Automatic Virtual Tracking in a Multi-Participant Scenario for a Mobile Device

Emelie Skoog
Elin Freberg

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6179
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2022

Abstract

The aim of this thesis is to identify a speaker in a video stream by only using the camera of a smartphone. This creates the opportunity for a mobile application to track the speaking participant during a conference call. Conference systems that include this feature use a microphone array to localize sound and who the speaker is. Mobile phones however, are generally not equipped with a microphone array and can therefore not recognize the source of the sound. Instead, this thesis will investigate how the speaker can be determined without using audio input. One method suggests that the speaker can be identified by detecting upper-body motion. This is based on the hypothesis that people often gesture while speaking. This method was implemented as a mobile application and the accuracy was tested. The application uses the camera of a mobile device for input frames and the built-in face detector to retrieve the position of the participants. The bounds of the upper-body for each participant were estimated by using the face bounds from the face detector. The application can then detect and compare motion by calculating the optical flow in each of these bounds. The participant with the most movement will be seen as the speaker. If movement is below a certain threshold value however the application will treat this as if no one is speaking. Lastly this approach was evaluated against a transcribed data-set with videos of two participants. Some variations were also investigated with the aim to increase the accuracy. The application had an accuracy of determining the speaker about 50% of the testing time. However, some alterations were implemented that did improve the results slightly, which indicates that a bigger parameter search could increase the result further.

Acknowledgements

This thesis would not have been made possible without the help of our supervisors. We would like to express our gratitude to our supervisor Sebastian Raase at Sony, who has shared so much of his knowledge with us. To our supervisor Richard Pates at the Automatic Control department we are very thankful for the guidance and support we have received. Lastly, thank you to Sony for giving us the opportunity to write our master thesis with you.

Thank you!

Contents

Acronyms

AMI	Augmented Multi-Party Interaction
API	Application Programming Interface
CMYK	Cyan-Magenta-Yellow-blacK
CSV	Comma-Separated Values
CV	Computer Vision
FPS	Frames Per Second
IDE	Integrated Development Environment
L-K	Lucas-Kanade
ML	Machine Learning
PTZ	Pan-Tilt-Zoom
RGB	Red-Green-Blue
ROI	Region of Interest
SSD	Sum Squared Difference
UI	User Interface
XML	Extensible Markup Language

1

Introduction

The constant digitalization of society has changed the way people communicate. Due to the recent pandemic, companies have had to change their way of working to adapt to challenges of social distancing. A more active use of digital meetings has been one of the solutions for communicating. Remote work is not only a way to ensure social distancing, but it also creates a more flexible workplace where less travel is needed. Instead, employees can participate from anywhere in the world. The equipment for digital meetings has evolved from just using a single microphone and web camera to more advanced conference systems. Many systems have the feature of identifying the speaking participant and focus the camera on that person to enhance the user experience. Creating this set-up requires special equipment, such as external cameras and microphones. In this paper however, the technique will be implemented using only a mobile device, providing the opportunity of creating such a conference room anywhere.

1.1 Problem Statement

Implementing the tracking technique on a mobile device comes with several challenges. One of these is how to actually identify the speaker using only hardware available in a smartphone. Conference systems on the market in general use a microphone array to detect where sound is coming from, and thus the speaking participant. Most mobile phones are however only equipped with a single mono or stereo microphone, which is insufficient for locating the source of sound in a 3D space. When implementing the technique of speaker identification on a mobile device, other data must be used, such as visual input. Processing images from the camera can nonetheless be computationally expensive, which can both drain the battery of the device and generate heat. The processing of frames must also be done in real-time and without unacceptable delays.

Research Questions

For this thesis the following research questions have been identified:

RQ1: Which methods for identifying the speaker can be implemented on a mobile device?

RQ2: To what extent does the chosen method identify the correct speaker?

RQ3: What are the advantages and disadvantages of the chosen method?

RQ4: What improvements can be done to the chosen method?

This thesis aims to research available methods for how to determine the speaker according to RQ1. From this research, one method will be implemented as an application and tested on a mobile device. The accuracy of the chosen method will be examined according to RQ2. Advantages and disadvantages of the chosen method will be investigated as stated in RQ3. Lastly, the thesis will discuss and suggest alterations which could improve the method, see RQ4.

Requirements

For a mobile device to be considered as an alternative to an advanced conference system, certain requirements have been defined for such an application. These are the following:

1. The application should rely only on the hardware of the mobile device.
2. The application should produce results that can be used for comparison.
3. The application should run in real-time with low latency.
4. The application should detect the correct speaker the majority of time.

Limitations

The scope of this project is to investigate the suitability to implement the technique of finding the speaker on a mobile device. This means that the application does not include a full tracking algorithm, instead limitations have been established. First, the feature of zooming in on the speaking participant is not implemented. This feature is relevant once the correct speaker is chosen most of the time and is therefore seen as future work. Secondly, the feature of people entering or leaving the frame is not implemented. This is also seen as future work since it does not improve the accuracy of the algorithm itself. The participants are also assumed to behave as in a meeting and be relatively stationary. Lastly the application will be tested on meetings with two participants. This was done because the reference data set used only contained two participants in the frame.

1.2 Approach

This thesis can be divided into three main parts; a literature study, an implementation and an evaluation. The literature study aimed to create a foundation for what methods could be used for the implementation. The findings of the literature study are presented in Chapter 2. Further, the implementation was based on one article from the study, *"The gesture is the speaker"* [11]. The article states that the speaking participant can be established by detecting movement, as the person with the most upper-body movement is anticipated to be the speaker. This assumption springs from the hypothesis that gesturing while speaking is a common human behaviour. The algorithm from the article achieved an accuracy of detecting the speaker in 85% of the time at best.

The method was implemented as an Android application that runs in real-time. It is based on the requirements and limitations presented in Section 1.1. The application shows a preview of the captured frames. Simultaneously frames are also processed to determine who the speaker is. This is done by calculating the motion of pixels between frames using the theory of Optical Flow, presented in Section 2.6. The results are written to files for later comparison. The comparison was done using a data set of pre-recorded meetings with corresponding transcripts. Several hypothesises for improvements of the application were also formulated and tested. The findings of the evaluation are presented and analyzed in Chapter 4.

1.3 Contribution

This paper evaluates one way of identifying the speaker during a conference call, using only a mobile device. The user will with the mobile application be able to set up a conference call with more advanced features anywhere, without additional external equipment. It provides a concrete example on how the speaker can be recognized by only using visual input.

The constructed framework should be seen as a starting point that can be used for implementing other methods that also rely on visual input. The evaluation step could be recreated to test changes done in the implementation, or to test other approaches. Further, this paper provides suggestions on other methods that could be used and ways to improve the chosen one. This is discussed in Section 5.2 where ideas for future work are presented.

1.4 Distribution of work

Throughout this thesis there has been constant collaboration between the authors. The literature study was divided by reading different papers, to then discuss relevant

theory to include. Most of the implementation was done in pairs, however Elin did most of the work related to the camera while Emelie implemented the image processing part. Emelie has been responsible for conducting the tests and Elin for compiling the results. The report has been written and edited by both authors. This distribution of work has made it possible for the authors to work in parallel whilst collaborating.

2

Background

This chapter will introduce previous work within the field of conference cameras and speaker identification. The research indicated that using both audio and visual input is a common approach, but this is not feasible on smartphones due to insufficient audio input capabilities. When using a smartphone, a better approach was to use a different method relying on visual input only. One of the approaches found was based on the theory that the speaker gestures. This is the approach that the application is built on. Theory behind this approach will be introduced in this chapter. This includes the theory of detecting motion with Optical Flow, and a method presented by Lukas and Kanade [16] for how to calculate it. Different methods for how to find points to track within a frame will also be introduced, including a method presented by Shi and Tomasi[24] that will be used in the application. Lastly, this chapter will introduce libraries and APIs which were used for the implementation.

2.1 Previous Work

This literature research focuses on different methods on how to both identify and track objects. When embarking on this project, different approaches were considered for how to find and track the speaker. Several difficulties regarding using a mobile device were identified, such as processing time and power consumption. However, the literature on the subject of finding a speaker using only visual input was limited. Instead, the combination of processing both visual and audio data on a computer was a more common approach.

The article *"Audio-Visual Data Fusion for Tracking the Direction of Multiple Speakers"* [19] presents an approach for tracking a speaker by using both audio and visual input. The proposed algorithm uses a camera and a microphone array for tracking the speaker. The microphone can localize the source of the sound and the camera can detect all faces in each image. The algorithm can then decide which face is closest to the source of the sound. By combining both inputs the algorithm can deal with absence in information. The sound localization can however suffer if there are

multiple people talking at once, or if there is a lot of background noise. A similar approach is presented in the paper *"Audio-Visual Speaker Tracking"* [14] where an audio cue is used in combination with a visual cue. Audio-visual tracking can be used with the combination of a particle filter and a Mean Shift algorithm. However, this is created for single participant use and additional framework is required if the frames include several participants.

An example of real-time tracking is presented in the article *"Real-Time Face Tracking and Replacement"* [7]. The article presents an algorithm that can track a face and replace it with a cartoon. The method of the paper is to apply the Haar-Like feature-based cascade classifier with modifications introduced by Lienhart et al[15] for face detection. The classifiers are pre-trained and can be used with OpenCV. For face tracking they use two different methods: Cam Shift and the L-K method[16]. The L-K method will also be presented further in Section 2.7. For simplification it is assumed that the video only contains one face. The paper concludes that the method of Cam Shift can handle larger movement, yet Optical Flow is the more systematic approach as it is better at tracking key points. It also determines that the efficiency of the algorithm is between 2 to 4 times higher if used on a computer compared to when used on a tablet. However, it can still be used in real time on mobile devices.

In the paper *"Low Vision Assistance Using Face Detection and Tracking on Android Smartphones"* [23] the focus is on how face detection and tracking can be used to help individuals with blind spots in their visual field. This is done with the hardware of the mobile device and by transmitting the images to a cloud-computer for processing. The technique uses face detection every few seconds to minimize the computational load and face tracking in every consecutive frame. The article investigates three different methods for face detection; Viola-Jones algorithm, Support Vector Machines and a method by Pai et al. Further it evaluates the following methods for face tracking: template matching, Lucas-Kanade tracking and Cam Shift. The results showed that the Viola-Jones face detection provided the best detection at acceptable power consumption and execution time. It was also determined that both the L-K and the Cam shift method offered acceptable performance, yet the L-K method was less prone to drifting.

"The Gesture is the speaker" [11] suggests that the speaker can be identified by detecting gestures. The article presents several arguments supporting this proposition. First, gesturing comes naturally to us. People born blind do gesture while speaking even though they have never seen this behaviour. Secondly, studies also show that gesturing occurs while speaking. However, while listening, people rarely gesture. Thirdly, using visual input does not generate disturbances when there is delayed auditory feedback. The article presents an algorithm which achieves an accuracy of detecting the speaker up to 85% of the time. The results however were inconsistent as they varied between each test. The algorithm did not handle real-time processing

either. Suggested improvements are to include audio and lip movements. The paper *"Look Who's Talking: Detecting the Dominant Speaker in a Cluttered Scenario"* [8] uses this presented method with a Canoical Correlation Analysis, CCA. This is done to minimize the probability of the algorithm reacting to movement that does not correlate to the speaker. By combining audio-visual input the algorithm can use a segment of the frame where the audio is located, to then select the dominant speaker in this pixel region.

The article *"Detection of Mouth Movements and its Applications to Cross-Modal Analysis of Planning Meetings"* [29] detects mouth movements to establish who speaks. The authors build a skin colour model to detect the face region. To detect the mouth, they create a mouth template. With this they preform image matching within the previously defined face area to locate the mouth. To determine whether the mouth is moving the article measure the change of the mouth area between two successive images with normalized cross-correlation coefficients.

Most of the papers presented use microphones to locate sound. The approach in this paper should however be implemented on a mobile application, that does not require external equipment. Therefore, other approaches that rely on input from the camera were considered. Both the paper *"The Gesture is the speaker"*[11] and *"Detection of Mouth Movements and its Applications to Cross-Modal Analysis of Planning Meetings"*[29] suggest that the speaker can be identified by detecting motion using visual input. One suggests upper-body motion and the other mouth movements. Because smartphones have built-in face detector it was decided that this would be used, to make the algorithm less computationally heavy. However, many face-detectors only find the bounds of the face. It can therefore be hard to find the exact bound of the mouth. On the other hand, the upper body could be estimated quite easily. Further, the most common ways proposed to track motion is by using the L-K method to calculate optical flow or to use Cam Shift. The paper *"Low Vision Assistance Using Face Detection and Tracking on Android Smartphones"*[23] states that the L-K method was less prone to drift and *"Real-Time Face Tracking and Replacement"* [7] states that it is better at tracking key points. With this as a background it was decided that the approach of the application was to find upper body movements by calculating the optical flow with the method presented by L-K.

2.2 Conference Cameras

Advanced cameras together with microphone arrays are a normal approach for conference calls as discussed in Section 2.1. One typical set-up of a conference system is presented in the paper *"A Hybrid Real-Time Face Tracking System"* [27]. These cameras usually have three degrees of freedom and are called PTZ cameras. PTZ stands for their features of panning, tilting, and zooming. The microphone arrays

can be used to determine in what direction the active speaker is, making sure that the camera is directed at the area of interest. The camera can then use face detection to ensure that it is the face of the speaker which is in focus when zooming.

