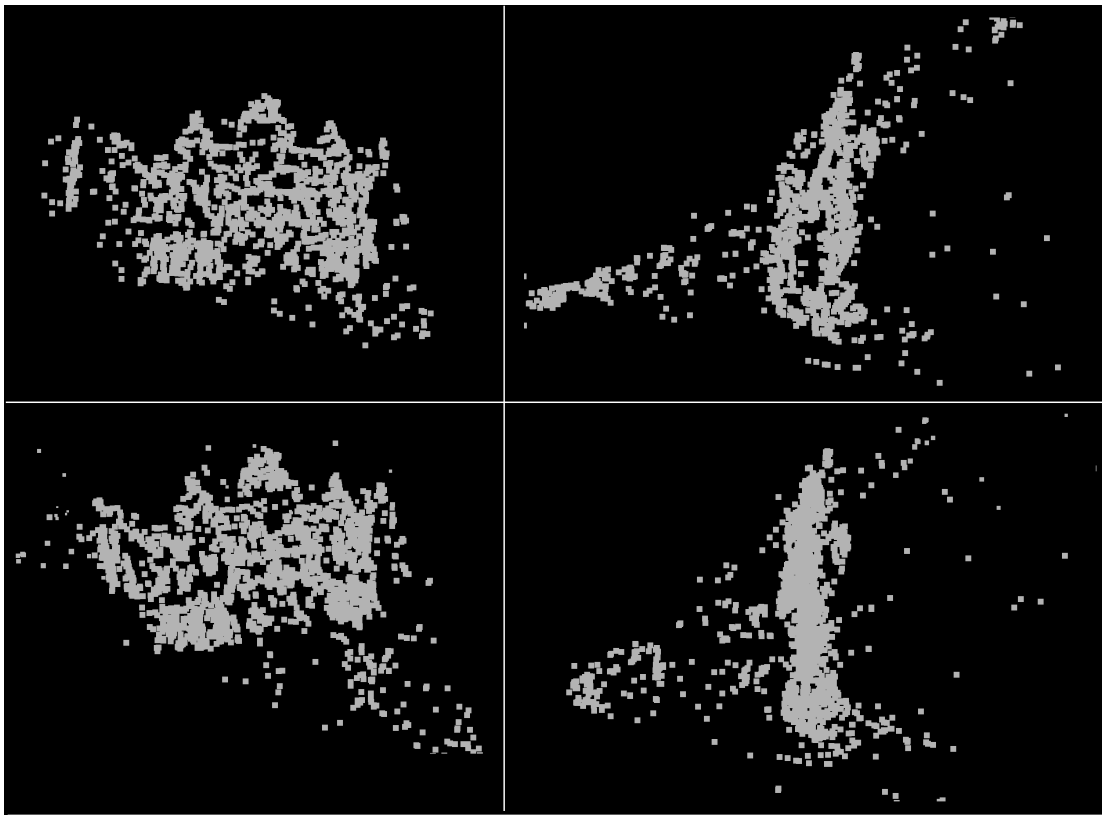

Figure 6.9: 3D point clouds before Bundle Adjustment(upper) and after(bottom) for Enhanced 8-point with Enhanced Pose Estimation. Warsaw University of Technology dataset for 1000 SIFT corresponding features(left - front, right - from side)

