

Enhancing 3D reconstruction using Mobile Sensors Data

L^AT_EX

Krzysztof Wrbel
CollegeOrDepartment
University

A thesis submitted for the degree of
PhilosophiæDoctor (PhD), DPhil,..
year month

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	vii
List of Tables	ix
1 Introduction	1
1.1 Purpose of this thesis	1
1.2 Scope	1
1.3 Initial assumptions	2
1.4 Thesis Outline	2
2 Fundamentals	3
2.1 3-D reconstruction in general	3
2.1.1 Feature extraction and correspondence matching	4
2.1.2 Fundamental & Essential Matrices	4
2.1.3 Triangulation	4
2.1.4 Common problems	4
2.2 Structure from Motion	4
2.2.1 Relative Pose Estimation	4
2.2.2 Homography estimation	4
2.2.3 Projective Factorization	4
2.2.4 Bundle Adjustment	4
2.2.5 Common problems	4
2.3 Mobile Sensors overview	4
2.3.1 Accelerometer	4
2.3.2 Gyroscope	4
2.3.3 Magnetometer	4

CONTENTS

2.3.4	Sensor Fusion	4
3	Related Work	5
4	Concept	7
4.1	Requirements	7
4.2	Reconstruction process strategy	8
4.3	Enhancing epipolar geometry equation	9
4.3.1	Rotation enhancements	9
4.3.2	Alternative 3-point algorithm for translation finding	10
4.3.3	Translation enhancements	11
4.4	Pose estimation	11
4.4.1	Known rotations & translations	11
4.4.2	Rotation enhancements	12
4.4.3	Rotation & translation enhancements	12
5	Implementation	13
5.1	Choosing Environment	13
5.2	Project Structure	13
5.3	"Sensor Enhanced Images" - Android Gradle based project	14
5.3.1	Installation	14
5.3.2	User Interface	14
5.3.3	Important Implementation Aspects	14
5.3.3.1	Rotation calculation	14
5.3.3.2	Custom heuristic for move estimation	15
5.3.3.3	Custom Sensor Data File format	16
5.4	"Enhanced Reconstructor" - OSX CMake based project	18
5.4.1	User Interface	18
5.4.1.1	Test Efficiency	18
5.4.1.2	Test reconstruction	19
5.4.2	Important Implementation Aspects	19
5.4.2.1	Enhancing epipolar equations	20
5.4.2.2	3-point translation estimation, between cameras	20
5.4.2.3	Enhancing pose estimation	20

6	Evaluation	21
6.1	Test Environment	21
6.1.1	Synthetic Data	21
6.1.2	Real world Data	21
6.2	Usability	21
6.3	Performance	21
6.3.1	Accuracy comparison	21
6.3.2	Time comparison	21
7	Conclusion	23
7.1	Summary	23
7.2	Dissemination	23
7.3	Problems Encountered	23
7.4	Future work	23
8	Materials & methods	25
	References	27

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

GLOSSARY

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 includes 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 enhancement with data acquired with sensors, which can be found in smartphones. At the end author presents evaluation and discusses 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.

1. INTRODUCTION

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 Estimation" Open-source project set up 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 in 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

2

Fundamentals

In order to help user understand topics mentioned in this thesis, small theoretical background is needed.

2.1 3-D reconstruction in general

Today we have many devices, which are capable of 3D reconstruction. Most popular Kinect(?) is capable of real-time 3D cloud point generation, but that's very special case, because it uses 2 camera: RGB camera and IR depth-finding camera.

2. FUNDAMENTALS

2.1.1 Feature extraction and correspondence matching

2.1.2 Fundamental & Essential Matrices

2.1.3 Triangulation

2.1.4 Common problems

2.2 Structure from Motion

2.2.1 Relative Pose Estimation

2.2.2 Homography estimation

2.2.3 Projective Factorization

2.2.4 Bundle Adjustment

2.2.5 Common problems

2.3 Mobile Sensors overview

2.3.1 Accelerometer

2.3.2 Gyroscope

2.3.3 Magnetometer

2.3.4 Sensor Fusion

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 estimate 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 robust 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 Technische Universität Berlin made a 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 accurately 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 shown in paper[reference] can outperform even some versions 5-point algorithm. Lately scientists are creating more complex models to estimate 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 estimate 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 show 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 the 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 were 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.

