UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Computer Science Curriculum

Andres Traumann

Smile Detector Based on the Motion of Face Reference Points

Bachelor's Thesis (6 ECTS)

Supervisor: Gholamreza Anbarjafari, PhD

Tartu 2014

Contents

Smile Detector Based on the Motion of Face Reference Points

Abstract: Human and computer interaction is without doubt a really important part of our modern society. In order to improve it even further it is possible to develop computer systems that react to gestures or facial expressions of its user. Smiling is an expression that gives probably the most information about a person. In this thesis we describe an algorithm that understands when a person is smiling. To achieve that we first detect a face of a person using the Viola-Jones algorithm. After that several facial reference points are located and then tracked across several consequent frames using optical flow. The motion of these points is analyzed and the face is classified as smiling or not smiling.

Keywords: Smile detection, Image processing, Computer vision

Naeratuse detektor näo kontrollpunktide liikumise põhjal

Lühikokkuvõte: Inimese ja arvuti suhtlus on kahtlemata tänapäeva ühiskonna väga tähtis osa. Et seda veelgi parandada on võimalik luua süsteeme, kus arvuti reageerib inimese liigutustele või näoilmetele. Naeratamine on ilmselt näoilme, mis annab inimese kohta kõige rohkem informatsiooni. Selles lõputöös kirjeldame algoritmi, mis suudab tuvastada seda, kui inimene naeratab. Selleks leiame kõigepealt Viola-Jones'i algoritmi abil näo asukoha. Seejärel leiame vajalikele näoosadele vastavad kontrollpunktid ning jälgime nende liikumist järgmiste videokaadrite jooksul. Tuvastatud liikumise järgi otsustab algoritm, kas inimene naeratab või mitte.

Võtmesõnad: Naeratuse tuvastamine, Pilditöötlus, Tehisnägemine

1 Introduction

In human communication information is transferred using the language, the tone of the voice and also a variety of different gestures and facial expressions. All of these components play a significant role in understanding other people well.

As it gives lots of useful information about people, it would be very good to be able to automatically classify the expressions using a computer program. In this thesis we describe an algorithm that detects whether a person is smiling or not smiling.

1.1 History

The study of facial expressions has a long history. In 1872 Darwin created the universality hypothesis [3]. It states that in all parts of the world and in all the different cultures people use exactly the same facial expression. They also have the same meaning, people smile when they are happy and feel well everywhere in the world.

The work by Ekman and Friesen [4] in 1971 is considered a very important step in facial expression research. They defined six main categories for all different human facial expressions: happiness, sadness, surprise, fear, disgust, anger.

With the development of faster computers this theoretical model started to inspire research in the field of computer science. One of the first works in this field was published already in 1978 by Suwa et al [5]. It became a common topic of research somewhat later in the 1990s.

1.2 Facial Expression Recognition

Facial Expression Recognition is a challenging problem in computer vision, it attempts to classify different human facial expressions into a discrete number of expression classes. This task is complicated, as the expressions create only slight changes in a face and different expression classes can look quite similar. Moreover, the facial expressions do not have a very good definition for an algorithm and they

are therefore rather hard to detect. Issues like illumination changes and different possible head rotations complicate the task even further.

Facial expression recognition is an open problem even today. There are methods that give quite good results and work in most of the cases, but researchers are still trying to develop a method that would work accurately in all the environments and lighting conditions.

1.3 Importance

As facial expressions provide useful information about a person, recognition techniques have a high importance in artificial intelligence and computer vision. It would help to create autonomous systems that are able to interact with people in a way that is the most comfortable to them.

As facial expression reflect the emotional state of a person, it is possible to create different monitoring systems. For example, it becomes possible to constantly observe patients in a hospital, the system would detect when the emotional state of a patient is not good and would automatically notify a doctor.

Conventional human and computer interaction methods can also be improved greatly. It enables to design computer systems that react to different facial expressions. It is possible to configure different actions for the computer in case it is detected that its user makes some specific facial expression.

1.4 Smile Detector

In this thesis a smile detector is described. Detecting a smile is a first step towards a general facial expression recognition. Smile detector is a computer program that analyzes video input received from a webcam and tries to detect a smiling person. To detect a smile we first need to locate a face of a person. If a face is found, then its location is used to calculate several reference points that are being tracked during the consequent frames. The motion of these points is analyzed and a decision is made about whether the face is smiling or not.

The working principles of our smile detector are the following: The face of a

person is located using the Viola-Jones algorithm [6]. Then we use the statistical information about the location of eyes, mouth, nose and chin to calculate the approximate locations for the reference points. The locations are refined even further with the help of Harris corner detector [14]. The reference points are tracked using the optical flow [9] method over several frames to extract motion information. The motion is analyzed and used to classify the facial expression as smiling or not smiling.

1.5 Outline of the Thesis

In section 2 the smiling expression is discussed more thoroughly. In section 3 there is a short review of different already existing detection methods. In the following sections 4-8 we introduce important methods that were used in this work. Section 4 describes the Viola-Jones object detection algorithm, section 5 introduces the concept of discrete derivatives in image processing that is needed in the following sections. Section 6 explains the optical flow method for tracking reference points and section 7 specifies how to find these points in an image, section 8 describes a method for detecting corners to improve reference point locations. Section 9 describes the actual smile detection algorithm and section 10 gives a brief overview of the program implementation details. Section 11 presents the results, section 12 introduces some applications and section 13 concludes the thesis.

2 Smiling Expression

Smiling is one of the basic facial expressions everywhere in the world. People smile to express their happiness, when they are in a good mood or when they are amused or when they are satisfied with their current situation. Smiling also has approximately the same meaning in all cultures.

2.1 Human Understanding

All people can easily understand when a person is smiling, it is easy for everyone who is at least three years old and does not have any specific medical conditions that would prevent them from seeing or thinking properly. It is intuitive for everyone and does not need any consideration or conscious analysis to tell if a person is smiling or not. It is possible due to a vast amount of previous experience of communicating with people, understanding their behavior and their reactions to different situations. People are unconsciously learning these things starting from a very young age, so that later they can do it without any conscious consideration process.