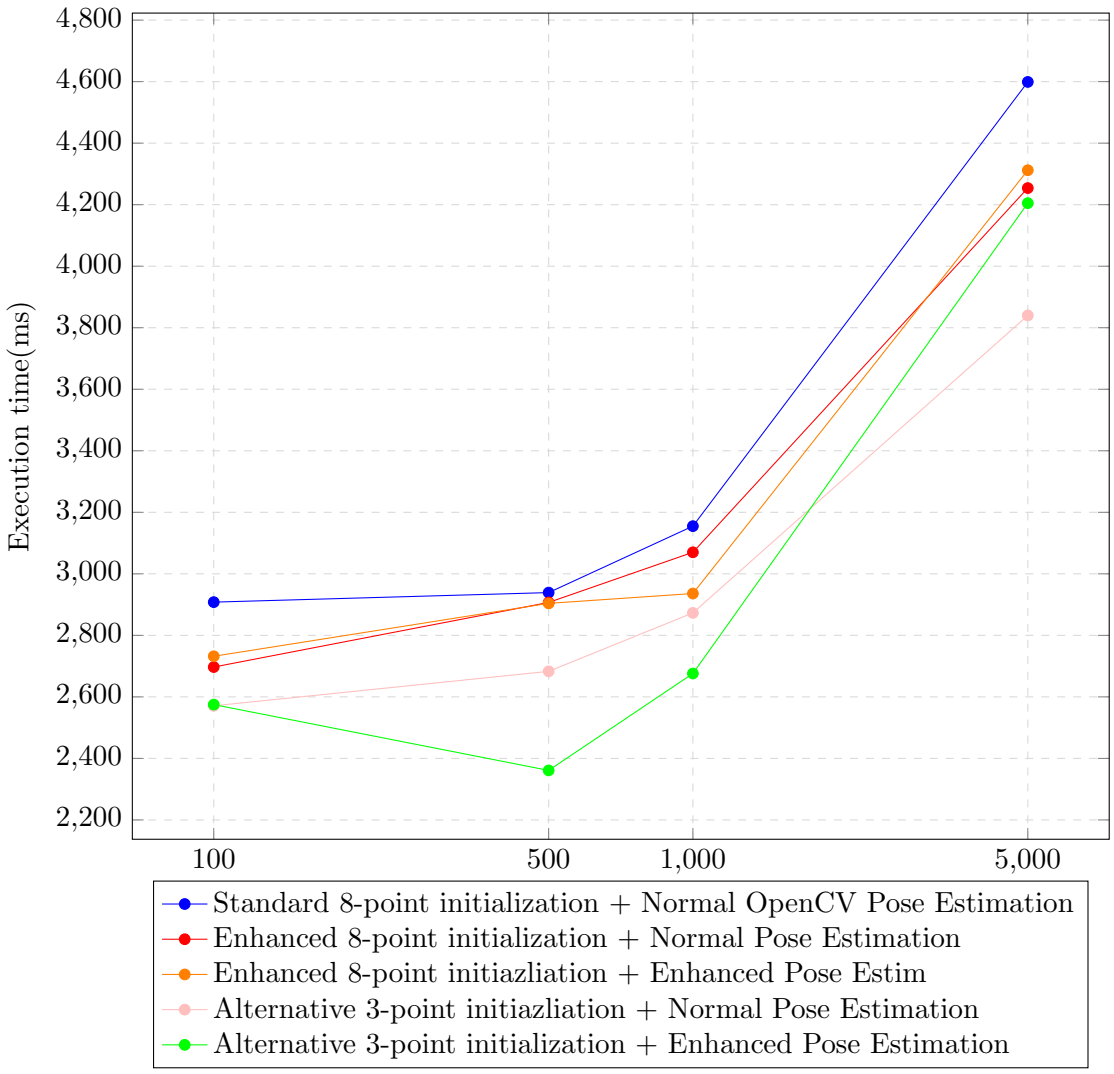

Figure 6.10: Total reconstruction execution time (4 images with resolution 1024x768pixels) in comparison to Sift feature number

