MID SWEDEN UNIVERSITY
Information and Communication Systems

**Rough concept**

# Camera movement detection using image analysis

Markus Ineichen

27. Januar 2017

# Contents

**1 Motivation** **3**

**2 Goals** **4**
2.1 Desired Goals . . . . . 4
2.2 Project Goals . . . . . 4
2.3 Research Questions . . . . . 4

**3 Results** **5**

**4 Rough planning** **6**
4.1 Masterplan . . . . . 6
4.2 Sprints . . . . . 6
4.2.1 Sprint 1 . . . . . 6
4.2.2 Sprint 2 . . . . . 6
4.2.3 Sprint 3 . . . . . 7
4.3 Milestones . . . . . 7
4.4 Measurement . . . . . 8
4.4.1 Scenes . . . . . 8
4.4.2 Method . . . . . 8

# 1 Motivation

Since the determination of extrinsic camera parameters is computationally intensive, redundant recalculations should be avoided. If the cameras position did not change, the parameters should not change either. Therefore, a component which could detect camera-position changes independent of movement in the scene could help to reduce the computing time by invoking the recalibration just as required.

# 2 Goals

## 2.1 Desired Goals

The following points describe a optimal solution. In chapter „2.2 Project Goals“, these points are converted into realistic goals for this project.

- Creation of a reliable, automated process that can detect when a camera has changed its position, based solely on the available images / video captured by the camera itself.
- One camera movement should trigger exactly one event after the position is stable again.
- The delay between the pose change and the recognition thereof should be close to zero.
- Associated calculation costs should be as small as possible.

## 2.2 Project Goals

The following goals are subjects to be achieved by the end of the project:

| | |
|---|---|
| G1 | false negative rate needs to be below 5% |
| G2 | false positive rate needs to be bellow 1% |
| G3 | The software should run on the raspberry pi hardware |

$$G1 = falseNegatives/numberOfExpectedChanges$$

$$G2 = falsePositives/numberOfFrames$$

Although a low *falsenegative* rate is much more important than a low *falsePositives* rate, the boundaries of G1 are higher than G2's. This is reasoned by the fact that $numberOfFrames \gg numberOfExpectedChanges$.

## 2.3 Research Questions

At the end of the project, reflection to the following questions will be done:

- Can camera motion be detected in respect of G1 and G2?
- Is the algorithm fast enough to run on the raspberry pi entirely (G3)?

# 3 Results

This project generates the following results:

**R1 Software** .

**R1-T1 Change detection algorithm** source code and build instructions

**R1-T2 GStreamer plugin** Integration of R1-T1 into GStreamer

**R2 Documentation** Comprehensible description about the measurements

# 4 Rough planning

## 4.1 Masterplan

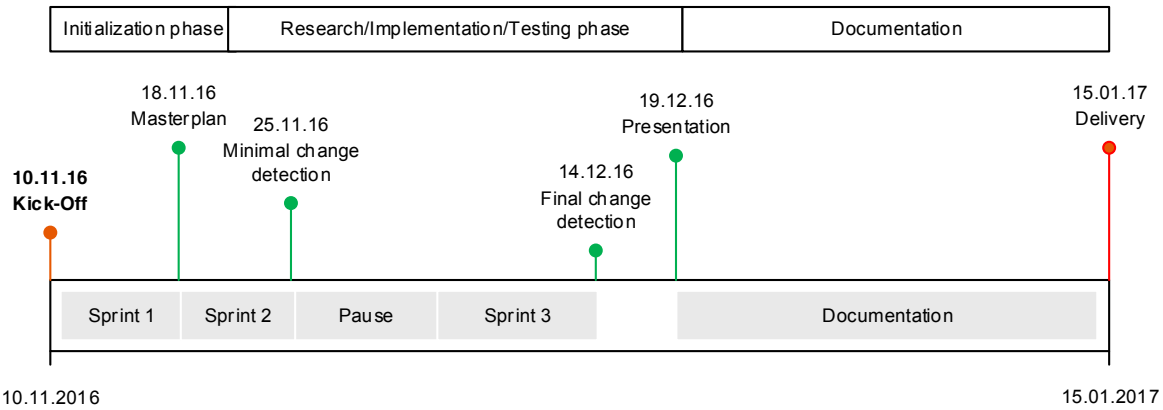


Abbildung 4.1: Masterplan

The project is split into three phases. *Initialization* phase is all about setting up the project. During the *Research/Implementation/Testing* phase, the programm is developed and in the *Documentation* phase, gained knowledge will be written down. Because of the absence of Markus Ineichen, the project stagnates from the 25.11.16 until 04.12.16.

## 4.2 Sprints

The time is split into three sprints with the following tasks:

### 4.2.1 Sprint 1

- project setup and properly define all results, their constraints and test sequences.
- Gain knowledge about gstreamer.

### 4.2.2 Sprint 2

- Setup the Raspberry PI and record test sequences.
- Implement a simple change detection, whose quality isn't relevant at the moment.
- Write GStreamer plugin for the change detection.

### 4.2.3 Sprint 3

- Iteratively measure and improve the change detection against the criterias defined in 4.4 Measurement.

## 4.3 Milestones

The following milestones are critical for punctual project termination. If milestones are not reached, replanning of the whole project is required.

**Minimal change detection** Implementation of R1-T1 exists which doesn't have to meet G1+G2 yet, but it runs on a Raspberry Pi as a GStreamer Plugin (R1-T2).

**Final change detection** R1-T1 now respects G1+G2

**Delivery** All Results are delivered and meet all conditions.

## 4.4 Measurement

### 4.4.1 Scenes

To measure the success, we record the following scenes to repeatedly test them against the project goals:

| Scene | Camera | Scene |
|---|---|---|
| S1 | unchanged | unchanged |
| S2 | unchanged | person walking around |
| S3 | move horizontally only | unchanged |
| S4 | move vertically only | unchanged |
| S5 | bump* | unchanged |

For all those scenes, G1 and G2 are measured and calculated separately.
* Touch camera to shake the image for a short time

### 4.4.2 Method

All scenes are captured with the raspberry pi 3 and raspberry pi camera 2.1. In order to measure a percentage of falseNegatives, scenes with camera changes have to capture these changes exactly 100 times. This number of 100 occurrences is big enough for significant results and is still doable within the timeframe of this project.
The positions of expected changes is made in advance and compared with the results the program came up with to distinguish between false- and true positives.

Test sequences should be captured in RAW format to avoid compression errors. It's important, that the full field of view is used. If the amount of data requires tradeoffs, they should occur in the following order:

- Lower resolution
- Lower Framerate (minimum 10/fps)
- Crop images (reduce field of view)
- Use compression