2.2 Algorithmic Approach

Detecting a smile computationally, however, is a really challenging problem. The reason is that none of the existing computer programs has a good enough understanding about the images or video stream it receives as input. It is not practically possible to write such program either. It is possible to process image or video input in a computer, but in most cases, some specific algorithm is used for that, there is rarely any intelligent understanding.

Therefore a detection algorithm can not usually rely on the background knowledge and visual understanding that humans use. Some more specific algorithm is needed to do it. In order to develop such algorithm, we need to define what we understand as a smile. The problem is that it does not have a good accurate definition. People smile in many different ways, some people have a really wide smile, some people only smirk, it is also possible to smile by using only one side

of the mouth. Ideally we would like the algorithm to detect all of these types of smiles.

2.3 Smiling Criteria

One of the common properties in all types of smiles is that the distance between mouth corners increases noticeably. It is also common that teeth become visible during a smile and it would give us a simple criterion to check, but it might happen during other expressions as well. The lips usually form a bit different shape compared to a neutral face and it could be used somehow. It turns out that in most cases the distances between the eyes also increases a bit, because most people narrow their eyes slightly while smiling. Surprisingly even the distance between nostrils increases a little bit during a smiling expression. We have now identified a few possible criteria that could be used to create an algorithm.

3 Overview of Different Methods

A lot of research has been done on the field of facial expression recognition. Smile detection is somewhat less studied, but the methods of facial expression recognition generally extend to smile detection as well. There are two kinds of methods: the ones that work on a single image and the ones that work on a video using the information obtained from movement.

3.1 Methods for Single Image

An image contains a huge amount of information about a face. The challenge is to select the most meaningful information that would enable us to process and classify it easily based on our specific needs. This makes it a really challenging problem.

Facial expression detection techniques often build on methods that were originally created to solve the general face recognition problem. One of those methods is the elastic graphs method [22]. It was proposed to recognize faces and classify them based on gender. The idea of this method is to use a small database of previously constructed graphs for matching the real images. Classifications of faces are made based on how well they match to these predefined graphs.

Another important theoretical idea is to use statistical methods to reduce the high dimensional information into much lower dimensions. This can be done quite effectively, as a considerable amount of data is usually irrelevant or with high correlation. It enables to express most of the data in terms of a small amount of chosen meaningful information. As there is less information, it is easier to analyze it too. Well known algorithms are the Eigenface [24] and the Fisherface [25] algorithms, which use the same concepts, but a different method for dimensionality reduction.

Lyons et al. use elastic graphs combined with a grid of features computed by the Gabor wavelet method to classify images based on facial expressions [26]. The feature grid is processed in a way that resembles the Eigenface and Fisherface algorithms. A dimensionality reduction is performed and the resulting information

is used to classify the images based on facial expressions.

A somewhat similar approach is used by Shinohara and Otsu in [23]. They use higher order local auto-correlation features instead of the Gabor wavelet features. Dimensionality is reduced by using Fisher weight maps, final classification is made using Fisher discriminant analysis.

Another method for facial expression classification uses discriminative scale invariant feature transform D-SIFT [27]. After constructing feature vectors with D-SIFT, weighted majority voting is used for classification.

Facial expression classification can be improved when 3D information is available. Soyel and Demirel proposed a method [28] that calculates 3D distance vectors and applies neural networks for the final classification.

Another much more simple and less theoretical method works using cross-correlation template matching. When the result of the cross-correlation of the mouth area is higher than some fixed threshold, then a smile detection is claimed.

It is also possible to detect the corners near the lips and then tries to draw a Bézier curve through them [29]. Based on how well the curve resembles the expected result, a decision is made whether a person on the image is smiling or not.

3.2 Methods for Video

Methods for video can use the additional information provided by movement throughout the video frame sequence. The methods that work on a video usually detect the location of a face and then find several reference points that will be used for tracking during the consequent frames. The information about the movement is used to make a decision about the facial expression. The algorithm that is described in the thesis also belongs to this category and it is somewhat inspired by the work of Huang and Fuh [18]. It uses the Viola-Jones [6] algorithm to detect the location of a face and then calculates reference point positions using statistics about their average locations. The points are tracked by optical flow [9] and the obtained information is used to decide whether a person is smiling or not. Similar detection techniques are described in [19], [20] and [21].

4 Viola-Jones Object Detection Algorithm

For face detection, the Viola-Jones object detection algorithm [6] is used. It is an algorithm that enables rapid processing and high detection rates and that makes it a very good choice for our purposes. The image is first converted to a specific representation called the integral image. Then the algorithm uses a set of simple rectangle features for testing all the sub-windows of the image that have a reasonable size. In each sub-window a cascade of classifiers is used to make a decision on whether the sub-window contains an object of interest or not.

4.1 Features

The Viola-Jones detector uses simple features instead of pixel intensity values. Examples of the used features are shown in Figure 1. All the features consist of two types of rectangles: white and gray. Viola and Jones originally proposed three

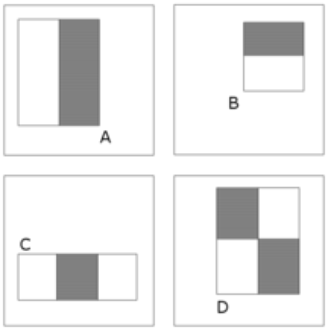

Figure 1: Feature types originally proposed by Viola and Jones in [6].

kinds of different features that are all displayed in Figure 1. Our implementation actually uses the extended set of features proposed by Lienhart et al. in [7]. Most notable addition of the extended set is the use of rectangular features that are rotated by 45 degrees. To calculate a feature it is placed on the current search window. The value of a feature is calculated by summing up the pixel intensity values in the white rectangles and subtracting it from the sum of the pixel intensity values in the gray rectangles. These features are used to achieve fast computation

that is possible with the use of a special image representation called the Integral Image.