2.3 Sony Xperia 1 III

Instead of using specialized conference equipment this thesis will create an application that will be tested on the Sony Xperia 1 III smartphone. It has three back-facing camera sensors with 12 megapixels each that are aligned vertically. The three different cameras have different lenses, where one is a normal wide-angle lens, one has an ultra-wide lens and the last one has a telephoto lens. The back cameras can record in 4K resolution (3840 × 2160 pixels) at a maximum rate of 60 frames per seconds (FPS), and in 2K resolution (1920 × 1080 pixels) with the maximum rate of 120 frames per seconds. The aspect ratio of the device is 21:9 and the 4K screen is 6.5 inches wide [25].

2.4 Digital images

Humans can easily identify objects visually. Yet, an image of a chair instead appears as a big array of numbers for an algorithm. This array is a numerical representation of a real image and is referred to as a digital image. This digital format enables computers to store and handle images. To create such an array the image is first divided into small picture elements known as pixels. The resolution depends on how many pixels the image is divided into. A higher resolution results in a more detailed image. Each pixel can be described with a number, or a small set of numbers, representing the intensity of colour in that area. These numbers are then organized into rows and columns that correlate to the horizontal and vertical position of the pixels. If the picture is in black and white it is common to use an 8 bit gray scale value to describe the intensity of lighting. This can be seen in Figure 2.1, where each pixel is represented by a number in the scale 0 to 255. Images in colour are normally represented by RGB (Red, Green, Blue) values which has three different scales instead. [28]

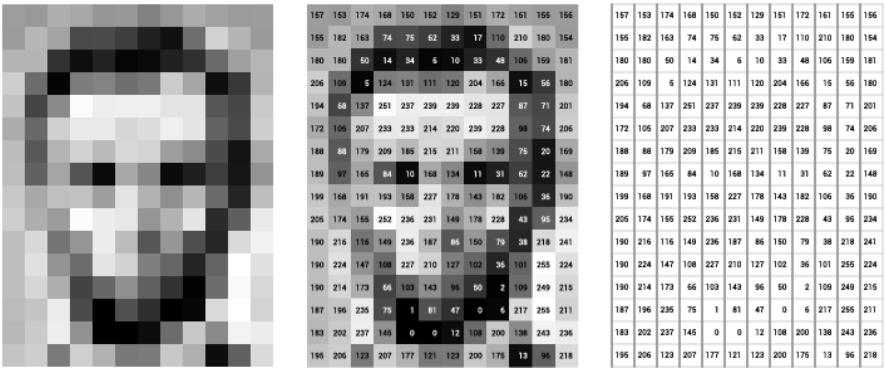

Figure 2.1 Image showing how each pixel is represented numerically by a gray scale value [1]

2.5 Computer Vision

For a computer to recognize objects in an image, it needs to analyze the image data and be trained to interpret it. Edges and curves in an image can be detected by comparing neighbouring pixels. If there for instance is a row of dark pixels and the neighbouring row of pixels are much lighter an edge has been detected. Computer Vision (CV) is usually based on pattern recognition which often uses machine learning, ML algorithm [22]. The algorithm is fed with an enormous amount of labelled data and patterns are detected between all images with the same labels. If the algorithm is fed with images of faces it might recognize a pattern of a circle (head) with two ovals (eyes) in the middle part of and an additional oval (mouth) in the bottom part. When the algorithm is fully trained and receives a new image of a face it recognizes this pattern and then also labels the image as a face.

2.6 Optical Flow

One way to detect motion in a video sequence is by using optical flow, which is defined as *"the apparent motion of individual pixels on the image plane"* [26]. As a point moves in the 3-dimensional room the motion can be projected onto the 2-dimensional plane and the motion of the brightness pattern can be tracked. The optical flow vectors u and v can be defined through Equation 2.1 by the displacement dx and dy of the pixel over the time dt.

[1] Source: `https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html`

$$(u,v) = (\frac{dx}{dt}, \frac{dy}{dt}) \tag{2.1}$$

To determine the optical flow, it is assumed that the brightness is invariant for the point between two following images and also that displacements are small. In Figure 2.2 there are two images taken at time t and $t+dt$, the point in the first image at location (x,y) at time t has the intensity $I(x,y,t)$. Between the two image frames the point will have moved by dx, dy, during time dt. The intensity for the second frame is then defined as $I(x+dx, y+dy, t+dt)$.

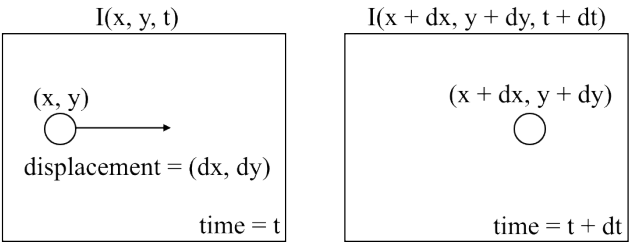

Figure 2.2 Point displacement [2]

With the assumption of invariant brightness however, the intensity of the point in the two images can be set to be equal as presented in Equation 2.2.

$$I(x,y,t) = I(x+dx, y+dy, t+dt) \tag{2.2}$$

As dx, dy and dt are assumed to be small, the right-hand side of Equation 2.2 can be approximated through a Taylor expansion neglecting the higher-order terms shown in the article *"Determining Optical Flow"* [13], see Equation 2.3.

$$I(x+dx, y+dy, t+dt) = I(x,y,t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial y}dt \tag{2.3}$$

By subtracting Equation 2.3 with Equation 2.2 and also dividing with dt it is obtained that:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial y}\frac{dt}{dt} = 0 \tag{2.4}$$

[2] Source: `https://nanonets.com/blog/optical-flow/`

If the subscripts $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$ and $I_t = \frac{\partial I}{\partial t}$ are used to denote the derivatives while Equation 2.1 are applied, the final equation can be written as in Equation 2.5.

$$I_x u + I_y v + I_t = 0 \tag{2.5}$$

Considering there is one equation with two unknowns, the equation cannot be solved without further assumptions.

Sparse and Dense Optical Flow

There are two different types of optical flow. These are sparse and dense optical flow. For sparse optical flow only a few pixels in each frame are processed. Computing a sparse optical flow is often done by using a method presented by Lucas and Kanade[16], which will be introduced in Section 2.7. When using a sparse flow, there are different methods for choosing what pixels to track since not all pixels will be analysed in between frames. These methods will be presented in Section 2.8. As seen in the left image of Figure 2.3, the sparse flow is presented as vectors for each point that is being tracked.

Figure 2.3 To the left a sparse optical flow can be seen, and to the right a dense one [3]

Dense optical flow calculates the flow for all pixels in the frame. In the right image of Figure 2.3, it can be seen that dense optical flow intensifies the pixels which are moving faster. This method leads to a more accurate result, as no point of interest is neglected. However, processing each pixel also causes dense optical flow to be more computationally expensive. For real-time applications, dense optical flow might not be the most optimal. Calculating the dense flow can be done through different methods. One method presented by Farnebäck from the article *"Two-Frame*

[3] Source: `https://nanonets.com/blog/optical-flow/`

Motion Estimation Based on Polynomial Expansion" [9] suggests analysing how the intensity of each pixel alternates between each frame using a polynomial equation.

2.7 Lucas-Kanade Method

Lucas and Kanade present one way of solving the optical flow equation in their article *"An Iterative Image Registration Technique with an Application to Stereo Vision"* [16]. In the article they suggest that instead of looking at the movement of a single pixel, a patch of neighbouring pixels can be analysed instead. The optical flow of each point $p_1, p_2, .., p_n$ in this path can be described by Equation 2.6 as:

$$\begin{aligned} I_x(p_1)u(p_1) + I_y(p_1)v(p_1) + I_t &= 0 \\ I_x(p_2)u(p_2) + I_y(p_2)v(p_2) + I_t &= 0 \\ &\cdot \\ &\cdot \\ &\cdot \\ I_x(p_n)u(p_n) + I_y(p_n)v(p_n) + I_t &= 0 \end{aligned} \tag{2.6}$$

Lukas and Kanade then made the assumption that the points in this patch would have the same optical flow, meaning that $u(p_1) = u(p_2) = ... = u(p_n)$ and that $v(p_1) = v(p_2) = ... = v(p_n)$. Equation 2.6 can then be rewritten as a matrix equation, see Equation 2.7.

$$\overset{A}{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ ... & ... \\ I_x(p_n) & I_y(p_n) \end{bmatrix}} \overset{x}{\begin{bmatrix} u \\ v \end{bmatrix}} + \overset{-b}{\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ ... \\ I_t(p_n) \end{bmatrix}} = 0 \tag{2.7}$$

x can then be derived as:

$$x = (A^T A)^{-1} A b$$

For this equation to be solvable, $A^T A$ must be invertible or well-conditioned. $A^T A$'s eigenvalues therefore need to satisfy $\lambda_1 \geq \lambda_2 > 0$. However λ_1 should not be significantly larger than λ_2 else this approach suffer from the aperture problem.

The Aperture Problem

In the book *"Encyclopedia of Neuroscience"* [3] the aperture problem is presented. It is described as the problem that occurs when a moving spatial structure such as an edge or bar is viewed through an aperture (an opening). In Figure 2.4 the bar has a motion to the right, which can be determined when looking through the lower aperture. In the upper aperture however, it would appear as if the bar was moving normal to the edge. This means that when the ends are not visible the motion of the object cannot be determined unambiguously.

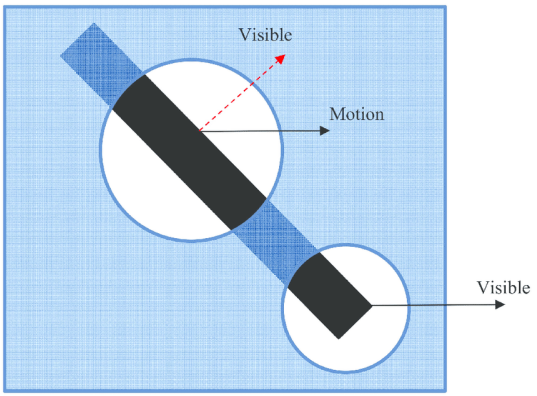

Figure 2.4 Visual explanation of the aperture problem [4]

Pyramidal Implementation of The Lucas-Kanade Method

Sometimes objects move fast, and the earlier assumptions of small displacements are false. When talking about small displacements it is assumed that a point in the first image will have only moved to a neighbouring pixel in the next image. When tracking a moving object in high resolution however, points might move several pixels away. Larger displacements invalidate the Taylor series approximation in Equation 2.3, which means that Equation 2.5 is no longer valid either.

This issue can be solved with a pyramidal implementation of the L-K method suggested in the article *"Pyramidal implementation of the L-K feature"* [4]. Denote that the original frames have a resolution of *NxN*, and the displacement between the two consecutively frames are *m* pixels big. If new images with a lower resolution of $\frac{N}{2}x\frac{N}{2}$ were to be computed the distance *m* would be halved. By continuing to lower

[4] Source: `https://www.researchgate.net/publication/309172033_A_Possible_Role_for_End-Stopped_V1_Neurons_in_the_Perception_of_Motion_A_Computational_Model`

the resolution the motion m will eventually by small enough for Equation 2.3 and Equation 2.5 to be valid. This procedure of lowering the resolution by half is called a pyramid representation of an image. The original full resolution image is then the zeroth level image and the highest level of the pyramid can be denoted L_n. For each time the resolution is halved the pyramid level is increased by 1, see Figure 2.5.

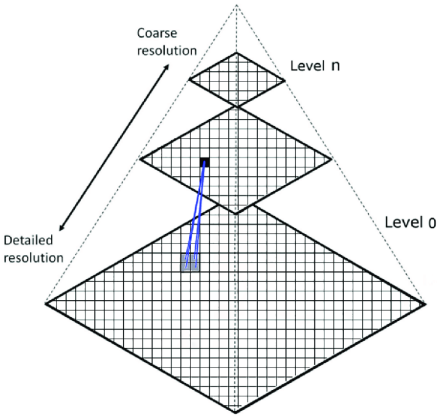

Figure 2.5 Pyramid representation of an image [5]

A simple way to compute the new resolutions is to take each 2x2 window and calculate the average colour value of this window. The calculated value can then be used for the new image, as illustrated in Figure 2.6.

[5] Source: `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0195290`

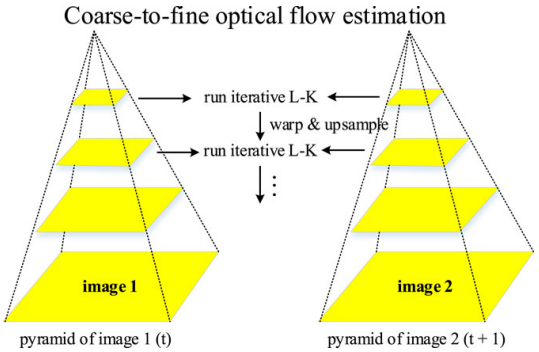

Figure 2.6 Illustration of how the optical flow is calculated using the pyramidal implementation of the L-K method [6]

With the lower resolution images the optical flow can now be calculated. The approach is to first calculate the flow of the deepest pyramidal level L_n. The result is used to make a warp image. Image warping will transform the image so that positions will be mapped from one level to positions in the next level. The warp image will be created to the upper level $L_n - 1$, given this image the optical flow is calculated at level $L_n - 1$. The computed flow is used for a new warp image at level $L_n - 2$. This goes on until the zeroth level has been reached (the original image).