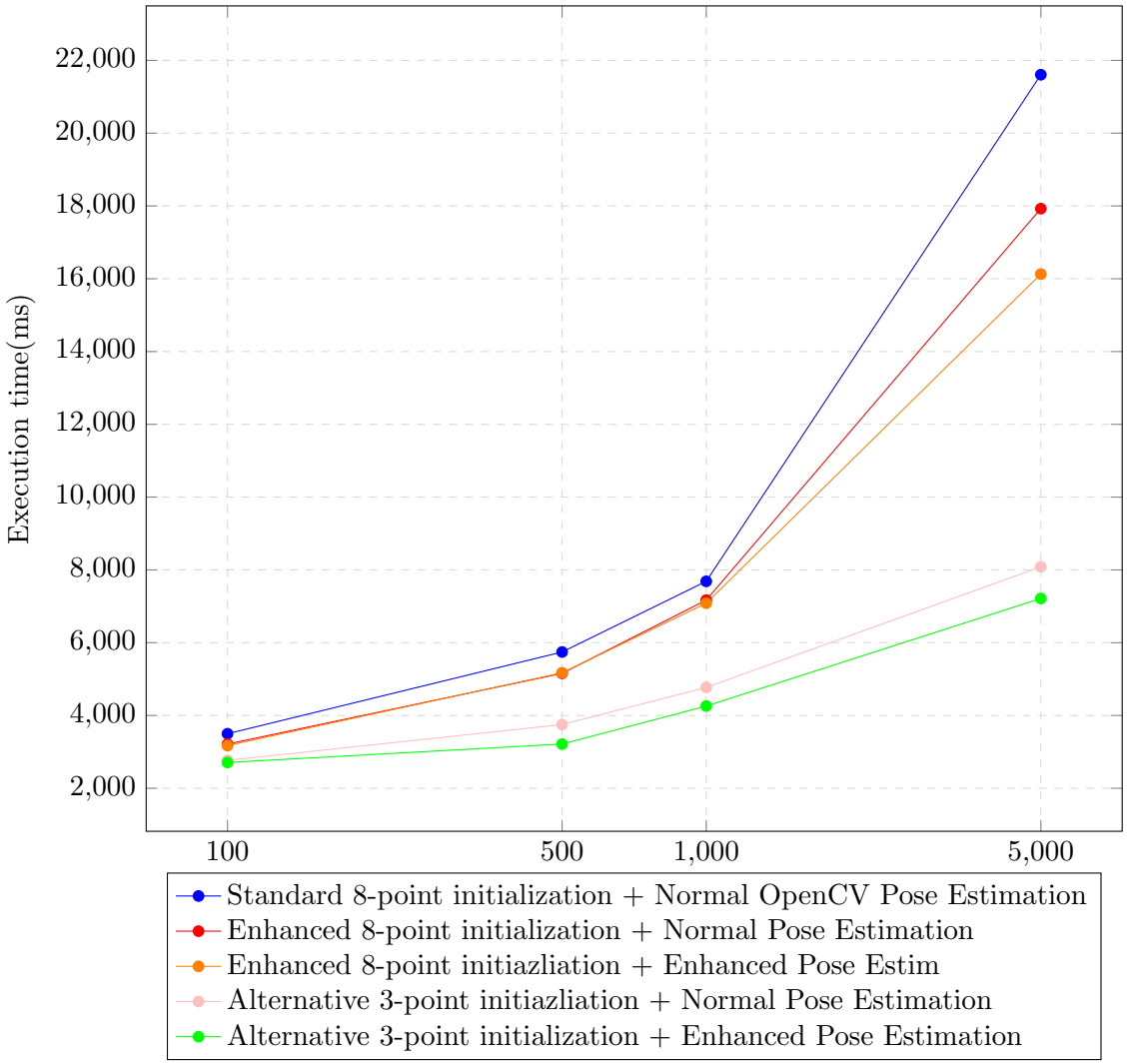

Figure 6.11: Total execution time of reconstruction with Bundle Adjustment(4 images with resolution 1024x768pixels) in comparison to Sift feature number

6.5 Effectivness

However numerical measuerments do not always results in proper 3D model reconstructions, so some of the intresting 3D reconstraction situations are presented on next 4 pages. In 6.12 different Initialization Pair methods were used to reconstruction models for 4000 SIFT features. It can be seen that standard and enhaced methods gave quite good results, which only differ in scale. 3-point algorithm produces quite good models, but slightly distorted due to noise in used rotations. Model reconstructed from known rotations and translations are very distorted, but stil readable. It could be used, when very fast reconstruction is needed. Essential Matrix estimation methods unfortunately failed in terms of relative pose estimation.
However situation changes a little bit for 400 points. Every algorithm managed to find solution reallt close to optimum. In some cases "Traingulation Test" used in Standard 8-point approach fails and produces unrecognizable model(6.14).
When it comes to Pose Estimations enhancments one can see in 6.15 that generally initial rotation and translation help to find better solution, which mainly manifests in less outliers number.
In figure 6.16 it can be seen, how many outliers using only noisy rotations and translations produces.
More reconstructed models can be found in [TODO Materials]