4.2 Integral Image

To make the calculation of features easier, the image is first converted to the integral image. Let $I(x, y)$ be an image. Then the integral image $I'(x, y)$ is defined by the equation

$$I'(x, y) = \sum_{x_0 \leq x, y_0 \leq y} I(x_0, y_0). \tag{1}$$

It means that the value at a coordinate (x, y) is the sum of all the pixel values above and to the left of the given coordinate in the original image.

By using this format, it is easy to calculate all the previously discussed features in only a few summations and subtractions. Consider Figures 2 for example. The left-side array of Figure 2 represents an arbitrary image and the array to the right is its integral image representation.

8	0	9	1	1
6	3	0	9	1
0	5	5	4	4
4	2	0	6	1
9	6	3	6	0

Image $I(x, y)$

8	8	17	18	19
14	17	26	36	38
14	22	36	50	56
18	28	42	64	69
27	43	60	88	93

Integral Image $I'(x, y)$

Figure 2: Example about integral image representation

If we want to calculate the sum of pixel values σ in a rectangle with (zero-based)

coordinates $(1,1),\ (3,1),\ (2,1),\ (3,2)$ for example, then

$$\sigma = \sum_{\substack{1\leq x\leq 3,\\ 1\leq y\leq 2}} I(x,y) = I'(0,0) + I'(3,2) - I'(3,0) - I'(0,2) = 26. \tag{2}$$

4.3 Cascade Classifier

For object detection a cascade of classifiers are used. Each test evaluates one of the previously discussed rectangle features and compares the value with a threshold to decide whether the test was passed or not. The cascade of classifiers means that the second classifier is only used when the first one gives a positive decision, the third one is used only when the second one gives a positive result etc. It means that a negative decision by any of the classifiers results in immediately stopping the evaluation of the current image sub-window. That way all the unpromising sub-windows could be discarded as early as possible, whereas the promising areas will be calculated more precisely. The first classifiers are designed in a way that lets all the correct sub-windows pass along with a large number of false positive areas. Each of the following classifier gradually lowers the false positive rate while trying to keep the detection rate of correct areas as high as possible. It means that the object is detected only when the investigated image sub-window passes all the tests in the cascade.

4.4 AdaBoost

To create a previously discussed cascade of classifiers a machine learning algorithm called AdaBoost [8] is used. AdaBoost is an algorithm that uses the output of some other learning algorithm, often called a weak learner. In each AdaBoost cycle the weak learner generates some form of classifier. The idea is that several weak classifiers are combined in a way that they enable much more accurate classification compared to running each of them separately.

For face detection the AdaBoost algorithm needs to identify the rectangular features that enable the best classification. The weak learner is designed, so that it always chooses exactly one of all the possible rectangular features, so it enables

to classify the positive and negative examples most accurately. In each round of the AdaBoost the following classifier function is generated by the weak learner:

$$h(x) = \begin{cases} 1 & \text{if } pf(x) < p\,\theta \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where f is a feature that was chosen, θ is the threshold for optimal classification, x is the search window and p is the polarity that indicates the direction of the inequality. These classifiers are combined and eventually used to create a cascade classifier that was introduced in the last subsection.

4.5 Scanning the Image

The whole image is scanned by moving a sub-window across the image and applying the cascade of classifiers in every sub-window. The initial size of the sub-window is chosen to be reasonable and the exact size was fixed by a series of experiments. After the whole image is scanned with this sub-window, then the dimensions of the sub-window are increased by a scale factor constant that was also chosen by experiments. The scaling is important, because the size of the face in an image is not known beforehand. The whole process of scanning and scaling the window is repeated until some fixed maximum window size is reached. When this happens then the detection algorithm has completed its work.

5 Derivative of a Discrete Function

In calculus the derivative of a function $f(x)$ with respect to variable x is defined as the function f' whose value at x is

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}, \tag{4}$$

provided the limit exists. For the limit to exist when $h \to 0$ the function f needs to be defined at every point of the open interval $(x - \epsilon, x + \epsilon)$, except possibly at the point x itself [2].

Digital images are defined as a discrete functions, however. This means that they are not defined at every point of an open interval. This also means that the limits of a discrete function are not defined. As the result we can not talk about image derivatives using the definition from calculus. In discrete cases, another definition is often used to obtain approximation to continuous derivatives. The derivative of a continuous function $f(x)$ is a function $f'(x)$ that describes how the values of f change near x. An intuitive approximation for the discrete function is obtained by defining:

$$f'(x) = f(x+1) - f(x) \tag{5}$$

where f is a discrete function $f : \mathbb{Z} \to \mathbb{Z}$. This definition is well-known and it is often used in image processing [1]. The definition can be generalized to higher dimensions as well. The derivative of a continuous function $f(x_1, ..., x_n)$ is defined as

$$\frac{\partial f}{\partial x_i}(x_1, ..., x_n) = \lim_{h \to 0} \frac{f(x_1, ..., x_i + h, ..., x_n) - f(x_1, ..., x_i, ..., x_n)}{h}, \tag{6}$$

provided the limit exists [2]. We will define the derivative of a discrete function of multiple variables as

$$\frac{\partial f}{\partial x_i}(x_1, ..., x_n) = f(x_1, ..., x_i + 1, ..., x_n) - f(x_1, ..., x_i, ..., x_n). \tag{7}$$

where f is a discrete function $f : \mathbb{Z}^n \to \mathbb{Z}$.

6 Optical Flow

Optical flow [9] can be defined as apparent motion of brightness patterns in an image. Optical flow is caused by the movement of the observer (e.g. camera) or the movement of the objects in the scene that is being observed. The concept of optical flow enables us to estimate the movement of objects during a sequence of images.