2.8 Feature Tracking

As stated in Section 2.6, the choice between using sparse or dense optical flow might come down to processing time. While quality will be lower, it could be beneficial to only track some of the pixels in each frame with a sparse flow. Further, there could be a certain region of interest of motion, and the optical flow must hence not be determined for the whole frame. Below different suggestions how to choose what pixels or points to track will be presented.

Grid Placement

A simple way to decide what points to track is to choose a grid pattern for the whole frame, or for a region of interest, ROI. The only parameters used for such a method are how many points should be placed on the certain region, or what distance to keep in between them. An example of how a distribution of points in a grid layout can be done is shown in Figure 2.7.

[6]Source: `https://www.frontiersin.org/articles/10.3389/fenrg.2022.764903/full`

Figure 2.7 Grid placement of points to track [7]

Harris Corner Detection

There are also more sophisticated ways to choose what points to track. One method presented in the article *"A combined corner and edge detector"* [12] is to identify corners in a frame. This is one way of trying to find important feature in the frame. A corner is commonly perceived as the junction of two edges. The advantage of choosing a corner for tracking, is that it can be easier for an algorithm to distinguish it in the frames that follows, and thereby to track the corner.

The main approach of the Harris Corner Detection is to first define a corner as a point in an image that have two dominant edges. The edges must have different direction. Then the differential of the score between the corners and edges are calculated. The score is then used as a reference to the direction of the object to determine this difference.

To be able to detect a corner, every unique window around each pixel in an image needs to be identified. The windows can be measured by shifting it in a given direction, to then register changes between every pixel. This is done by computing the sum squared difference (SSD) between the pixel before and after a shift. This is done to find windows with large SSD in all directions, which is then a corner. The SSD is determined with the use of Equation 2.8.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2, \tag{2.8}$$

[7] Source: `https://www.edge-ai-vision.com/wp-content/uploads/2019/03/Football-512x288-1536x864.png`

where E refers to the sum of the SSD for every u, v which refers to coordinates x, y for every pixel in a window. As a high value for the SSD indicates a corner, the Equation 2.8 is to be maximized, which is done through simplifications. An approximation is done firstly with Taylor expansion. The expansion is presented in matrix form in Equation 2.9.

$$E(u,v) \approx [u,v] \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}, \tag{2.9}$$

From Equation 2.9 it is the summed matrix that is presented in Equation 2.10 that is to be maximized, also called the structure tensor of the SSD.

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}. \tag{2.10}$$

The eigenvalues for the structure tensor can be used for finding the directions for the smallest and largest increases in SSD. A score based on the eigenvalues is then calculated. The value of the score is then used to determine if the window consists of a corner that is to be tracked. For the Harris corner detection the score R is expressed as in Equation 2.11.

$$R = \lambda_1 \lambda_2 - k(\lambda_1 - \lambda_2)^2 \tag{2.11}$$

where λ_1 and λ_2 are the eigenvalues of the summed matrix. The value of k is based on observations that should be in the interval $k \in [0.04, 0.06]$. The score R will determine what classification the window should have. A high score will indicate high eigenvalues, and the window is then assumed to be a corner while a low score will represent a flat window. If the score is negative, one eigenvalue will be high and the other one low, indicating that the window is an edge.

Shi-Tomasi Method

In the article *"Good Features to Track"*, Shi and Tomasi [24] presents a few alterations to the Harris Corner Detection. The alterations provide a more distributed way of identifying corners. An example of how the result can vary between the method of Harris Corner Detection and the one presented method by Shi and Tomasi can be seen in Figure 2.8.

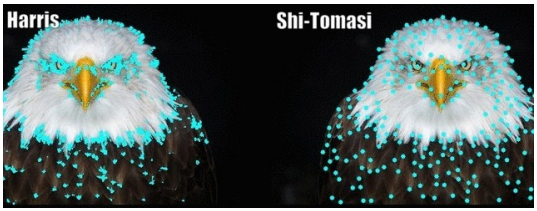

Figure 2.8 Results of the Harris and Shi-Tomasi method for corner detection [8]

The method that Shi and Tomasi presents follows the first steps shown for the Harris Corner Detection. The main difference is regarding how the score is calculated. The new way of calculating the score that Shi-Tomasi propose is presented below in Equation 2.12

$$R = min(\lambda_1, \lambda_2) \tag{2.12}$$

When the score is higher than a certain value the window will be seen as a corner. These thresholds values are illustrated in Figure 2.9.

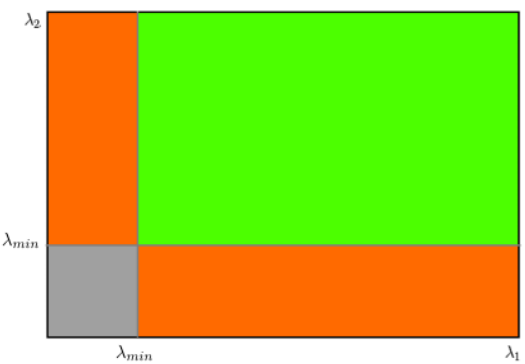

Figure 2.9 Eigenvalue regions for Shi-Tomasi method where green bounds represents corners, orange bounds represents edges and gray are flat areas [9]

[8] Source: `https://medium.com/pixel-wise/detect-those-corners-aba0f034078b`

[9] Source: `https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html`

2.9 Body Proportions

In image processing, the whole frame is rarely of interest. Instead, one might be interested in a certain region of interest, ROI. This application aims to recognize gestures; therefore, the region of interest is the upper body. It is in this region where points should be placed and tracked. As the phone has a built-in face detector, a simple estimation can be done to also find the bounds of the upper body. Body proportions are usually measured using the head as a reference. An adult is approximately 7,5 heads tall as seen in Figure 2.10. The distance from the top of the head to the belly button is approximately 3 heads high[10]. A simple approach to finding the upper body bounds is therefore to use the bounding box of the face, and then multiply its height by three.

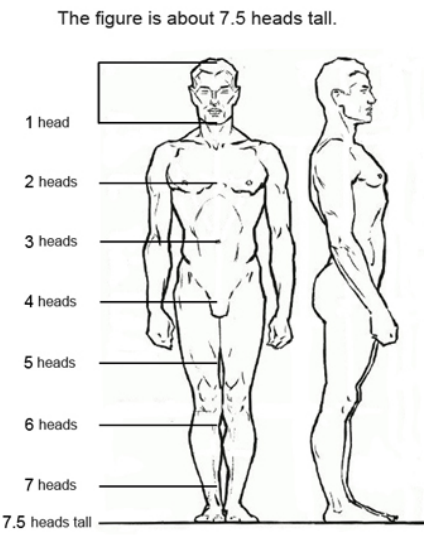

Figure 2.10 Body proportions [10]

2.10 Android Studio

Android Studio is a platform for development of applications that can be used on Android units. It uses the IntelliJ integrated development environment, IDE, and a Gradle software building system. The Gradle tool is used for automatically building

[10] Source: `https://www.thedrawingsource.com/figure-drawing-proportions.html`

projects. It does this by creating links in the project to then package and compile the code. This is sufficient also when the code consists of different files as source code in Java and Kotlin, as well as resources such as XML files used for UI. Every project consists of a manifest. The manifest is also an XML file that includes all important data for the project. This can be data such as requirements, versions and permissions for the project which simplifies the building of the application. Another common concept which is widely used in Android is callbacks. Callbacks are a function which are passed as an argument to another function. They are then triggered and executed when a specified event occurs [18].

Android Hardware API

The Android Platform consists of different packages, one of them being the `android.hardware` package. The package can assist the user with features for the hardware of the device, such as cameras and sensors. As not all devices have the same hardware, an application has set up requirements that the device must fulfil to use it.

Camera2 API

The Camera2 is one of the packages included in the Android Hardware API. It contains features for all internal and external cameras that the device is connected to. The package Camera2 uses a `CameraManager` instance for listing all available cameras of the device. Every connected camera will be a `CameraDevice`. The properties and settings of the `CameraDevice` can be reached with the `CameraCharacteristics` instance [2]. How different instances operates in the Camera2 API is illustrated in Figure 2.11.

The instance `CaptureRequest` uses a `Capture` callback. This creates a capture request for a given `CameraDevice`. As seen in Figure 2.11 every request will be linked to a target Surface. The request can be a single one, or a repeating request. Requests are handled in the order they are made, in a request queue that every `CameraDevice` has. However, the repeating requests will have a lower priority. The output of each request will be a queue of in-flight requests, as seen in Figure 2.11. From each request the output frames can be accessed. The frames will be linked to the target `Surface` earlier defined in the `CaptureRequest`. A surface can then be used for preview, photo capturing and video recording [5].

A `OnCaptureComplete` callback will be invoked when all meta data of the capture is available. With the instance `CaptureResult` defined in the callback the statistics from the single capture request can be accessed [5]. The statistics includes a list of all faces detected in one frame, as `Face` objects.

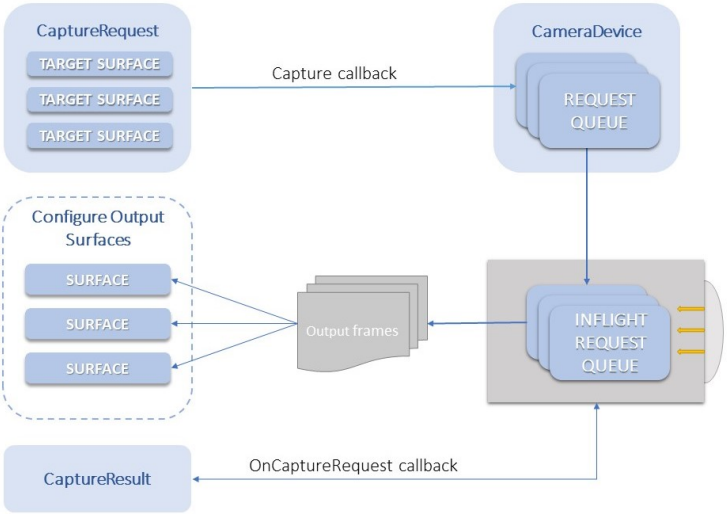

Figure 2.11 Illustration of how different objects and methods for Camera2 operates

The `Face` object consists of information such as ID, bounds and placement of eyes and mouth. It also provides a score for how certain the detection is. If the certainty value is less than 50 percentage, the `Face` is recommended to be removed. The data retrieved for the face detection depends on what mode for the detection is used. The modes include a full one where all metadata such as the face rectangles including mouth and eye position, scores, face ID and landmarks. It also includes a simple mode where only rectangles and their confidence values are included or an off mode which disables face detection. Many mobile devices today only support the simple or off mode [6].

2.11 OpenCV Library

OpenCV is a library created by Intel which was built to enable developers to create applications with features such as ML and CV. The library consists of algorithms to identify faces, detect movements of objects, and produce high-quality representations of a scene for example. It supports different platforms, one of them being the Android platform [1].

In OpenCV images are stored as `Mat` objects which is a single or multi channelled

matrix. The OpenCV library contains a method for calculating the optical flow in between frames using the L-K method. The input of the method is two `Mat` objects (two consecutive frames), a matrix of the position of the points to track and parameters for the L-K algorithm. The parameters are used to determine which pyramidal level of the L-K method to use, criteria for how many iterations to do, size of the search window and operational flags. The flags can be seen as error measures to choose thresholds for when a point should be removed or not considered. If a point has move to far or cannot be localized the flags will determine the outcome. The output of the method will be the coordinates of the points in the second frame and a status vector which states if the points were found or not [21].

The library also consists of a method of determining corners in a frame using the Shi-Tomasi method presented in Section 2.8. This input of the method will regard things such as quality level and number of pixels in between corners. The output will be a matrix of points that are to be seen as the corners or important features of the frame [20]. These features will then be tracked with the L-K method as discussed previously.

3

Method

The method includes an implementation of the theory and an evaluation of it. The implementation is an Android application that allows the user to see a preview of the captured images in real-time. With the use of a processing thread, the motion of each participant is analyzed and compared. The idea is to detect the person with the most movement. As the frames are processed, the bound with the most optical flow is determined to belong to the speaker. The results are written to a file for later comparison. The evaluation was done by capturing frames from a screen previewing videos from an existing data set. The environment for the testing was kept constant, for valid comparison. The AMI data set used for testing includes data such as pre-recorded videos with corresponding transcripts for each participant. These transcripts were compiled to a single file and then compared to the file produced by running the application. To compile and compare files, a Python project was created. The Python project also includes scripts for visualizing the results.

3.1 Implementation

The implementation is an application created in Android Studio. The application consists of several classes for handling UI and processing. How these classes relate to each other will be explained in Section 3.1 and further it can be illustrated in Figure 3.1 and 3.2. When running the application, it connects to the back camera of the device and a preview is shown on the screen. The preview includes bounds for each participant and the points that are being tracked. Simultaneously some frames are processed to decide who is talking when.

System Design

The application is divided in different classes which handles different proceedings. All code related to the camera is collected int the `CameraActivity`. The `CameraActivity` then provides the `UI` with frames for preview as well as it also stores frames and its associated face data in the `Monitor`, see Figure 3.1. The

`Monitor` was created to store all shared resources. This enables the `ProcessingThread` to retrieve data in a thread safe way. The aim of the processing is to calculate the mean optical flow within the upper body region for each participant. Therefore, each person is also represented by an `OpticalFlow` object. Each object tracks its respective points. To place these within the right region of interest the `Bounded Region` class was created. It is the `ProcessingThread` which creates the `OpticalFlow` objects and compared their mean value to each other. The Participant with the biggest motion (if it is above a certain threashold value) is believed to be the speaker, this is written to the `result file`. All parameters associated to the application is stored in the `Param` file, for easy access. This design is presented in Figure 3.1 and Figure 3.2. Below is a more in depth explanation of each class.