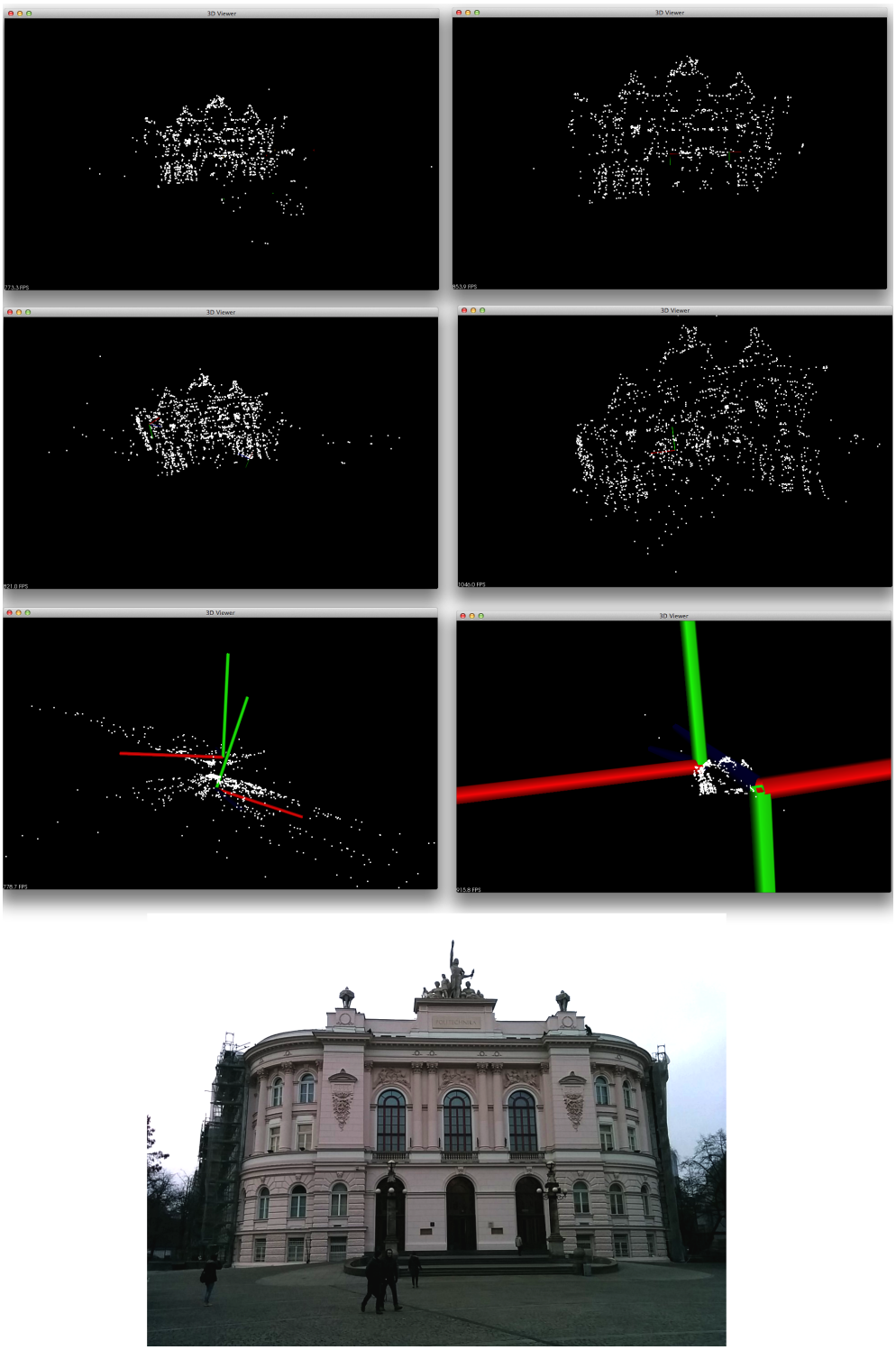

Figure 6.12: Reconstructed models for proposed Initial reconstruction methods and 4000 Sift Features. From upper left to bottom right following: 1) Standard 8-point, 2) Enhanced 8-point, 3) Alternative 3-point, 4) Known rotations and translations, 5) Standard 5-point, 6) Enhanced 5-point