6.1 Optical Flow Equation

We will derive an equation that represents the previously explained concept of optical flow in an image. Let $I(x, y, t)$ denote the intensity value of an image at spatial coordinates (x, y) at time t. We will assume that pixel intensities are constant in time. It means that

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{8}$$

Assuming that the movement in space and time is relatively small, we can approximate the right-hand part of the equation using the Taylor series [2].

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \alpha \tag{9}$$

where α contains the higher order terms. When we substitute into (8) and subtract $I(x, y, t)$ from both sides of the equation, we get

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \alpha = 0 \tag{10}$$

After dividing the equation with Δt we have

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} + \beta(\alpha) = 0 \tag{11}$$

Now, we know that x and y change only when t changes, it means that if $\Delta t \to 0$, then $\Delta x \to 0$ and $\Delta y \to 0$. Therefore, when $\Delta t \to 0$, then $\alpha \to 0$ and $\beta(\alpha) \to 0$. Let's now denote $u = \frac{\Delta x}{\Delta t}$ and $v = \frac{\Delta y}{\Delta t}$, we obtain

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{12}$$

We have now derived the optical flow equation. The equation has two unknown variables: u and v and it can't be solved in that form without any additional constraints.

6.2 Lucas-Kanade Method

To actually solve the optical flow equation (12), there are many different methods. In my application, it is needed to calculate the optical flow vectors for a few chosen points and not the whole vector field. That is the reason why I decided to use the Lucas-Kanade method [10], [11]. The Lucas-Kanade method introduces a new assumption: it assumes that the motion is locally the same. It means that we can take a 3 x 3 pixel neighborhood and assume that they all have the same movement vectors. As the result we get a system of 9 equations with 2 unknowns, so it is possible to solve it now. We have the following over-determined equation system:

$$\begin{cases} \frac{\partial I(x_1,y_1,t)}{\partial x}u + \frac{\partial I(x_1,y_1,t)}{\partial y}v = -\frac{\partial I(x_1,y_1,t)}{\partial t} \\ \frac{\partial I(x_2,y_2,t)}{\partial x}u + \frac{\partial I(x_2,y_2,t)}{\partial y}v = -\frac{\partial I(x_2,y_2,t)}{\partial t} \\ \quad \vdots \\ \frac{\partial I(x_9,y_9,t)}{\partial x}u + \frac{\partial I(x_9,y_9,t)}{\partial y}v = -\frac{\partial I(x_9,y_9,t)}{\partial t} \end{cases} \tag{13}$$

where the optical flow equations are evaluated at points $x_1, ..., x_9$, $y_1, ..., y_9$ and at the current time t.

The Lucas-Kanade method was initially introduced as a method for image registration. To achieve the image registration, is necessary to find the displacement vector of an image patch from one image to another. The method uses two functions: $F(x)$ and $G(x) = F(x+t)$, where $x = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}$ and $t = \begin{pmatrix} t_1 & t_2 & \dots & t_n \end{pmatrix}$ is the disparity vector, the objective is to find that vector when the functions $F(x)$ and $G(x)$ are known. This is done by minimizing the error with between these two functions. The goal is to find the displacement vector that minimizes the least squares error

$$E = \sum_{x \in X} [F(x+h) - G(x)]^2 \tag{14}$$

where X is the set of all the points that are used for finding the disparity vector. The following Taylor series based approximation is used:

$$F(x+h) \approx F(x) + h\frac{\partial F(x)}{\partial x} \tag{15}$$

where $\frac{\partial F}{\partial x} = \begin{pmatrix} \frac{\partial F}{\partial x_1} & \dots & \frac{\partial F}{\partial x_n} \end{pmatrix}^T$. We are trying to obtain such h that the square error would be minimal.

$$\begin{aligned} 0 = &\frac{\partial E}{\partial h} \approx \frac{\partial}{\partial h} \sum_{x \in X} \left[F(x) + h\frac{\partial F(x)}{\partial x} - G(x) \right]^2 = \\ &\sum_{x \in X} 2\frac{\partial F(x)}{\partial x} \left[F(x) + h\frac{\partial F(x)}{\partial x} - G(x) \right] \end{aligned} \tag{16}$$

We can now use the previous theory to solve our system of partial differential equations. The images have two dimensions, so we are looking for $h = \begin{pmatrix} u & v \end{pmatrix}$. As two functions F and G we have two images $I_1(x,y) = I(x,y,t)$ at times and $I_2(x,y) = I(x,y,t+\Delta t)$. It means that

$$I_2(x,y) - I_1(x,y) = \frac{\partial I}{\partial t}(x,y,t) \tag{17}$$

according to the way the discrete function partial derivatives are defined.

Finally we obtain the following equation system:

$$\begin{cases} u\sum_{i=1}^{n} \left(\frac{\partial I(x_i,y_i,t)}{\partial x} \right)^2 + v\sum_{i=1}^{n} \frac{\partial I(x_i,y_i,t)}{\partial x} \frac{\partial I(x_i,y_i,t)}{\partial y} = -\sum_{i=1}^{n} \frac{\partial I(x_i,y_i,t)}{\partial x} \frac{\partial I(x_i,y_i,t)}{\partial t} \\ u\sum_{i=1}^{n} \frac{\partial I(x_i,y_i,t)}{\partial x} \frac{\partial I(x_i,y_i,t)}{\partial y} + v\sum_{i=1}^{n} \left(\frac{\partial I(x_i,y_i,t)}{\partial y} \right)^2 = -\sum_{i=1}^{n} \frac{\partial I(x_i,y_i,t)}{\partial y} \frac{\partial I(x_i,y_i,t)}{\partial t} \end{cases} \tag{18}$$

It is a linear system with two equations and two variables: u and v, so we can easily solve it after finding the partial derivatives.