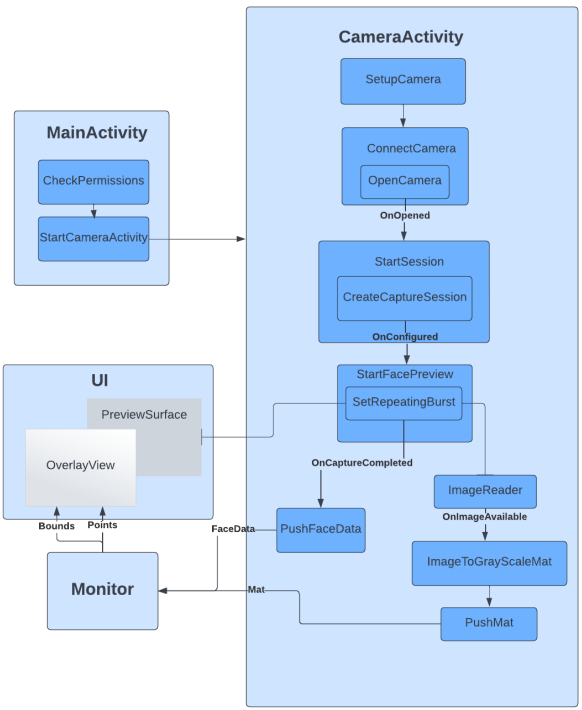

Figure 3.1 Flowchart of the application showing the MainActivity, the CameraActivit and the UI classes.

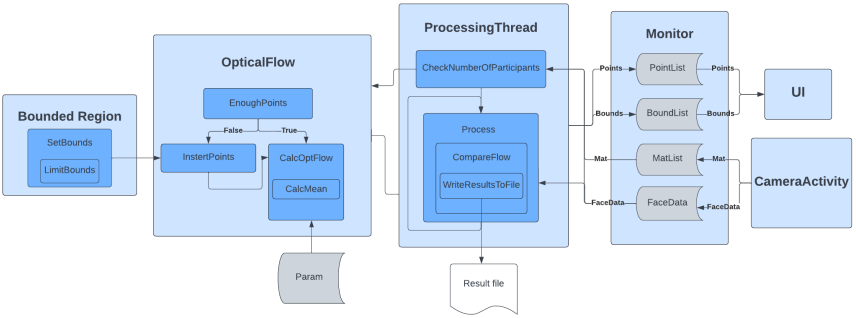

Figure 3.2 Flowchart of the application showing the Monitor, the processingThread, the OpticalFlow and BoundedRegion classes.

MainActivity

The application is invoked in the `MainActivity`. For the application to function properly it needs permissions from the user to access the camera as well as writing to the external storage of the device. These permissions are handled in the class `MainActivity`. With the use of pop-up windows, the user is asked to approve all required permissions. When all permissions are approved, the class will start the `CameraActivity` with the method `StartCameraActivity`.

CameraActivity

The `CameraActivity` relates to all methods used for the camera. The CameraAPI presented in Section 2.10 is used for accessing and setting up the back camera of the device. The most significant methods of the `CameraActivity` can be seen in Figure 3.1. With the method `SetUpCamera` information is retrived about the different cameras of the device. Further, it retrieves the characteristics of the chosen camera, for instance in what resolution the camera can capture frames. The method `ConnectCamera` connects to the chosen camera with the method `OpenCamera`. This method has a callback as argument. When the camera is connected the callback function invokes the `OnOpened` method. This method then calls the `StartSession` method to create a camera session. A camera session is created by providing output targets. The session is linked to a surface in the UI, which is used to preview the captured frames and an `ImageReader`, which receives the frames that are to be processed.

Once the configuration is done, the callback `OnConfigured` calls the method `StartFacePreview`. In `StartFacePreview` different captures are requested. This is done with the request `SetRepeatingBurst`, which allows each capture to have a different target surface, as described in Section 2.10. In

this case every frame is sent to the preview surface, while every n^{th} frame is also sent to the `ImageReader`. These are the frames which are to be processed. When an image is available for the `ImageReader` a call will be made for the `OnImageAvalible`. The image is the transformed to a gray-scaled `Mat` with `ImageToGrayScaleMat` before it is pushed to the `Monitor`. In the `Monitor` class, every frame will be kept in a list, the `MatList`. The corresponding face data is also pushed to the `Monitor` and kept in another list called the `FaceDataList`. This data is received when a Capture has been completed and the `OnCaptureCompleted` method has been invoked.

Monitor

The application uses different threads for processing and retrieving images. The threads are to acquire the same resources. It is therefore important to protect the shared resources, to ensure that they are not accessed and modified simultaneously. This is done with the use of the class `Monitor` which is a thread safe class. As already mentioned, the frames and face data are stored in lists which corresponds to the `MatList` and the `FaceData` list. The monitor also stores data of what bounds each participant has in the `BoundList` and what points are being tracked in the `PointList`. The list of points and the bounds are created in the `ProcessingThread` and then pushed to the monitor. They are then used by the `OverlayView` by pulling the lists from the `Monitor`. The list of `Mat` and `Faces` are pushed from the `CameraActivity` and polled from the `ProcessingThread`. All of these calls are made with mutual exclusion.

ProcessingThread

The frames are processed by the thread `ProcessingThread`. When the first frames are retrieved, the thread creates an `OpticalFlow` instance for each participant in the meeting. Each participant is mapped together with a corresponding `OpticalFlow`. The `OpticalFlow` instance refers to the motion of pixels as presented in Section 2.6.The method `rocess` is called repeatedly for calculating and comparing the Optical Flow of each participant. The calculations are done in each Optical Flow instance by calling on the `CalcOptFlow` method. The comparison of the Optical Flow, and thereby the motion, for every participant is done with the `CompareFlow` method. This method establishes which participant that has the largest movement, or if no one is moving enough for it to be seen as gestures. If none of the participants are moving above a certain threshold, the method will determine that no one is speaking. The results from the comparison will be written to a CSV file on the storage of the device; the `Result File`, with the method `WriteResultsToFile`.

OpticalFlow

Every time the Optical Flow is calculated, the method `EnoughPoints` will check if the number of points are above a certain threshold. If there are enough points, the Optical Flow can be calculated using the pyramidal implementation of L-K presented in Section 2.7. This is done in the method `CalcOptFlow`. The flow is calculated by locating the new position of the points from the previous frame in the new frame. If a point cannot be found in the new frame, it will be removed. The mean displacement of the points from the previous frame to the next one is then calculated. The displacement will be sent back to the `ProcessingThread` for the comparison described above. If the number of points are below the threshold, new points will be chosen. The points are chosen within a given area of the frame with the Shi-Tomasi[24] approach presented in Section 2.8. This is done in the `InsertPoints` method, and the area is determined with a call to the method `SetBounds` in the class `BoundedRegion`.

BoundedRegion

The class `BoundedRegion` has methods for creating regions of interest. By using the position of the face and its height a region for the upper body can be estimated, according to the theory of body proportions presented in Section 2.9. This is done in the method `SetBounds`. To ensure that the bounds are not outside the frame the method `LimitBounds` is used.

Param

The application consists of several parameters that can be tuned. All parameters are stored in the class `Param` for easy access. With easy access, all the parameters can quickly be changed. All parameters are presented in Table 3.1.

Table 3.1 Parameters used in the application

Parameter	Description
MAX_CORNERS	Maximum amount of corners which should be found
MIN_CORNERS	Minimum amount of corners before points are to be updated
QUALITY_LEVEL	Minimal acceptable eigenvalue for Shi-Tomasi equation, see Equation 2.12
MIN_DISTANCE	Minimum distance points are allowed to be chosen
BLOCK_SIZE	Size of block for every co-variation matrix that is computed for a pixel neighborhood
USE_HARRIS_DETECTOR	If true the Harris detector is used, if false Shi-Tomasi is used
WINDOW_SIZE	The size of a search window for every pyramidal level
MAX_LEVEL	Maximum pyramidal level
MIN_MOVEMENT	Threshold value for if movement is classified as a gesture
SKIPPED_FRAMES	Number of frames which are not processed
FRAMES_BEFORE_SWITCHING	Value for how many frames which shall pass with a certain participant being the speaker before the application changes to this speaker
NUMBER_OF_PARTICIPANTS	Set value of participants in the meeting

CSV File

The result of who is speaking when is written to a CSV file. The format is designed to match the files which are to be used for testing. When a participant is speaking, their ID is written to the file as well as the start time. Once the speaker changes, the end time of the dialogue is also written to the file. When a new person has been identified as the speaker the corresponding speaker ID as well as the start and end time is written to the next line in the file. If the movement is below a fixed threshold no one will be considered as the speaker. This will also be written to the file with a fixed ID indicating that there is No Speaker.

User Interface

The user interface, UI, consists of a surface for the preview of the collected frames and a transparent `OverlayView`. The `OverlayView` is placed on top of the preview surface where information and results can be drawn. The points which are

tracked, and the regions of interest are polled from the `Monitor` and drawn on this view. This view is used for a visual understanding of how the points are behaving, and it gives a visual verification that all participants are found.

3.2 Testing

To test the application, results are compared to a chosen data set. The idea was to capture frames with the mobile device from a pre-recorded video with an associated transcript. The data set used was the AMI Meeting Corpus that consists of recordings from a meeting setting [17]. This data should correspond to the environment intended for this application. More information about this data set can be found in the section below. Considering some alteration was to be investigated, it was important to keep constant conditions for valid comparisons. The tests were therefore conducted in a closed environment. The environment was created with a cardboard box combined with a piece of fabric to cover the opening. The set-up inside the box consisted of a screen to display the video and the phone attached to a tripod. Both the screen and the tripod were placed on specific marks made on the cardboard box. The set-up can be seen in Figure 3.3.

Figure 3.3 Set-up for testing the application

AMI Meeting Corpus

To test the application the Augmented Multi-Party Interaction, AMI, data set was used. How the data set was created and the parts it includes are presented in the ar-

ticle *"The AMI Meetings Corpus"*[17]. It consists of one hundred hours of recorded videos of both mocked scenario-driven and real meetings. The videos are recorded in a set-up that should recreate the surroundings and conversations of a real meeting. An example of the set-up can be seen in the snapshot from one of the recordings in Figure 3.4. The rooms used for the data set were mounted with equipment such as microphones, cameras and white board captured slides. Besides input from the equipment the data also includes speech transcripts, dialog acts and summaries.

Figure 3.4 Snapshot from AMI meeting corpus

Result Processing

To compare the results from running the Android application with the data set, a Python project was created. A flowchart for the project can be seen in Figure 3.5. The project consisted of scripts for CSV conversion, CSV comparison and creating charts.

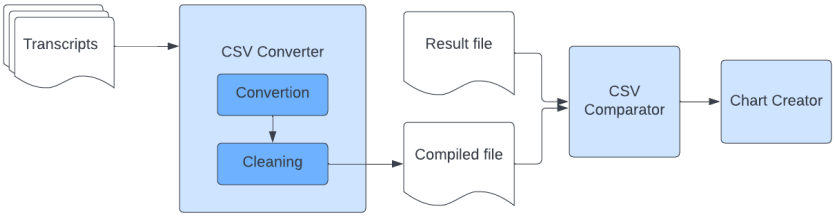

Figure 3.5 Flowchart of evaluation scripts

The transcripts from the data set are XML files with all spoken segments of a given participant. This means that every person in the meeting has a designated file for all words spoken during that certain meeting. These files will be the input to the CSV Converter script as seen in Figure 3.5. The XML files are converted to one single CSV file for every video sequence used for the tests. The conversion consists of changing the format of the file, the format of the speaker and the format of the speaking time. Lastly data cleaning was done on the combined CSV file. This was due to errors found regarding the end time being lower than the start time for some spoken sequences. The output of the CSV converter will then be a CSV file with cleaned data from all the XML files. This will be the compiled file seen in Figure 3.5.

When a CSV file has been created the comparison can be done. The script for CSV comparison will then be used. The input will be the compiled CSV file and a CSV file with the result of running the Android application. Every row of the CSV file with results will be compared with the other file to find the total amount of time the correct speaker was chosen. For the data set, overlapping between speakers can occur. When there is an overlap, the choice of the application will be considered correct regardless. The application however can only choose one speaker at a time.

To visualize the results, different charts can be drawn. In Python there are libraries which can convert CSV files to what is known as data frames. Python then has libraries for visualizing these data frames, which was used to plot results. The plots included are:

- A timeline of who is chosen as the speaker, an example of this can be seen in Figure 4.5
- A bar chart of the division of speaking time, an example of this can be seen in Figure 4.2

- A scatter plot of how the accuracy depends on a specific parameter, an example of this can be seen in Figure 4.12.

Metrics

To evaluate the application the results, need to be quantified. The metric used to tell the accuracy of the algorithm was a percentage of how often the correct person was identified as the speaker. Each test also tracks the mean processing time. This is important as it both impact the battery consumption and as it can cause delays in the processing step. The investigation can therefore be divided into the two main following metrics: accuracy and processing time.