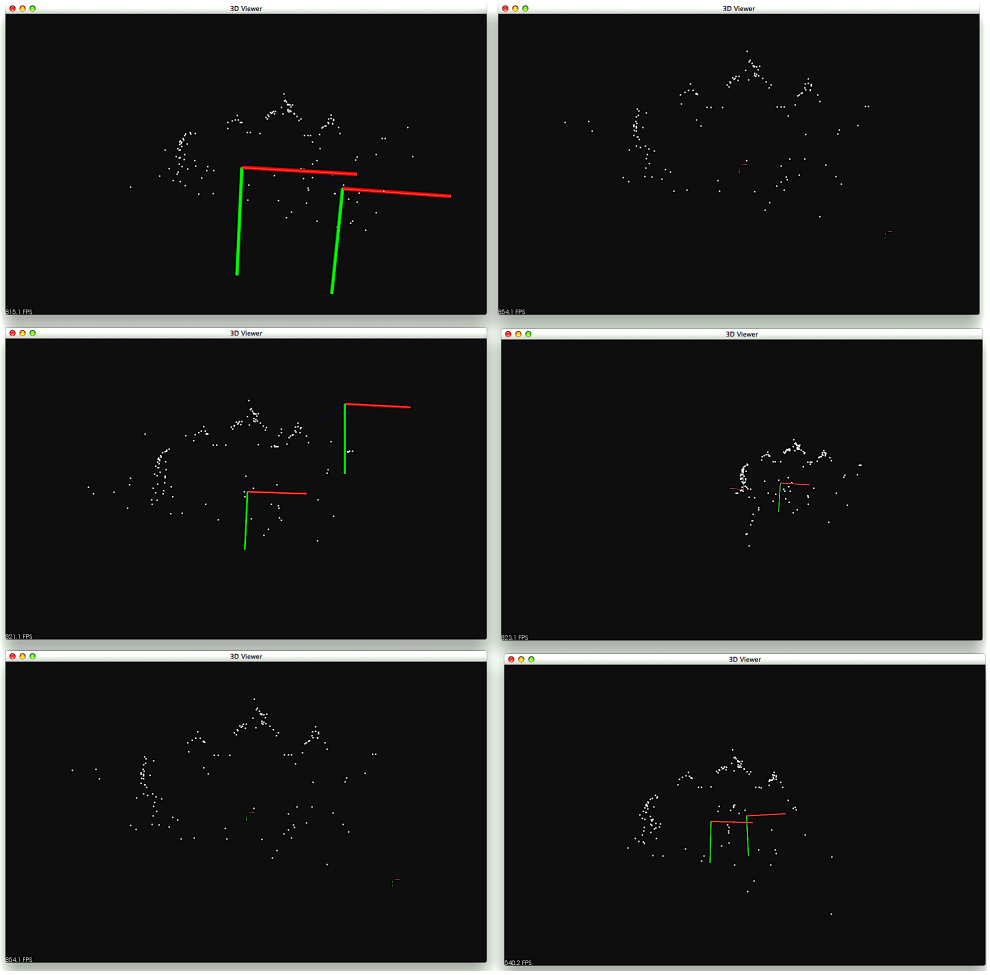

Figure 6.13: Reconstructed models for proposed Initial reconstruction methods and 400 Sift Features. From upper left to bottom right following: 1) Standard 8-point, 2) Enhanced 8-point, 3) Alternative 3-point, 4) Known rotations and translations, 5) Standard 5-point, 6) Enhanced 5-point