6.3 Pyramid Representation

The major drawback of the Lucas-Kanade method is that it can only used when the movement is small. All the calculations are made using a small image window,

so obviously it can not track larger movements. To make it possible, it is necessary to use pyramid representation [12]. The basic idea is to use the Lucas-Kanade method several times while reducing the scale of the images. When applying the Lucas-Kanade method on a scaled down versions of images, the small image window, that is used for calculation, now covers a larger area of the image and makes it possible to estimate a larger movement.

Let L_m be the number of pyramid levels. Also, let I^0 denote the lowest level of the pyramid, the image itself in its original measurements. I^{L_m-1} will denote the highest level of the pyramid. The image of level L can be calculated from level $L-1$ image in the following way:

$$I^L(x,y) = \frac{1}{4}\Sigma_1 + \frac{1}{8}\Sigma_2 + \frac{1}{16}\Sigma_3, \tag{19}$$

where Σ_1, Σ_2 and Σ_3 denote

$$\begin{aligned}
\Sigma_1 =& I^{L-1}(2x, 2y) \\
\Sigma_2 =& I^{L-1}(2x-1, 2y) + I^{L-1}(2x+1, 2y) + \\
& I^{L-1}(2x, 2y-1) + I^{L-1}(2x, 2y+1) \\
\Sigma_3 =& I^{L-1}(2x-1, 2y-1) + I^{L-1}(2x+1, 2y+1) + \\
& I^{L-1}(2x-1, 2y+1) + I^{L-1}(2x+1, 2y-1).
\end{aligned} \tag{20}$$

Let's now explain how the different levels of the pyramid work together to determine the final motion vectors of our point of interest $p = (x, y)$.

First the motion vector is calculated at the highest level of the pyramid image I^{L_m-1}. In the next level I^{L_m-2} the motion vector from the upper level is used as an initial guess. It means that the image patch that is used for comparison to find the displacement vector is determined by the coordinates that are received from the upper level. That way it is possible to use the guess from the higher levels of the pyramid and then fine-tune it when the resolution of the image is higher.

Let $g^L = \begin{pmatrix} g_x^L & g_y^L \end{pmatrix}^T$ denote the initial guess for level L image I^L that is obtained by applying the Lucas-Kanade method on higher levels $L+1$, $L+2$ and so on. If there are no higher levels, then the initial guess will be the zero vector, i.e. $g^{L_m-1} = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$. To improve the initial guess, the following sum of square errors needs to be minimized:

$$E = \sum_{i=1}^{n} \left(I_1^L(x_i, y_i) - I_2^L(x_i + g_x^L + h_x,\ y_i + g_y^L + h_y) \right)^2 \tag{21}$$

where the sum is calculated over n pixels in the image window. After finding the new disparity vector $h = \begin{pmatrix} h_x & h_y \end{pmatrix}^T$, then the new vector is calculated by

$$g^{L-1} = 2(g^L + h) \tag{22}$$

and passed on to the lower levels if there are any. If there are no more levels then $g^L + h$ will be the final result.

7 Calculating Face Reference Points

To use the optical flow tracking method effectively for our purpose, we need to select reference points that would give us useful information about a person's facial expressions. It is intuitively clear that the points near the eyes, mouth and nose would provide more useful information compared to some other point locations.

The smile detection algorithm that is being described uses the following reference points to make a decision: mouth corners, inner and outer eye corners, nostrils and the tip of the chin - 9 points in total. We believe these points would give the algorithm enough information to decide whether the person is smiling or not.

7.1 Detection by Using Statistics

Accurate detection of the required points, however, is not a simple task. As our algorithm needs to work in real-time speed, we need a computationally efficient solution. Fortunately, the locations of eyes, mouth and nose do not change very much, they are always approximately in the same place for all people. It means that we can make an accurate guess using statistical information about the locations of these points.

We used the BioID image database [13] to extract information about the average locations of the face reference points. The BioID database consists of over 1500 images that show a face of a person while doing different facial expressions. Along with these images they provide data about the locations of 20 different reference points of these faces (see Figure 3). We detected these faces with the Viola-Jones algorithm and calculated the provided locations in respect to the face rectangle location given by the Viola-Jones algorithm. As a result, we can now give quite good estimations for the reference points that are needed by smile detection algorithm by simply using the average positions gathered from the BioID data.

Of course, this kind of approach depends heavily on the accuracy of the Viola-Jones detector. In practice, however, it seems to work quite well. The reason is that the Viola-Jones algorithm gives quite stable output, as the result we obtain

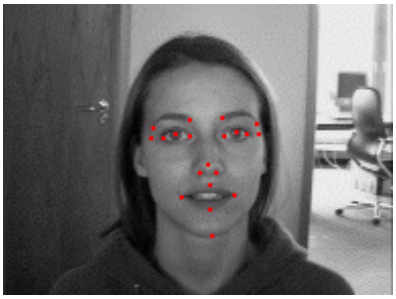

Figure 3: A subset of these marked reference points are used. Image courtesy by the BioID database [13].

quite accurate locations of our reference points.

7.2 Improvement by Corner Detection

This approach can be improved even further. The optical flow algorithm works better for areas that contain edges or corners. It means that we can make the optical flow tracking more precise, when we detect the strongest corners near the estimated locations of our reference points.

We tested the smile detector with and without the use of the corner detector and got different results depending on the lighting conditions of the working environment. In stable and good lighting conditions the corner detector generally improves the detection process by locating the reference points more precisely. In conditions where part of the face is in shadow, however, the corner detector often detects false points like the edges of the shadows and the overall detection accuracy is much worse.

8 Corner Detection

In order to refine the interest point locations, we can use a corner detection algorithm. We know the approximate locations of the points, so we can detect the strongest corner near the previously computed reference point position. The reason for doing so is that we the optical flow tracking is much more accurate near edges and corners.

We used the Harris corner detection algorithm [14] in our smile detector. We also had several good alternatives like the Susan detector [17] or the Shi-Thomasi detector [16] that would both have been suitable for our application. The Harris detector performed as well as its alternatives and while we had no clear preference, some choice had to be made.