Accuracy

As the application is running it writes the results to a CSV file. These files are then compared to the data set, as described above. The script calculates a percentage of how often the application chooses the right speaker. There are three possible options for who the speaker is; one of the two participants or none of them.

Processing Time

The mean processing time is calculated as the algorithm will keep track of how many frames it has processed, and the total run time. Each time a processing step of the algorithm is invoked, a timer will be started. When the processing is done the timer will be stopped. The duration will then be added to the total processing time and the number of frames processed will be increased. When the test is finished the meantime will be calculated as total run time divided by number of frames processed. With a frame rate of 60 FPS a new frame is available every 16.6 milliseconds. This means that if the processing time is above this a delay will be accumulated. As mentioned above only every n^{th} frame is processed to avoid latency. If the processing is up to 1000 milliseconds the frame rate can only be 1 FPS.

4

Evaluation

This chapter will present and discuss results from the testing the application. The first part of the testing was to evaluate the application. This included three different tests. First, the application was tested on the same video sequences several times. This was done to investigate if there was an element of randomness to the results. The conclusion from this test was randomness that could not be neglected. Therefore, throughout the testing phase each test was conducted 5 times on each video sequence. Each sequence were 8 minutes long and were retrieved from different meeting with different participants. Secondly, the algorithm was tested on three different video sequences from the data set to see how results varied in between different sequences. When testing the application on different sequences the result also varied from mean accuracy of 44% on one sequence to a mean accuracy of 56% on another. Lastly the set-up was tested in order to validate that similar results were produced when using the application in a real scenario. The result indicated that the previously chosen set-up with a constant environment gave slightly better results. Therefore, it was kept for the remaining part of the testing. To then improve the results, different hypotheses were to be tested. These are presented below. The results did not improve significantly when testing the different hypothesis, and some of them were even rejected. What did seem to improve the results however was using a lower resolution, and with this use finer parameters to achieve a higher quality for point distribution.

4.1 Hypotheses

Possible improvements to the application were based on hypotheses set up regarding how the method could give better results. The hypotheses were the following:

- **H1**: *A grid layout for placing the points will improve the results and reduce processing time. The points will be distributed over the bounded area including the upper body and the frame will not need processing for finding points using the Shi-Tomasi method.*

- **H2**: *A higher value for how long a participant needs to speak before switching the speaker will give better results, as short gestures would be ignored.*
- **H3**: *A higher value of how much a participant needs to move before it is seen as gesturing can improve the result as this determines how often "No Speaker" will be chosen.*
- **H4**: *Only using the face bound will minimize the processing time and also focus more on mouth movement. Allowing finer parameters to be used to improve the results.*
- **H5**: *A faster frame rate could favor the L-K calculations as displacements are assumed to be small.*
- **H6**: *By lowering the resolution processing will be faster. A higher frame rate can then be used to improve the result.*

4.2 Element of Randomness Test

When implementing the processing algorithm, it was noted that points could be chosen differently between runs when using the Shi-Tomasi [24] method. Points are also lost and updated every now and then. Further, both the application and the video sequence are started manually, and tests are therefore started at different milliseconds compared to one another due to human error. This can all result in element of randomness. Therefore, before testing the method, the consistency of the results needed to be evaluated. When conducting these tests all the parameters were constant and the same sequence was used.

Results

Table 4.1 below shows the percentage of how often the algorithm chooses the correct speaker. In Appendix A there are charts for a segment of each test, illustrating who the algorithm choose at what time.

Table 4.1 Results for element of randomness test

Test	1	2	3	4	5
Correct Speaker	43.30%	43.64%	43.71%	43.98%	45.72%

Discussion

Table 4.1 shows quite similar results, yet with small variation. However, when looking closer on the charts in Appendix A it can be seen that the chosen speaker varies between every test. If there was no element of randomness, more identical charts

would be expected. This test therefore indicates that the results will vary with a certain percentage, and that the element of randomness cannot be neglected. This is believed to be caused by the circumstances presented in the introduction above. Because of this, each video sequence will be tested 5 times in the tests that follows.

4.3 Application Test

To evaluate how well the algorithm works it was tested on three different video sequences from the data set. Considering there was an element of randomness, a single test was completed five times for each video sequence.

Results

Table 4.2 below shows the mean percentage of accuracy for each sequence, and the mean processing time. Figure 4.1 is a snapshot from one of the test to illustrate how the points are distributed. Figure 4.2 shows a comparison of the decision made by the algorithm and the real data of who speaks when. Lastly Figure 4.3 shows how the decision is divided between Speaker 1, Speaker 2 and No Speaker for both the transcriptions and for the results of running the application. These charts will be used for comparison in later tests. In later tests only sequence 1 and 2 will be tested due to time limitation.

Table 4.2 Results for application test

Sequence	Correct Speaker	Processing Time [ms]
1	44.07%	854.62
2	46.64%	780.75
3	56.35%	836.14

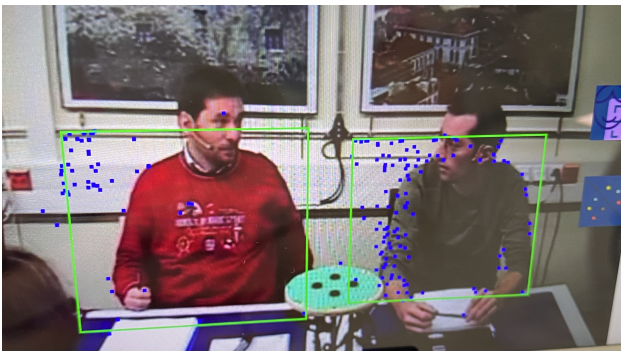

Figure 4.1 Example for how points are distributed in method test

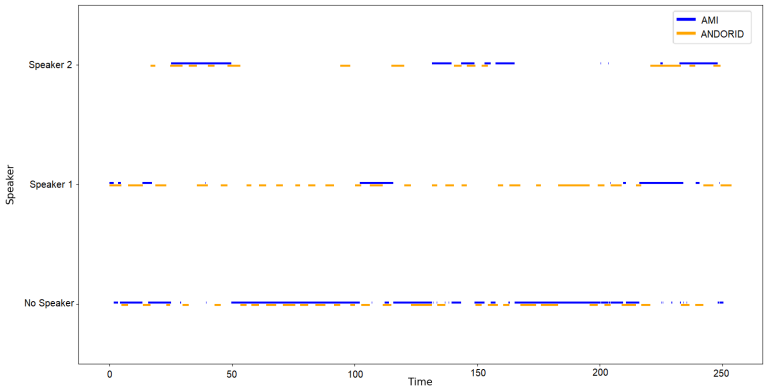

Figure 4.2 Timeline of who the speaker was according to the application (yellow) and who actually spoke according to the transcript (blue)

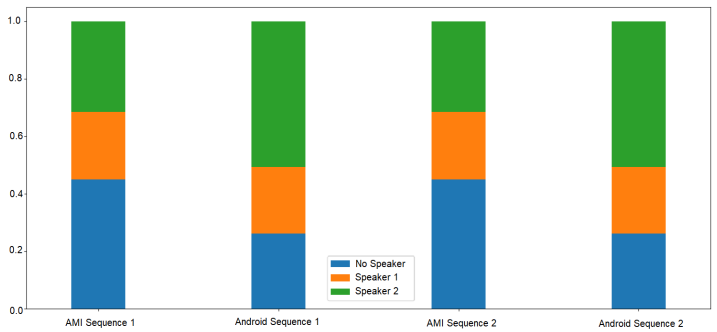

Figure 4.3 Speaker chosen for sequence 1 and 2

Discussion

When conducting the test above some thoughts came to mind. First, when looking at the overlay view in Figure 4.1 and how the points where distributed, it was noted that the points often where stuck to the background or in other unfavourable positions, and not on the body. When looking for body gestures the main goal would be to track hands and arms, or even the face. Using the method presented by Shi-Tomasi [24] points can be positioned anywhere. Since the method finds the "best corners"

these could for instance be placed in the background. If a person then moves, and all points are in the background these movements will not be noticed. In Figure 4.1 below one person has points around his upper body, yet the other persons points are in the background. Because of this, one possible improvement could be to use a grid layout when placing the points. This would not find corners, but it would guarantee that there are points placed on the arms and hands area. Corners are considered easier to track though, as they stand out to the background. For a more advanced implementation the best way would be to include a computer vision algorithm which can find specific body parts. This could however take a lot of computational power, resulting in a higher processing time.

When looking in the results from Figure 4.2 it is notable that the algorithm switches who speaks more often than the data set does. The algorithm has a value for how many frames should pass for a certain speaker before it switches. This value could be investigated. If this value is increased, less switches would be expected. However, the algorithm could also chose the wrong speaker for a longer period with this change.

The results also show that the application has a quite high processing time compared to how often a new frame is available, which is every 0.2 second. If the processing of a frame is not finish in time to when the next frame is retrieved, it will lead to a delay in the system. This implementation therefore only process every n^{th} frame to avoid this issue. However, could a higher frame rate give better results? In Section 2.6 it is stated that there is an assumption that displacements are small. A lower n would result in less movements between each processed frame. One suggestion is therefore to investigate how a higher frame rate impacts the results. However, if the frame rate is increased and the processing time is high this could cause delays.

One thing to keep in mind is also that the application is tested on two participants, and thus two optical flows are calculated. If the meeting includes more participants, the processing time is believed to increase due to more optical flow calculations.

One way to speed up the processing time is to change the resolution of the processed frames. It would give the algorithm less pixels to process. Another hypothesis is that fewer points result in faster processing. This means that the algorithm might be faster if a smaller area is used. One suggestion then is to only look at the face or mouth of each participant. Tracking the face instead of the body might also be better in the sense that people sometimes are placed close to, or in front of each other in conferences or meetings. Considering the mouth moves when someone is talking this could also be an interesting investigation area since some people tend to gesture less.

4.4 Live Test

The set-up for the tests above was mainly chosen to keep a constant environment. This resulted in a set-up which captures a video. This test, however, aims to provide an environment more close to how the application is meant to be used. This to see if the set-up has an impact on the result. When the set-up is used the resolution is only as good as the video quality.

For this test two persons are engaging in a conversation in front of the camera. The two participants are not aware of what is to be tested. The conversation is then transcribed and compared with the output of the application. This new set-up that was tested can be seen in Figure 4.4.

Figure 4.4 Figure showing the new set-up that were to be tested

Results

The results of the two tests conducted are presented below in Table 4.3 which shows the mean accuracy of when the right speaker was chosen and also the mean processing time.

Table 4.3 Results for set-up test

Test	Correct Speaker	Processing Time [ms]
1	37.28%	969.40
2	37.05%	941.36

Discussion

The results of this test were worse than for the test conducted with the video sequences. The transcription was made manually and may therefore include human errors which can impact the results. Also, the transcriptions could only be made with the accuracy of a second, compared to the transcripts from the data set which had the accuracy of a millisecond. However, since there were no big improvement from when capturing frames from a video the previous results are to be seen as plausible.

4.5 Placement of Tracking Points Test

This test relates to hypothesis **H1**:

- *A grid layout for placing the points will improve the results and reduce processing time. The points will be distributed over the bounded area including the upper body and the frame will not need processing for finding points using the Shi-Tomasi method.*

This test investigates if a grid layout could improve the results. The argument for using a grid instead of the Shi-Tomasi [24] method is that it will guarantee that the points will be evenly spread out over the speaker as discussed above.

Results

Table 4.4 shows the mean percentage of accuracy for each sequence and the mean processing time, for the two different methods: the Shi-Tomasi method and a grid layout.

Table 4.4 Results for placement of tracking points test

Sequence	Point Placement Method	Correct Speaker	Processing Time [ms]
1	Grid Layout	43.04%	615.11
1	Shi-Tomasi Method	44.07%	854.62
2	Grid Layout	44.61%	564.40
2	Shi-Tomasi Method	46.64%	780.75

Discussion

The hypothesis that a grid layout would work better did not show to be true. The reason for this could be that the points are not updated often enough. Other reasons could be that the algorithm loses more points considering it does not track corners.

It could then be more difficult for the L-K method to find the points in the next processed frame. The method does however have a shorter processing time. This is believed to correlate to the fact that points are placed faster with a grid layout, as it does not require searching for corners to choose.

4.6 Switching Speaker Test

This test will investigate the hypothesis **H2**, which states:

- *A higher value for how long a participant needs to speak before switching the speaker will give better results, as short gestures would be ignored.*

As mentioned earlier, the application has a given value for how many frames must pass before a new participant is chosen as the speaker. This value is used to filter out small movements which are not considered to be gestures. This test investigates how increasing the number of frames impacts the result.

Results

Table 4.5 shows the mean percentage of accuracy for each sequence and the mean processing time. All parameters are kept constant, except for the value of how many frames should pass with a new speaker, before the algorithm changes to this person. The original value of 2 frames was used when testing the method, is compared to results when using a value of 4, 6 and 8 frames before switching frames. Figure 4.5 shows the result for each single test and in Figure 4.6 - 4.8 there are charts of how the decisions were made for a segment of each sequence. These can be compared to Figure 4.2 which show the speaker decision for the initial value used, which was a value of 2 frames.