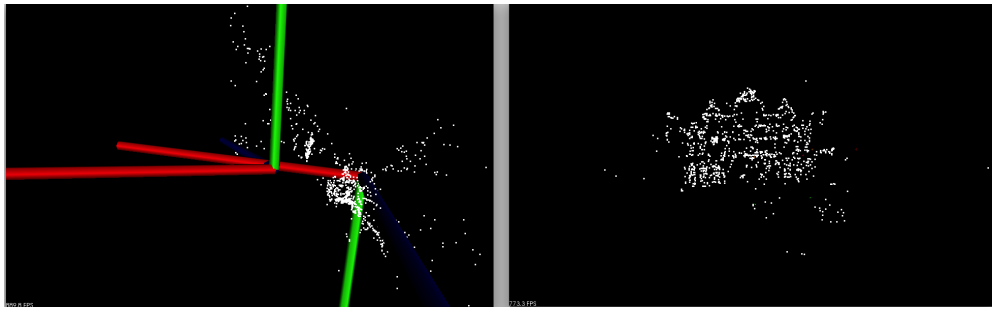

Figure 6.14: Fail test case of Standard 8-point triangulation(left) in comparison to fortunate reconstruction

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 are reconstructed when enhancment is used.

Figure 6.16: Reconstruction results from known translations and rotations from different angles. Upper shows front face of building, other from different angles. Many outliers in reconstructed model.

Chapter 7

Conclusion

This chapter describes the whole work performed during master thesis, discusses encountered problems and proposes ideas for future work.

7.1 Summary

The idea of enhancing standard 3D reconstruction algorithms with usage of sensor data was concepted, implemented and verified in few different version. Reconstractiun strategy can be adapted, depending on accuracy and speed compromis. Proposed standard 8-point algorithm improvments allows to quickly determine proper Essential matrix decomposition to rotation and translation. Proposed standard 5-point algorithm improvements resulted in better accuracy than standard 5-point algorithm in terms of Sampson Error. However execution time of improved 5-point algorithm version was slightly bigger than standard version. Using initial rotation and translation estimation in Pose Estimation results in better reconstruction accuracy especially in terms of outliers removal and Bundle Adjustment convergence speed. Generally sensor enhanced algoritms resulted in bigger Bundle Adjustment error reduction in much shorter time. One of the bottle-neck in proposed reconstruction process time was correspondences matching. Using only sensor data is not enough to create accurate 3D models. However it's possible to briefly catch the outline of reconstructed models, which can be used in order to enhance object recognition in images[reference to similar papaer]. The only problem, which have to be solved is how to distuinguish inliers from outliers in such cases. TODO quick talk about proposed heuristic on rotation and movement. And

general capability of reconstruction, and estimating dR instrad of R and ambiguity reduction.

7.2 Dissemination

Up to this point no one used implemented applications. However Android app can be interesting for people, who deals with 3D reconstruction techniques and it is planned to be published once improved and properly tested. "Enhanced 3D reconstructer" is alread open-sourced GitHub project with LGPL license.

7.3 Problems Encountered

Most of the problems were related to implementation of proposed algorithms. Decomposing rotation to Euclidan angles and combining angles can be tricky and should be done in the same order. Android API allows user to get rotation quaternion and a way to decompose it, but it doesn't explain how it's calculated. Some tests were made in order to figure it out. Also since it was first serious C++ and OpenCV project made by author, it was hard to deal with some memory related issues were hard to resolve. It was really hard to combine all necessery libraries together, especially Bundle Adjustment, which requires storing point visibility informations and is hard to implement. Preparing heuristic for movement measurments was also tricky, but after few basic tests it turned out it works at least up to relative differences in direction of X,Y and Z axis

7.4 Future work

It would be good to test reconstruction process with video sequence and optical flow estimation. This would both allow for quicker relative pose estimation correspondences matching and later could be used for dense reconstruction. All of used librieries are also available in Android versions, so it would be good to see is it possible to achieve real-time tracking and mapping(similar to PTAM). What's more interesting idea would be to store each reconstructed cloud with GPS camera position in cloud. Later first step in reconstruction could be searching the server for initially reconstracted model and whole analysis could be based on mapping, matching and improving existing model.

However it would be essential to create algorithm, which would decide, which cloud points should be sent to server as reference

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,