The corner detector proposed by Harris and Stephens is actually an improvement over the detector proposed by Moravec [15] (in his work he referred to it as the interest operator). That is why we start our discussion by explaining the working principles of the Moravec detector.

8.1 Moravec Corner Detector

The Moravec Detector scans the image using a small fixed size search window. In every window location it shifts the search window and then calculates the sum of squared differences of pixel intensities in the initial and in the shifted search window. When the search window contains a corner, then shifting it in any direction would make the pixel intensity difference quite large. In case it is an edge, the difference would be large only in the direction perpendicular to the edge, the change would be small in the direction along the edge. If the search window does not contain an edge or a corner, then a shift in either direction would give a small difference.

Let $I(x, y)$ denote the intensity value of image I at position (x, y) and let $E(u, v)$ denote the sum of squared intensity value differences when the window is shifted

by a vector (u, v).

$$E(u,v) = \sum_{(x,y)\in W} \Big[I(x+u, y+v) - I(x,y) \Big]^2, \tag{23}$$

where W is the set of all the pixel coordinates of the current search window. The shift by (u, v) is done by one pixel in every possible direction vertically, horizontally and diagonally. After calculating all the shift differences, only the minimum of them is used, because this way the algorithm avoids detecting edge points. When the minimum is larger than some fixed threshold, the algorithm has found a corner candidate. There are usually several candidates near a corner, only one of them is chosen by selecting the corner with the highest minimum shift value.

8.2 Harris Corner Detector

The Harris detector improves on the Moravec detector by removing many of its drawbacks.

Moravic detector has only a discrete number of possible shifts, all shifts are multiples of 45 degrees. We can obtain arbitrarily many shifts with the following method: We can use the Taylor series expansion [2] to approximate

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v. \tag{24}$$

When substituting it into (23), we obtain

$$E(u,v) = \sum_{(x,y)\in W} \left[\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \right]^2. \tag{25}$$

This clearly enables us to have infinitely many different shifts. To make this work even better, Harris and Stephens point out that the rectangular search window W can be replaced with a circular window, possibly with a Gaussian weighting function.

The Harris detector also uses a different scheme to make a final decision about

the existence of a corner. Instead of using a minimum of shift differences, the Harris detector relies on eigenvalues. First of all, we can write (25) using matrices:

$$E(u,v) = \begin{pmatrix} u & v \end{pmatrix} M \begin{pmatrix} u \\ v \end{pmatrix}, \tag{26}$$

where

$$M = \sum_{(x,y)\in W} \begin{pmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{pmatrix}. \tag{27}$$

For a search window W to contain a corner, there needs to be a large change in all the possible shifting directions. The eigenvalues λ_1 and λ_2 of matrix M can be used to check this. After analyzing the dependance between M and the changes in intensity values, Harris and Stephens concluded that a large change in any direction means that both eigenvalues of M are large. In case of an edge, one of the eigenvalues is much greater than the other and in a nearly constant image patch, both eigenvalues are relatively small.

As calculating eigenvalues is computationally a quite expensive operation, the Harris detector works using the determinant and trace (sum of elements in the main diagonal) of matrix M. It is possible to use them, as it holds that $\mathrm{Det}(M) = \lambda_1\lambda_2$ and $Trace(M) = \lambda_1 + \lambda_2$. Finally, the following criterion is used to evaluate the probability that the search window contains a corner:

$$R = \mathrm{Det}(M) - k\ (Trace(M))^2, \tag{28}$$

where k is some fixed constant. R is compared to a threshold to decide if the search window contains a corner or not.

9 Smile Detection Algorithm

The smile detection algorithm uses the motion of the reference points as input. The points are tracked across several frames to gather enough information about their movement. The motion is analyzed and finally a decision it is made, whether the motion corresponds to a smiling facial expression or not.

The main features that are used for detection are the changes of the distance between the mouth corners, nostrils and eye corners. Experiments clearly indicate that all these distances increase when a person is smiling. Using this knowledge it is possible to recognize most of the smiling expressions.

9.1 Sequence of Positions

From here on, we use the word *sequence* to refer to an ordered list of reference point positions. Each element of the sequence is a set that consists of the coordinates of 9 reference points. Each element describes the positions of the reference points at different moments in time. In short, a sequence of n elements contains full information about the motion of the reference points over n frames. As the algorithm often processes several sequences at the same time, the previous definition was necessary to describe the algorithm more clearly.

9.2 Workflow of the Algorithm

After a face has been detected by the Viola-Jones algorithm, the positions of all the reference points are calculated and a new sequence is started based on the coordinates of these points. The newly created sequence is added to the list of all the current sequences. After that all the previous sequences in the list are also updated. It means that the most recent reference point coordinates of a sequence are used to calculate a new element by the Lucas-Kanade optical flow calculation. It means that the number of elements of all the previous sequences is increased by one.

After all the sequences in the list are updated, they are all analyzed to interpret the motion of their reference points. The analysis procedure has three possible

results:

- Keep the sequence;
- Discard the sequence;
- Smile was detected.

The sequence can be either kept in the list for further processing or discarded as unpromising. If the result is a smile detection, then the sequence is obviously removed from the list as well. The sequences are mostly removed, because the motion that they represent is not interesting or relevant for our algorithm. It is also possible that the sequence analysis found them to be too noisy or the measurements were abnormal and considered to be corrupted.

Sequences are also automatically discarded after they reach some maximum limit. Our algorithm uses 20 as the maximum limit, because it corresponded to roughly one second in time in the computer that was used for testing. We estimate that most of the smiling expressions last a bit less than one second, so it should be a quite good upper limit. Overall it means that the algorithm never has to process more than 20 sequences at the same time, so the computational resources are not spent more than needed.