Table 4.5 Results for switching speaker test

Sequence	Frames Before Switching	Correct Speaker	Processing Time [ms]
1	2	44.07%	854.62
1	4	44.65%	825.97
1	6	40.09%	890.14
1	8	39.08%	860.89
2	2	46.64%	780.75
2	4	48.19%	741.24
2	6	43.80%	801.03
2	8	41.92%	794.98

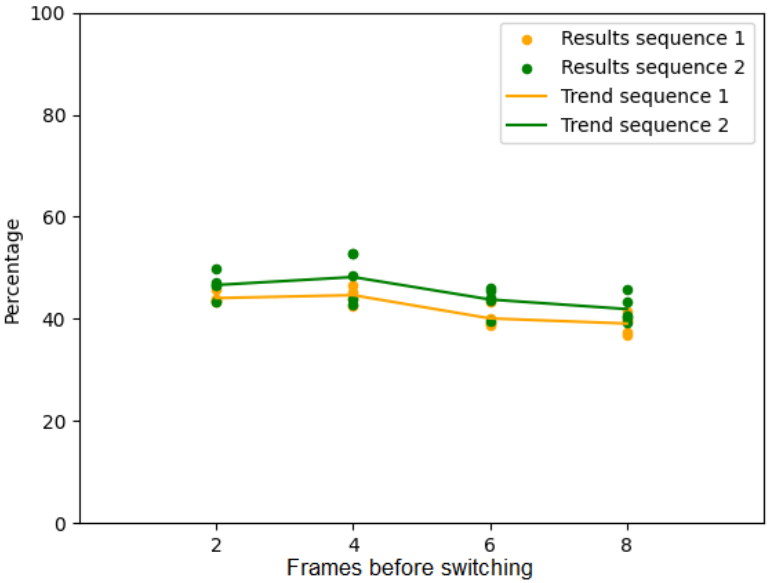

Figure 4.5 Results for a value of 2, 4, 6 and 8 frames before switching the speaker, for switching the speaker test

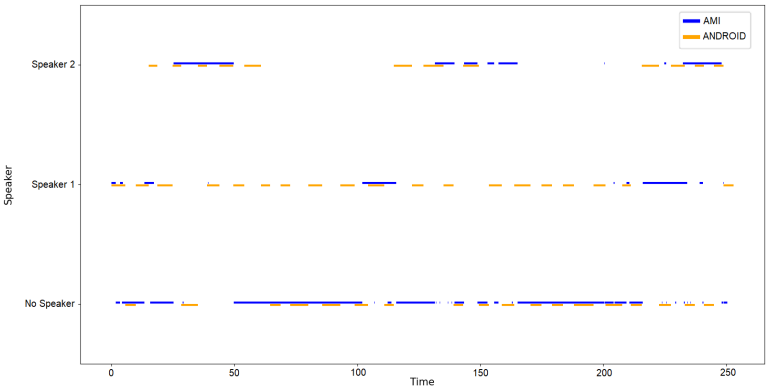

Figure 4.6 Segment of speaker decision (yellow), for switching test with value of 4 frames before switching the speaker compared to the decision of the data set (blue)

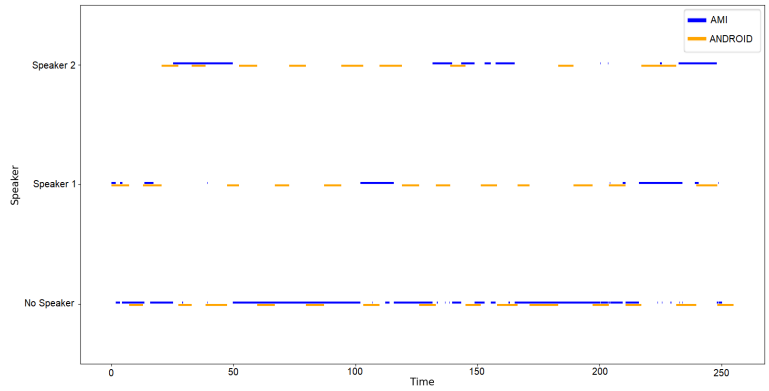

Figure 4.7 Segment of speaker decision (yellow), for switching test with value of 6 frames before switching the speaker compared to the decision of the data set (blue)

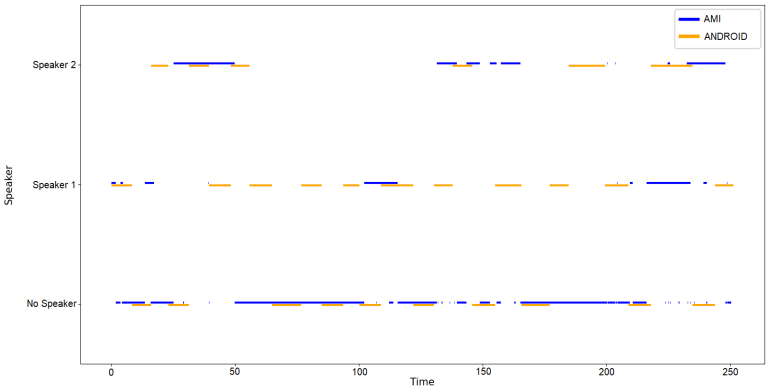

Figure 4.8 Segment of speaker decision (yellow), for switching test with value of 8 frames before switching the speaker compared to the decision of the data set (blue)

Discussion

The results show that a small increase of the value of frames before switching the speaker improves the result slightly. When looking at the charts in Figures 4.6 - 4.8 it can be seen that the algorithm changes less often compared to the original value. However, the algorithm still changes more often than the data set yet increasing the value more does not indicate a better result.

4.7 Movement Test

In this test the threshold for how small movement there could be before the algorithm determine that no one is speaking will be tested. This relates to the mean value of how many pixel the points for one participant has moved, which is presented in Table 3.1. This relates to the following hypothesis, **H3**:

- *A higher value of how much a participant needs to move before it is seen as gesturing can improve the result as this determines how often "No Speaker" will be chosen.*

Result

Table 4.6 below shows the mean percentage of how often the algorithm chooses the correct speaker, and the mean processing time. The threshold value of a movement of 4 pixels was used in the method test, for the algorithm to determine that there was is no speaker. It was the compared to a value of 6, 8 and 10 pixels. All other

parameters are kept constant in the test. Figure 4.9 illustrated how the speaker choice was done and the charts 4.10 and 4.11 shows how the decision is divided between No Speaker, Speaker 1 and Speaker 2.

Table 4.6 Results for movement test

Sequence	Pixels Moved	Correct Speaker	Processing Time [ms]
1	4	44.65%	825.97
1	6	44.09%	882.56
1	8	45.24%	896.10
1	10	44.37%	841.62
2	4	48.19%	741.24
2	6	47.77%	772.52
2	8	49.11%	720.28
2	10	47.67%	706.39

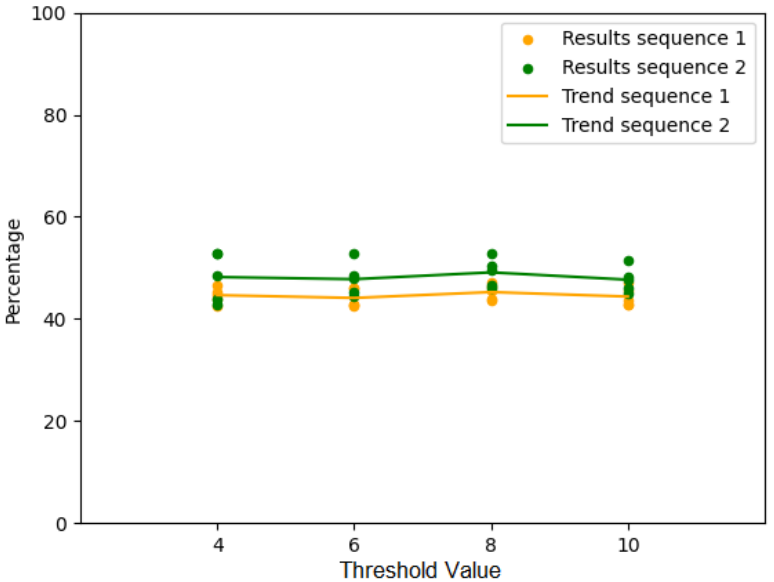

Figure 4.9 Results for movement test, for a mean threshold value of 4,6,8 and 10 pixels moved

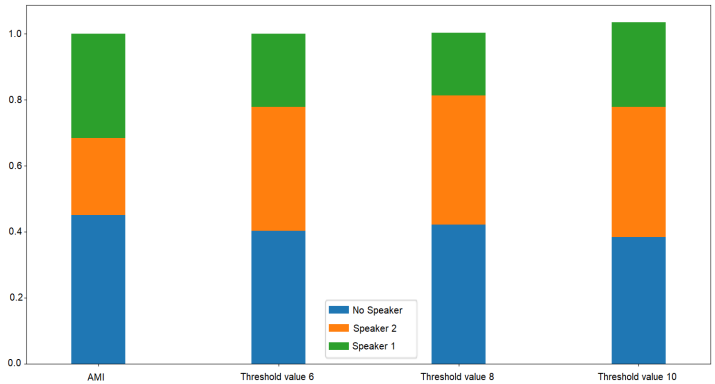

Figure 4.10 Speaker chosen during movement test for sequence 1, for a mean threshold value of 6, 8 and 10 pixels moved compared to the data set

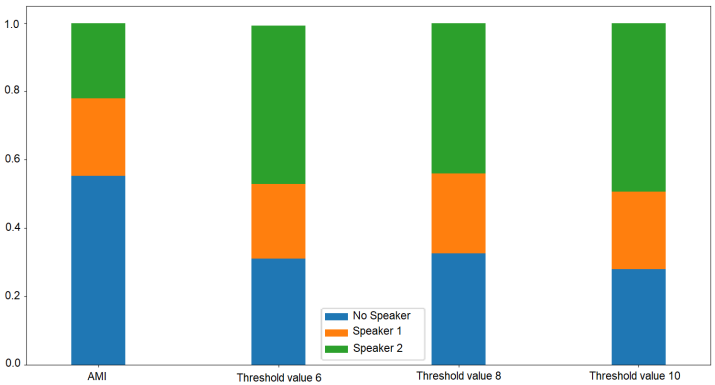

Figure 4.11 Speaker chosen during movement test for sequence 2, for a mean threshold value of 6, 8 and 10 pixels moved compared to the data set

Discussion

In this test, the value for movement was solely increased, since earlier result indicated that the data set had a higher percentage of No Speaker. When looking at Figure 4.10 and Figure 4.11 the first chart has a quite similar percentage of No Speaker

whilst the other has a much lower. This might indicate that some people gesture more than others. Instead of relying on the speaker to move a certain amount the visual input could be combined with audio input. Then the decision of No Speaker could be done using the microphone of the device instead.

4.8 Bound Method Test

The bounds for each speaker were chosen upon the belief that gesturing occurs within the bounds where the body of the speaker will be. In this test the hypothesis is that better results could be achieved by using a smaller bound, only consisting of the face of the speaker. This relates to hypothesis **H4**:

- *Only using the face bound will minimize the processing time and also focus more on mouth movement. With this, finer parameters can be used to improve the results.*

Results

The results of the bound tests are presented below in Table 4.7 where using body bound can be compared to using only a face bound.

Table 4.7 Results for bound method test

Sequence	Bound Method	Correct Speaker	Processing Time [ms]
1	Body	45.24%	896.10
1	Face	42.91%	1087.92
2	Body	49.11%	720.28
2	Face	47.32%	900.67

Discussion

The hypothesis for this test was rejected. When using the smaller bound this led to less accurate result and a longer processing time. What was noted when conducting the tests was that the points were lost and therefore also updated more often. As the same frame rate is used but for a smaller bound, points can more easily end up outside the bound for the next processed frame. Therefore, a higher frame rate could be one way of improving the results, if this would lead to more points being found in between frames. Another impact of using the face bound could be that there is not enough movement for the face area, or too much movement while not speaking. This approach does not support the theory of gesturing, only on face movement. Perhaps a person is moving their face a lot when not speaking, or a person could end up not moving their face enough when speaking.

The higher processing time is believed to be a direct cause of the higher updating rate of points, as using the Shi-Tomasi method is time consuming. It is more favourable for the processing time for the update of points to occur less often. However, the Shi-Tomasi method should take less time to perform as a smaller area is used when distributing the points, but if the method is invoked more often the total processing time might still not be shorter. With the higher processing time finer parameters cannot be used, as this would make the processing time longer which can lead to delays. Therefore, changing to this approach did not allow for using finer parameters and trying to improve the quality of the method.

4.9 Frame Rate Test

One explanation for the poor results could be that the frame rate in the original application is quite low. In this test the frame rate will be increase to investigate if this can affect the results. This according to the hypothesis **H5** which suggests:

- *A faster frame rate could favor the L-K calculations as displacements are assumed to be small.*

Results

Table 4.8 below shows the mean percentage of when the algorithm chooses the correct speaker and the mean processing time for a frame rate between 20 and 10 skipped frames. It also shows the mean list size when the application stopped running. The size of the list represents the number of frames which have not yet been processed. Figure 4.12 shows the result for each test.