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 achieve 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 rely 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 algorithmic 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 - euclidean rotation and optionally translation. Usage of smartphone is actually not necessary. Any camera with SensorFused accelerometer and gyroscope(magnetometer 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. CONCEPT

4.2 Reconstruction process strategy

Both Epipolar equations and Pose Estimation improvements can be used in different configurations. It's expected that some of them are more accurate and some are faster. It was necessary to evaluate these approaches both in terms of speed and accuracy. The author of this thesis proposes an environment, where the user can decide what type of strategy she/he wants to use. In terms of initialization of 3D cloud point reconstruction can be made using:

1. Known rotations and translations - relative poses are calculated from sensor data - euclidean 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 extrinsic parameters - standard fundamental/essential for R and Translation

Later one new images are analyzed in comparison to previous one to enrich initial cloud point with new points. It's only necessary 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 rotation and translations - no additional calculations, just triangulation - euclidean 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 extrinsic parameters - standard pose estimation

Of course in all of these steps it's really important to have as minimum outliers as possible. All of these methods are pretty good in terms of removing outliers, especially 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 ($P = [I|0]$, $P' = [R|t]$) we can note that:

$$x'^T * K^{-T} * [T]_x * R * K^{-1} * x = 0 \quad (4.1)$$

It's also good to note that:

$$[T]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \text{ where } T = [t_x, t_y, t_z] \quad (4.2)$$

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

$$R = R_{error} * R_{init} \quad (4.3)$$

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 multiplying 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:

$$x'^T * K^{-T} * [T]_x * R_{error} * R_{init} * K^{-1} * x = 0 \quad (4.4)$$

Having:

$$\begin{aligned} h'^T &= x'^T * K^{-T} \\ h &= R_{init} * K^{-1} * x \\ G &= [T]_x * R_{error} \end{aligned} \quad (4.5)$$

With such notation one can notice that

$$h'^T * G * h = 0 \quad (4.6)$$

quite resembles both standard fundamental and essential equation [reference]. Of course h' 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

4. CONCEPT

matrix can be resolved for instance by both 5 and 8-point algorithms. So basically standard fundamental and essential equation solvers can be used in order to retrieve both $[T]_x$ 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 Rodrigues 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:

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

[Refernece to Hartley]. It can be used when decomposing G to ensure proper decomposition.

4.3.2 Alternative 3-point algorithm for translation finding

Starting 4.6 it can be written that:

$$x_i^T * K^{-T} * \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} * R * K^{-1} * x = 0 \quad (4.8)$$

Having:

$$\begin{aligned} h_i^T &= x_i^T * K^{-T} \\ h &= K^{-1} * x \end{aligned} \quad (4.9)$$

$$[h_1 \quad h_2 \quad h_3] * \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} * \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0 \quad (4.10)$$

and multiplying it we will end up with

$$h_1 * h_2 * t_z - h_1 * h_3 * t_y - h_2 * h_1 * t_z + h_2 * h_3 * t_x + h_3 * h_1 * t_y - h_3 * h_2 * t_x = 0 \quad (4.11)$$

what can be grouped:

$$t_x * (h_2 * h_3 - h_3 * h_2) + t_y * (h_3 * h_1 - h_1 * h_3) + t_z * (h_1 * h_2 - h_2 * h_1) = 0 \quad (4.12)$$

rewritten as:

$$[t_x \quad t_y \quad t_z] * \begin{bmatrix} (h_2 * h_3 - h_3 * h_2) \\ (h_3 * h_1 - h_1 * h_3) \\ (h_1 * h_2 - h_2 * h_1) \end{bmatrix} = 0 \quad (4.13)$$

and solved for instance with SVD with only 3 points. This is very attractive way of reconstructing images from only 3 points especially in terms of speed. Overall accuracy strictly relies on precise measurements 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 retrieved and only affine reconstruction can be done. As described 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 Euclidian 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 continuous 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:

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

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

$$P = K * [R | t] \quad (4.15)$$

4.4.1 Known rotations & translations

In situation where rotations and translations of cameras are known no additional 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

4. CONCEPT

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:

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

Substituting $x_m = x - K * [Rinit | 0]$ we get:

$$x_m = K * [dR | t] * X \quad (4.17)$$

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:

$$\begin{aligned} x &= K * [Rinit + dR | Tinit + dt] * X \\ x &= K * [Rinit | Tinit] * X + K * [dR | dt] * X \\ x - K * [Rinit | Tinit] &= K * [dR | dt] * X \end{aligned} \quad (4.18)$$

end in the end by Substituting $x_n = x - K * [Rinit | Tinit]$ we get:

$$x_n = K * [dR | dt] * X \quad (4.19)$$

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

5

Implementation

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

5.1 Choosing Environment

One thing was to prepare application, which would be capable of taking both pictures with additional SensorFused data. Currently Android has one of the best APIs which allows user to get both raw and fused sensor data. In order to avoid coding from scratch all stereo 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 debugging and speed performance. This is why author decided to separate 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 separated to different subprojects: Android and Desktop. In addition some sample datasets were added to Dataset folder.