9.3 Motion Analysis of a Sequence

9.3.1 Sanity Checks of the Measurements

Every sequence analysis starts with a few sanity checks. The measurements might contain noise and the optical flow tracking might also cause some inaccuracies, then the errors might accumulate over several frames. For this reason the newest element of a sequence is always checked using some rough estimations for the distances between the reference points. We know the approximate distances between the reference points, if some of our reference points have positions that do not fit into the overall model, then we will assume that the measurements are corrupted and the sequence will be discarded immediately.

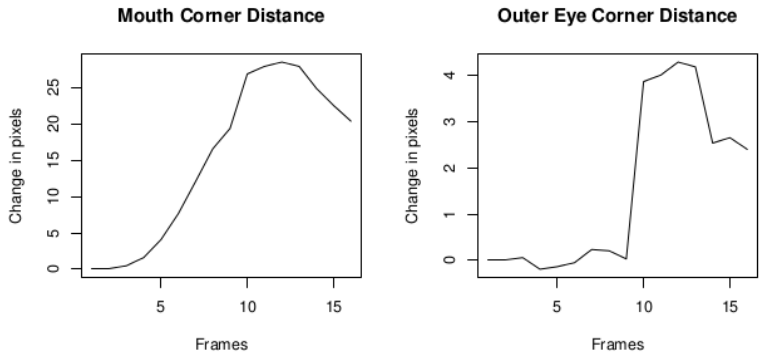

Figure 4: Example of the distance changes of mouth corners and outer eye corners in a sequence where a smile was detected.

9.3.2 Checks of Superfluous Face Movement

The second step of the analysis checks if the face has moved too much during all the previous frames. Too much movement sometimes confuses the detection mechanism of the algorithm and therefore it is better to discard such sequences to avoid false positive detections. Additionally, this analysis step also discards sequences where the person is talking in front of the camera. The movement is checked by summing up the position changes between the current and the initial reference point positions of the four eye reference points. The movement of the chin reference point is also added to the sum to avoid analyzing the sequences with too much mouth movement any further.

9.3.3 Main Detection Conditions

The detection of a smile relies on comparing the distance changes between the mouth corners, nostrils and inner and outer eye corners. When the sequence has exactly two elements, then the movement of the mouth corners is checked. If they are moving apart horizontally, then the sequence is marked as promising and kept for further tracking during the upcoming frames. All the sequences that do not have this property are automatically discarded as unpromising.

During the next frames the distance between the mouth corners is checked to detect the moment when it reaches its peak i.e. maximum value. When it hap-

pens, then the total changes of the mouth corner, nostril and eye corner distances are calculated by comparing the initial and the current measurements. All these distance changes are compared to some experimentally fixed thresholds. If the distance changes are larger than the thresholds, then the sequence is considered to have filled the conditions for a sufficient peak. Figure 4 displays the distance changes of a sequence that has filled these conditions.

9.3.4 Final Conditions

The final decision about the sequences that have a sufficient peak is made during the next n frames, we used $n = 4$. During these frames it is checked whether all the reference points indicate a movement back towards their initial positions. This stage is also useful to discard sequences that are caused by a person talking in front of the camera and accidentally making some expression that looks similar to smiling. The rationale is that usually a talking person does not stop talking exactly after accidentally making such an expression. It is probable that they will say something more immediately and the motion will not be detected as a smile. If the sequence that has shown a sufficient peak and is not discarded during these n final frames, then the algorithm claims to have detected a smile.

10 Implementation

The algorithm is written in C++ and implemented to work on a Linux system. More specifically, the algorithm was developed on an Arch Linux [31] system, nevertheless, it should work on other distributions as well.

The camera input is read by Video4Linux2 API [32]. It should be supported in most of the Linux distributions, since the API is included in the Linux kernel starting from version 2.5. Our implementation uses the API directly without using any third party libraries, as it enables fast frame acquisition that way.

For image processing the OpenCV computer vision library [33] was used. It provided implementations for the Viola-Jones algorithm, Optical flow calculation and corner detection.

10.1 GUI Tool

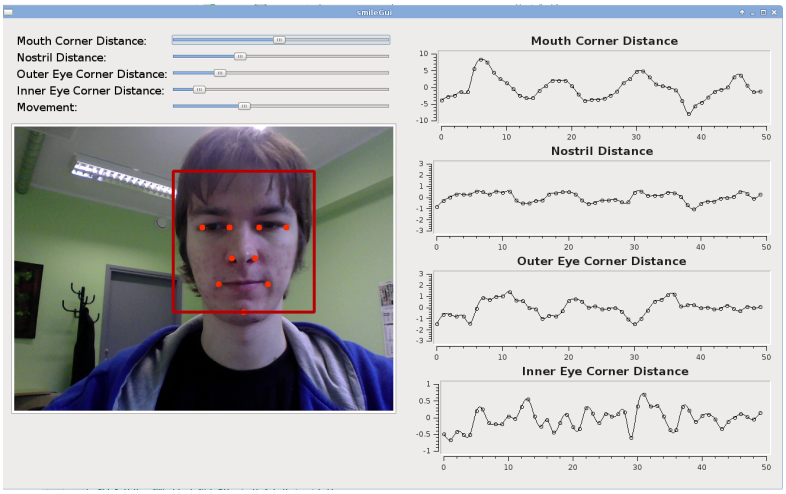

Figure 5: We created a special Gui that enables us to test the algorithm, change the detection thresholds and also shows a real-time plot of the changes in reference point distances.