Table 4.8 Results for movement test

Sequence	Skipped frames	Correct Speaker	Processing Time [ms]	List Size When Stopping
1	20	45.24%	896.10	1
1	15	44.39%	923.22	2.4
1	10	44.27%	982.80	79
2	20	49.11%	720.28	1
2	15	48.42%	743.13	2.2
2	10	46.01%	790.24	63

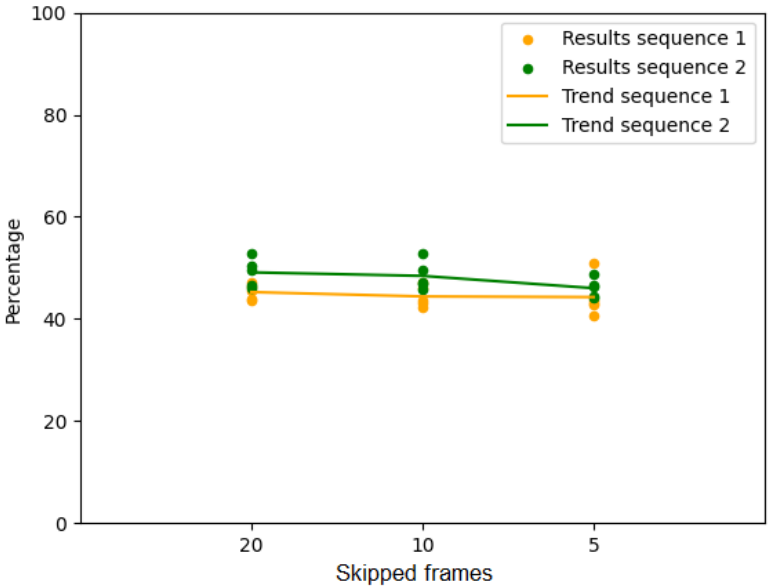

Figure 4.12 Results for frame rate test

Discussion

When conducting the test above it was noted that a high frame rate lead to delays. When the list size is larger than 1 it is not the most recent frame which is processed. In other words there is a correlation between the list size, and if there is a delay in the system. The results above show that a frame rate of 20 has a list size of 1 when the tests are stopped. That means that there was no delay. However, tests with a frame rate of 10 resulted in a list size of around 60. This means that the frames being processed were captured about 10 seconds ago. This could be an explanation to why a frame rate of 10 results in a lower accuracy. A frame rate of 15 does not show considerable delays. However these delays might become bigger the longer the application will run. Using a frame rate which just manage to process a frame might not be the best practice considering the mean processing time tends to differ between video sequences. In one meeting a certain frame rate might work, but in the next this frame rate might cause delays. This test therefore showed that a frame rate of 20 both gave the best results, but also did not cause delays. It is the processing time which indicates what frame rate is possible, and if a higher frame rate is desired then the processing time needs to be decreased by changing other parameters.

4.10 Resolution Test

As concluded in the test above, the only way to use a higher frame rate would be to decrease the processing time. One way to do this could be to lower the resolution. With a lower resolution, the results might improve according to the hypothesis **H6**:

- *By lowering the resolution processing will be faster, and a higher frame rate can be used which can improve the result.*

In this test the resolution was reduced by half. Instead of using a resolution of 3840 × 2160 pixels a resolution of 1920 × 1080 pixels was used.

Results

The results of the resolution test are presented below in Table 4.9. It shows the mean accuracy and mean processing time when using 20, 10 or 5 skipped frames. Figure 4.13 shows the accuracy for each run. The results can be compared to the results for a resolution of 3840 × 2160 pixels, presented in Table 4.8.

Table 4.9 Results for resolution test

Sequence	Skipped Frames	Resolution	Correct Speaker	Processing Time [ms]
1	20	1920 × 1080	37.22%	501.10
1	10	1920 × 1080	38.74%	546.51
1	5	1920 × 1080	44.02%	629.56
2	20	1920 × 1080	41.64%	424.38
2	10	1920 × 1080	43.04%	471.52
2	5	1920 × 1080	46.29%	503.69

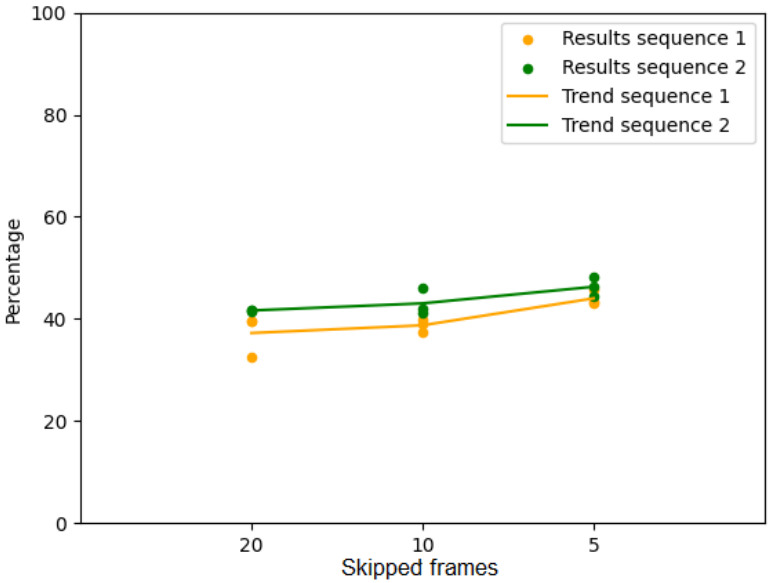

Figure 4.13 Results for resolution test

As a lower processing time achieved, a higher value for the quality level for the Shi-Tomasi method could also be tested. The result is presented in Table 4.10 and compared to results using full resolution (3840 × 2160 pixels).

Table 4.10 Results for resolution test

Sequence	Skipped Frames	Resolution	Quality Level	Correct Speaker	Processing Time [ms]
1	20	3840 × 2160	0.01	45.24%	896.10
1	5	1920 × 1080	0.1	46.67%	700.84
2	20	3840 × 2160	0.01	49.11%	720.28
2	5	1920 × 1080	0.1	51.56%	597.45

Discussion

The hypothesis that a higher frame rate would give better result showed to be true. As the processing time was decreased, this offered the opportunity to use finer parameters. Using finer parameters can improve the results but is not possible to do

if the processing time is too high as this will cause delays. In this test the only parameter that was tuned was the quality level used when distributing points. Using a higher quality did improve the results.

This test also proved that if fewer frames are skipped, the results are improved. The frame rate test, see Section 4.9, did not indicate this. However, as delays occurred when using a higher frame rate at full resolution, conclusions cannot be drawn in the same way as in this test. No delays occurred for a lower resolution, and therefore a higher frame rate gave better results.

4.11 Summary

From the testing it was noted that there was an element of randomness to the results. This is believed to relate to the fact that points can be distributed and lost differently in between tests. The fact that the element of randomness could not be neglected resulted in a more limited testing of different parameters, since every test had to be run multiple times to confirm the output. When testing the application on different sequences from the data set, the results also varied greatly in between sequence. This was firstly believed to relate to the fact that the amount of time that no one was speaking also varied between the sequences. When using a higher threshold value for how much movement any of the participants should have for the algorithm to determine that someone was speaking, this only improved the results slightly, however. From testing the algorithm, it was also noted that the speaker switched much faster than for the data set. Therefore, the value for how many frames should pass before switching the speaker was increased. This also gave minor improvements for the results.
A main conclusion from conducting all tests were that trade-offs were a big part of it. When improving one parameter, a trade-off for another had to be done. With a lower resolution the accuracy did deteriorate, yet the processing time lowered. A lower processing time gave room for other improvements to be done. With a lower processing time the quality of the point distribution could be increased. Even though a lower resolution was used, the higher quality gave the best results as for all the test completed. This resulted in the conclusion that a much larger search for the optimal parameters could improve the accuracy even further.

5

Conclusions

In conclusion, this thesis did show that there was some truth to the theory that the speaker could be recognized by finding movements. The results however did not show accuracy high enough for the algorithm to be used as input for tracking a speaker. The accuracy is believed to improve if more parameters are tuned. In this section the findings, methods and results of this paper will be reflected upon. Future work such as introducing audio input, finding optimal parameters and improving the processing will be presented.

5.1 Reflection

The aim of this thesis was to answer the defined research questions. First, different approaches for implementing the feature of identifying the speaker on a mobile device were investigated. The use of audio-video input is a common approach, but the hardware of a mobile device does not always support it. Instead, different methods that solely rely on visual input were investigated. The findings indicated that the L-K method could be a systematic approach. With the L-K method, motion could be found within different regions of the frame, containing the upper body of each participant. The speaker was then determined by comparing the movement within every bound, with the assumption that the speaker would be gesturing and therefore be the participant with the largest movement.

The implemented method was later tested on an existing data set. Results were produced determine how often the correct speaker could be identified with the chosen approach. In the given set-up the results indicated that there was an element of randomness, both between sequences but also when the same sequence was tested several times. With the given set-up, the speaker was identified in 44-56% of the total test time. A different set-up was also tested, to see if a set-up more like the usage that the application is created for impacted the results. However, capturing

frames from a video resulted in a better accuracy. This approach could therefore be validated and be used when testing possible improvements for the application.

The advantage of the approach was that it did not require any external hardware. With the use of open-source code, no model had to be trained. However, the processing was time consuming, and some changes led to delays when running the algorithm. This could however also be due to an insufficient code in the implementation. The processing algorithm could then be more effective if the code was improved and all steps of the processing was evaluated further. The conclusion is that it was difficult to simply refine parameters to obtain a better result, as it was constraint by the processing time.

Several tests were conducted to investigate improvements as the last research question state. The algorithm could be improved with a higher threshold value for how much movement there needed to be for it to be seen as a gesture. Also increasing the value slightly for how many frames should pass before the algorithm changes the speaker gave better results. Further, using a lower resolution did lead to a lower processing time. This allowed for using finer parameters. Only the quality level in the Shi-Tomasi parameter was changed with the lower resolution, yet it was proven that a higher quality did improve the results. This led to the indication that tuning other parameters could also improve the results.

As presented above the article *"The Gesture is the speaker"* [11] achieved an accuracy up to 85%. However, the results from the article also vary a lot in between video sequences. The same variance were also shown in test 4.3 Application test. One major difference between the implemented application and the algorithm presented in the article, is that the article does not process images in real-time. The processing time has caused limitations to several parameters considering it is done in real-time. As already mentioned, it is believed that the accuracy can be improved with better parameter values. If the processing time did not act as a limitation, perhaps better results as in the article could have been achieved.

5.2 Future Work

Mobile phones have evolved quickly over the past years and will continue to do so. This means that cameras and other hardware of the device could improve further. As mentioned above, Android phones have built in face detectors. Such a face detector cannot only give the bounds and position of the face, but also the bounds of the eyes, nose, and mouth. This might be the standard on future phones. With a more advanced face detection, it could be investigated if a similar application could find mouth movements and, in that way, identify the speaker. As discussed in Section 2.1

there are also algorithms for detecting mouths. With the use of ML, one could train a model for detecting both the mouth of the participants, and for it to react to a mouth moving. Simply using a bounded region for the upper body, or the face allows for a participant to make movement which is not correlated to speech. If a more advanced model was trained, such movement could be neglected, and the model should then only react to mouth movement. If mouth movement was the target, the belief is that a higher frame rate would be needed so that the algorithm would react to small changes.

This also correlates to gesturing. Perhaps the theory of gesturing is not always applicable and reacting to mouth movements or sound localization are more reliable methods. However, to recreate the results achieved in the article "The Gesture is the Speaker" should be possible. Therefore, it is believed that the results depends on the created algorithm and limitations of the mobile device for the processing step, and as results are measured in a different way.

To minimize the processing time, several changes could be made in the approach. There are other methods for face tracking that should be investigated. Even though the L-K method is favourable as it is better at tracking key points, using for example Cam Shift could lead to less processing. Such a method would not require finding key points, but the performance could suffer considering drifting could occur. Comparing different methods for calculating the mean movement for each participant would be a favourable next step for the implementation. This as different methods for finding key points are evaluated, yet not different methods for optical flow or other method for calculating movement are evaluated.

The approach of completing the processing on the mobile device could be investigated. Perhaps, using a cloud-computer as described in Section 2.1 could result in a lower processing time. This would then offer the opportunity of a finer processing of each frame. This must however be tested, to ensure that compromising and sending the frame to such a computer is in fact faster than simply doing the processing on the device. Yet, if an L-K algorithm could be done 2-4 times faster, as described in Section 2.1, such an approach could be beneficial and should be tested. However, this would in fact change the requirements of the application, as a cloud-computer would be necessary. The requirement of using only the hardware of a mobile device would then not be met.

The algorithm is tested in a very limited way. For future work it would be of interest to conduct longer tests, and for other video sequences. To draw conclusion from this limited amount of testing is not scientific enough, and to further prove the results from the algorithm more tests should be conducted. Conducting more tests is also of importance as there was an element of randomness detected in the testing stage. However, the project included several time-consuming parts such as a substantial

literature study and the implementation of the application. If a larger testing were to be done, this could have resulted in a simpler application if time had to be distributed differently.

If further work is done on the application, more parameters should be investigated. Different combinations of parameters could give better results. In the resolution test, a lower resolution gave a much lower processing time. This led to the opportunity of invoking finer parameters. When improving the quality for distributing points the results did also improve. Hence one thought is that a finer tuning of other parameters could improve the results further. One way of finding the most favourable parameters is to do a hyper parameter search. However, there are a lot of parameters, and a single test has a long running time. If the process could be automated, and tests could run independently, such a search could be made possible. Another issue however is the processing time. If it ends up being above a certain value, delays will occur as seen in the frame rate test. Therefore, the processing time must first be lower so that finer parameters can be used without delays occurring. As the processing time was lower for a lower resolution this could be investigated further, to see if an even lower resolution with finer parameters could give a higher accuracy. For an automated process the goal would then be to find the most optimal parameters that had such a processing time that delays did not occur.