5. IMPLEMENTATION

5.3 "Sensor Enhanced Images" - 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 fused" 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 heuristic 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_ACCELERATION* 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 data vector from local camera coordinates to global reference system(multiply with inverted rotation matrix)
3. Decide depending on current state, if deviceMovementState has changed:
 - (a) If device was previously IDLE and incoming Acceleration value is bigger than $0.5 \frac{m}{s^2}$ change device state to MOVING and reset current velocity to $0 \frac{m}{s}$
 - (b) If device was previously MOVING and incoming Acceleration value is smaller than $0.1 \frac{m}{s^2}$ change device state to IDLE
4. Only if device is currently moving:
 - (a) Update device velocity

5. IMPLEMENTATION

- (b) Check current speed with walk constraint(People walk with average speed $1.5 \frac{m}{s}$) and eventually scale it down to maximum value
- (c) Update device position

Described algorithm implementation snippet can be found in listing 5.1. Human walking model was introduced in order to constrain the velocity. Without it integrating small noise over time, would eventually result in high unrealistic movements. Maximum walking speed can be adjusted but by 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 influences each axis, so it should be statistically correct.

5.3.3.3 Custom Sensor Data File format

As mentioned 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 attachments.

5.3 "Sensor Enhanced Images" - Android Gradle based project

```
1 public void onSensorChanged(SensorEvent event) {
2     ...
3     } else if (event.sensor == mLinearAcceleration) {
4         ...
5         //Filtering out some noise
6         newGlobalAcceleration = lowPass(newGlobalAcceleration,
7             currentGlobalAcceleration);
8         //Switch linear acceleration from phone local coordinates to
9             global coordinates
10        Matrix.multiplyMV(newGlobalAcceleration, 0,
11            invertedRotationMatrix.clone(), 0, newGlobalAcceleration,
12            0);
13
14        double distance = getLength(currentGlobalAcceleration);
15        long currentTimeMillis = System.currentTimeMillis();
16        //Decide state of the device. Distance is in m/s^2
17        if (distance > 0.5 && deviceState == State.IDLE) {
18            if (currentTimeMillis - movingEndTime > 300) {
19                deviceState = State.MOVING;
20                currentGlobalVelocity = {0,0,0};
21            }
22        } else if (distance < 0.1 && deviceState == State.MOVING) {
23            if (currentTimeMillis - movingStartTime > 350) {
24                deviceState = State.IDLE;
25            }
26        }
27
28        if (deviceState == State.MOVING) {
29            //Update current device velocity
30            currentGlobalVelocity += currentGlobalAcceleration * dT;
31
32            double velocity = getLength(currentGlobalVelocity);
33            //Check and adjust current velocity. People walk with
34                average speed 1.5\frac{m}{s}
35            if (velocity > WALKING_MAX_VELOCITY) {
36                currentGlobalVelocity /= velocity /
37                    WALKING_MAX_VELOCITY;
38            }
39
40            //Update device relative position, s = V0 * t + a * t
41                ^2/2;
42            currentRelativePosition += currentGlobalVelocity * dT +
43                currentGlobalAcceleration * dT * dT / 2;
44        }
45        ...
46    }
```

Listing 5.1: Snippet from Android source code position estimation heuristic

5. IMPLEMENTATION

5.4 ”Enhanced Reconstructor” - 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 reconstrution - 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. IMPLEMENTATION

5.4.2.1 Enhancing epipolar equations

5.4.2.2 3-point translation estimation, between cameras

5.4.2.3 Enhancing pose estimation

6

Evaluation

6.1 Test Environment

6.1.1 Synthetic Data

6.1.2 Real world Data

6.2 Usability

6.3 Performance

6.3.1 Accuracy comparison

6.3.2 Time comparison

6. EVALUATION

7

Conclusion

7.1 Summary

7.2 Dissemination

Who uses your component or who will use it? Industry projects, EU projects, open source...? Is it integrated into a larger environment? Did you publish any papers?

7.3 Problems Encountered

7.4 Future work

7. CONCLUSION

8

Materials & methods

8. MATERIALS & METHODS

```
1  [  
2    {  
3      "photoPath": "20141210_145643/0.jpg",  
4      "rotationMatrix": [],  
5      "azimuth": 121.88075,  
6      "posX": -1.7521392107009888,  
7      "posY": -1.4345977306365967,  
8      "posZ": 0.9248641133308411,  
9      "photoId": 1,  
10     "pitch": 13.867888,  
11     "roll": 178.16968  
12   },  
13   {  
14     "photoPath": "20141210_145643/1.jpg",  
15     "rotationMatrix": [],  
16     ],  
17     "azimuth": 110.66925,  
18     "posX": -4.244707942008972,  
19     "posY": -1.1443554759025574,  
20     "posZ": 0.9647054933011532,  
21     "photoId": 2,  
22     "pitch": 11.625216,  
23     "roll": 179.73383  
24   }  
25   ...  
26 ]
```


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,