Another improvement for this application, mentioned above, would also be to use the microphone of the phone. Even if the microphone cannot be used to localize sound as in audio-video application, it could be used for determining when there is No Speaker. If the audio input were to be under a threshold value that was found through testing, the algorithm could use this input to determine that there is No Speaker. This presumes that the meeting is held in an environment without disturbing background noise. Implementing this will not enhance the algorithms way of choosing between the participants, but it could improve the results for the choice of No Speaker. Such an implementation would require extra processing time, as for example one thread of the application would have to analyze the audio input. However, the frames which are determined to have No Speaker would not need visual processing. Such an implementation could therefore lead to a lower processing time overall and possibly a higher accuracy.

If the application does recognize who the speaker is with a high accuracy, there is more implementation to be done regarding the UI. Most importantly the zooming feature needs to be implemented. Then the application could not only recognize who is speaking, but also focus on and track the speaker. This implementation will need a control system in order for the view to change focus in a way to enhance the user experience.

Bibliography

[1] *About OpenCV*. 2022. URL: https://opencv.org/about/. (Accessed: 2022-03-07).

[2] *Android Hardware Camera2*. 2021. URL: https://developer.android.com/reference/android/hardware/camera2/package-summary. (Accessed: 2022-02-08).

[3] Marc D Binder, Nobutaka Hirokawa, and Uwe Windhorst. "Aperture Problem". In: *Encyclopedia of Neuroscience*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 159–159. ISBN: 978-3-540-29678-2. DOI: 10.1007/978-3-540-29678-2_310. URL: https://doi.org/10.1007/978-3-540-29678-2_310.

[4] Jean-Yves Bouguet. "Pyramidal Implementation of the Lucas-Kanade Feature Tracker". In: *Intel Corporation, Microprocessor Research Labs* (2000).

[5] *Camera Capture Sessions and Requests for Camera2*. 2021. URL: https://developer.android.com/training/camera2/capture-sessions-requests. (Accessed: 2022-02-09).

[6] *Camera2 Face Param*. 2021. URL: https://developer.android.com/reference/android/hardware/camera2/params/Face. (Accessed: 2022-02-09).

[7] Qi Cao and Ruishan Liu. "Real-Time Face Tracking and Replacement". In: *CS231M Stanford Project* (2015).

[8] Eleonora D'Arca, Neil M Robertson, and James R Hopgood. "Look who's talking: Detecting the dominant speaker in a cluttered scenario". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 1532–1536. DOI: 10.1109/ICASSP.2014.6853854.

[9] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Image Analysis*. Ed. by Josef Bigun and Tomas Gustavsson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370. ISBN: 978-3-540-45103-7.

[10] Marina Fridman. *7 Figure Drawing Proportions to Know*. 2022. URL: https://www.thedrawingsource.com/figure-drawing-proportions.html. (Accessed: 2022-04-20).

[11] Binyam Gebrekidan Gebre, Peter Wittenburg, and Tom Heskes. "The gesturer is the speaker". In: *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*. Oct. 2013, pp. 3751–3755. DOI: 10.1109/ICASSP.2013.6638359.

[12] Chris Harris and Mike Stephens. "A Combined Corner and Edge Detector". In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151.

[13] Berthold K P Horn and Brian Schrunk. "Determining Optical Flow". In: *Artificial Intelligence*. Vol. 17. 1. Elsevier B.V, 1981, pp. 185–203. ISBN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(81)90024-2. URL: https://www.sciencedirect.com/science/article/pii/0004370281900242.

[14] Volkan Kılıç and Wenwu Wang. "Audio-Visual Speaker Tracking". In: *Motion Tracking and Gesture Recognition*. Ed. by Carlos M. Travieso-Gonzalez. Rijeka: IntechOpen, 2017. Chap. 3. DOI: 10.5772/intechopen.68146. URL: https://doi.org/10.5772/intechopen.68146.

[15] R. Lienhart and J. Maydt. "An extended set of Haar-like features for rapid object detection". In: *Proceedings. International Conference on Image Processing* 1 (2002). DOI: 10.1109/ICIP.2002.1038171.

[16] Bruce D Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)* 81 (Apr. 1981).

[17] Iain Mccowan et al. "The AMI Meeting Corpus". In: *Int'l. Conf. on Methods and Techniques in Behavioral Research* (2005), pp. 3751–3755.

[18] *Meet Android Studio*. 2022. URL: https://developer.android.com/studio/intro. (Accessed: 2022-02-02).

[19] Quang Nguyen and JongSuk Choi. "Audio-visual data fusion for tracking the direction of multiple speakers". In: *ICCAS 2010*. 2010, pp. 1626–1630. DOI: 10.1109/ICCAS.2010.5669639.

[20] *OpenCV Feature Detection*. 2022. URL: https://docs.opencv.org/3.4/dd/d1a/group_imgproc_feature.html#ga1d6bb77486c8f92d79c8793ad995d541. (Accessed: 2022-03-07).

[21] *OpenCV Optical Flow*. 2022. URL: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html. (Accessed: 2022-03-07).

[22] Azriel Rosenfeld. "Computer vision: basic principles". In: *Proceedings of the IEEE* 76.8 (1988), pp. 863–868. DOI: 10.1109/5.5961.

[23] Andreas Savaki et al. “Low Vision Assistance Using Face Detection and Tracking on Android Smartphones”. In: *IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)* (2012), pp. 1176–1179. DOI: 10.1109/MWSCAS.2012.6292235.

[24] Jianbo Shi and Carlo Tomasi. “Good Features to Track”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (1994), pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[25] *Sony Xperia 1 III*. 2021. URL: https://electronics.sony.com/mobile/smartphone/all/p/xqbc62-v. (Accessed: 2022-05-06).

[26] Pavan Turaga, Rama Chellappa, and Ashok Veeraraghavan. “Advances in Video-Based Human Activity Analysis: Challenges and Approaches”. In: *Advances in Computers*. Vol. 80. Elsevier, 2010, pp. 237–290. DOI: https://doi.org/10.1016/S0065-2458(10)80007-5. URL: https://www.sciencedirect.com/science/article/abs/pii/S0065245810800075.

[27] Ce Wang and Michael S Brandstein. “A Hybrid Real-Time Face Tracking System”. In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP ’98 (Cat. No.98CH36181)* 6 (1998), 3737–3740 vol.6.

[28] Isaac Weiss. “Digital Images”. In: *Encyclopedia.com* (2022). URL: https://www.encyclopedia.com/computing/news-wires-white-papers-and-books/digital-images.

[29] Yingen Xiong, Bing Fang, and Francis Quek. “Detection of Mouth Movements and its Applications to Cross-Modal Analysis of Planning Meetings”. In: *International Conference on Multimedia Information Networking and Security*. Vol. 1. 2009, pp. 225–229. DOI: 10.1109/MINES.2009.258. URL: https://ieeexplore-ieee-org.ludwig.lub.lu.se/document/5368362.

A

Charts from randomness test

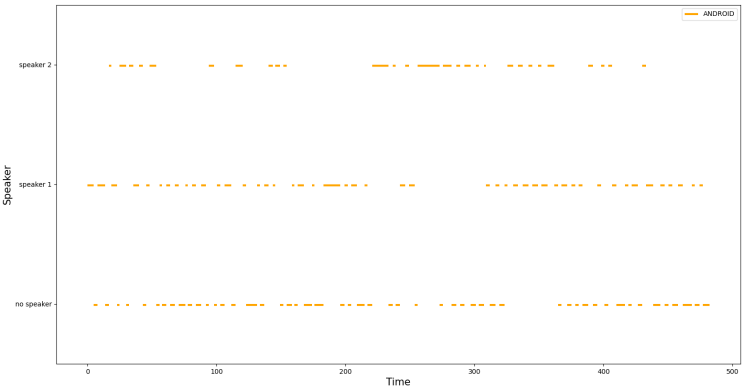

Figure A.1 Chart of test 1

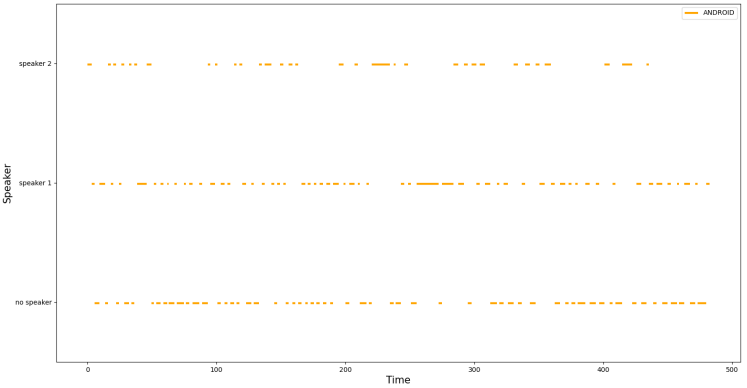

Figure A.2 Chart of test 2

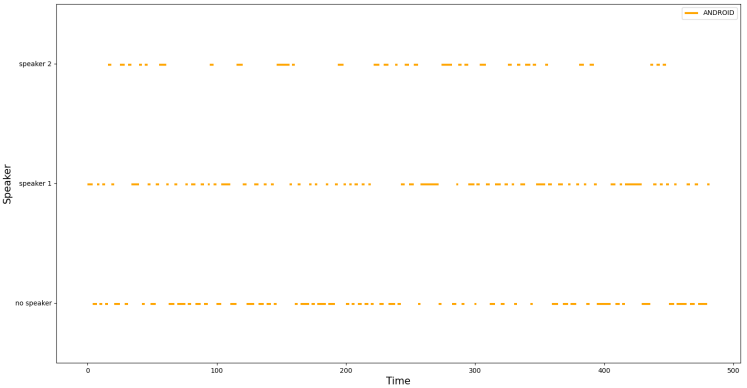

Figure A.3 Chart of test 3

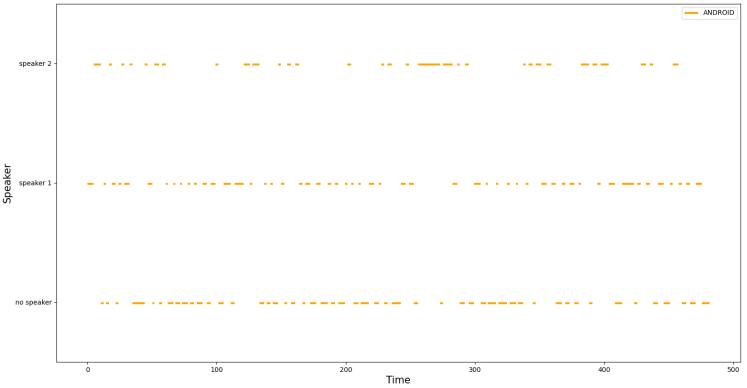

Figure A.4 Chart of test 4

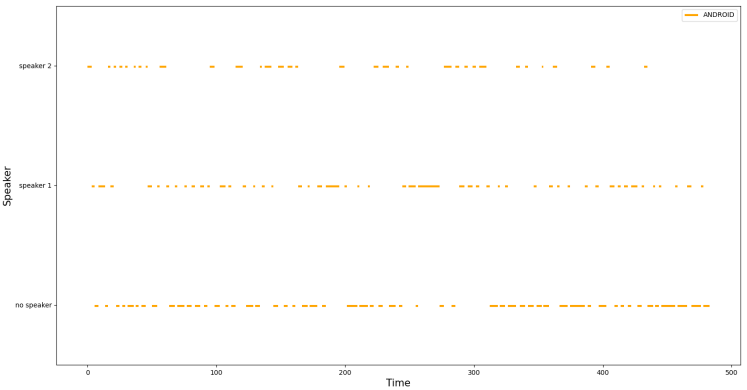

Figure A.5 Chart of test 5

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2022

Document Number
TFRT-6179

Author(s)
Emelie Skoog
Elin Freberg

Supervisor
Sebastian Raase, Sony Nordic, Sweden
Richard Pates, Dept. of Automatic Control, Lund University, Sweden
Anton Cervin, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Automatic Virtual Tracking in a Multi-Participant Scenario for a Mobile Device

Abstract

The aim of this thesis is to identify a speaker in a video stream by only using the camera of a smartphone. This creates the opportunity for a mobile application to track the speaking participant during a conference call. Conference systems that include this feature use a microphone array to localize sound and who the speaker is. Mobile phones however, are generally not equipped with a microphone array and can therefore not recognize the source of the sound. Instead, this thesis will investigate how the speaker can be determined without using audio input. One method suggests that the speaker can be identified by detecting upper-body motion. This is based on the hypothesis that people often gesture while speaking. This method was implemented as a mobile application and the accuracy was tested. The application uses the camera of a mobile device for input frames and the built-in face detector to retrieve the position of the participants. The bounds of the upper-body for each participant were estimated by using the face bounds from the face detector. The application can then detect and compare motion by calculating the optical flow in each of these bounds. The participant with the most movement will be seen as the speaker. If movement is below a certain threshold value however the application will treat this as if no one is speaking. Lastly this approach was evaluated against a transcribed data-set with videos of two participants. Some variations were also investigated with the aim to increase the accuracy. The application had an accuracy of determining the speaker about 50% of the testing time. However, some alterations were implemented that did improve the results slightly, which indicates that a bigger parameter search could increase the result further.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-70

Recipient's notes

Security classification