The algorithm does not show any visual output except logging information and messages written to output stream. To develop the algorithm more intuitively, we created a graphical user interface (see Figures 5 and 6). It enables to test the algorithm with different thresholds that can be adjusted with sliders and it also

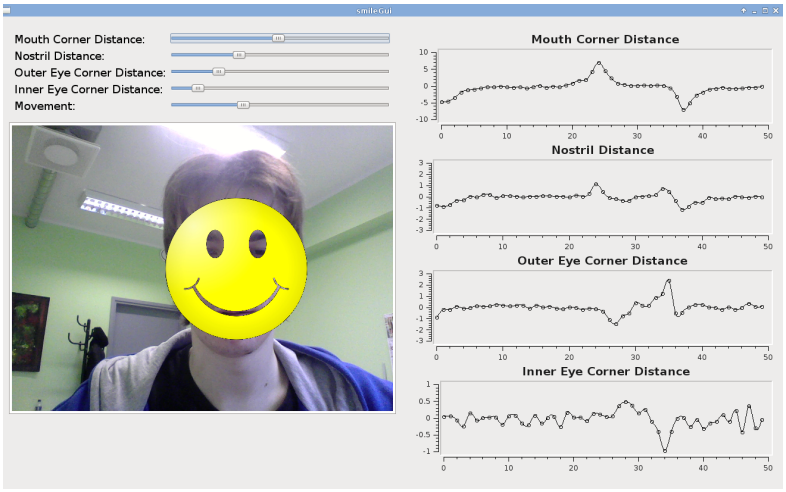

Figure 6: The algorithm has detected a smile and a big smiley is displayed.

shows real-time plots of the important reference point distance changes that are used by the algorithm.

The GUI tool is also written in C++. The graphical interface uses the Qt library [34] and the plotting is done using the Qwt library [35].

11 Results

11.1 Performance of the Detector

The algorithm works reasonably well in most environment conditions. Of course, in different lighting conditions it might be necessary to change the thresholds a little, that way the algorithm works more accurately.

The Viola-Jones algorithm locates a face really well, it is probably the most stable part of the algorithm. The reference point calculation assumes that the head is not rotated much, otherwise it will not be accurate. When using a corner detector, then the lighting must be quite good, as shadows might cause the detection of corners that do not correspond to facial reference points. The optical flow calculation works quite well, but it depends on the previously calculated reference points. If their locations are noisy, then the whole smile detection algorithm can not work very precisely. The algorithm that analysis the motion of reference points must consider all the noise that has accumulated over the last steps. In most cases it manages to detect when the measurements are too noisy or unrealistic. Basically the main weakness of the algorithm is caused by the detection of the reference points.

11.2 Detection Rate

The correct detection rate depends on the choice of the thresholds. It is possible to choose thresholds in a way that nearly all of the smiles are detected, but this also results in a relatively high misdetection rate. In that case many face movements like talking or making a grimace cause a misdetection. It is also possible to set so strict thresholds that the misdetection rate goes to almost zero, the drawback, however, is that the detection rate of the correct smiling expressions also goes quite low.

A good compromise threshold configuration should be chosen that detects most of the smiles with as few misdetections as possible. Of course, an optimal configuration depends on an application. In some cases it is not a big problem to have

some misdetections. In other cases a low misdetection rate with a bit lower correct detection rate is preferred.

11.3 Speed of the Detector

The smile detector was developed and tested on a modern computer with 8-core Intel i7 processor and the processing speed was never a problem. Nevertheless we believe it should work reasonably well on slower computers too, as multithreading was not used and the limit was the 30 frames per second maximum frame rate of the camera.

12 Applications

There are many possible applications for the smile detector. First of all, the smile detector can be used to improve human and computer interaction. A smile can be interpreted as some specific signal to a computer, resulting in some predefined action by the computer program. It can also be used in systems involving artificial intelligence, as it provides information about a person and hence enables a sophisticated response.

12.1 Smile Detector Lock

The smile detector was initially created with a specific application in mind: The idea was to use the detector to control a digital lock. The digital lock is connected to an Arduino board [30] (see Figure 7), which is communicating with the computer that runs the smile detection algorithm. When the algorithm detects a smile, then the computer sends a command to the Arduino board to open the lock for a few seconds. If the algorithm does not detect anything, then the lock will of course not be opened.

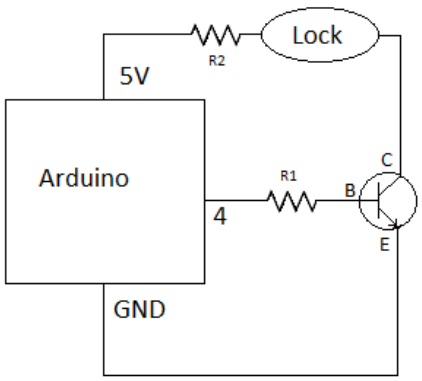

Figure 7: The electric lock is controlled using digital output from PIN 4, the lock gets power from 5V output PIN and is switched using a transistor.

The idea is to install the locking system to a seminar room to make it "guard" the entrance to the room. The door of the seminar room will be opened only if

a smiling person is detected by the smile detection algorithm. The smile detection lock helps to increase the moral of the people by making them smile more when entering the room. The system is very suitable for a seminar room, because a seminar room generally does not need any real security. An interesting and entertaining application is needed instead.

12.2 Plugin for an Instant Messenger

People use instant messenger applications like Skype to communicate with friends and colleagues and it is common to use many different smileys. It is possible to apply the smile detection algorithm to automatically enter a smiley whenever a person smiles during an instant messaging conversation. This kind of system can potentially be implemented as a plugin for a messenger application.

12.3 Capturing a Photograph Automatically

A common challenge when trying to take a good photo is to capture the moment when a person is smiling. The smile detector can make this process much more convenient, as it is possible to take the photo automatically at the exact moment when a person smiles. There are already cameras with this kind of functionality on the market, but it is likely that the accuracy can be improved.

12.4 Robot Interaction

Robots can be successfully used in many different fields of life. It is very likely that in the future they will also be used for tasks that require interaction with people. A large amount of information in a human communication is delivered using speech, but a substantial part is also delivered using facial expressions. Hence it is necessary for a robot to understand them. The smile detection algorithm can be used to obtain a part of this information.

13 Conclusion

We implemented a smile detection algorithm. It works by locating a face of a person and calculating several of its reference points like the corners of the mouth, corners of the eyes, nostrils and chin. These reference points are tracked across several frames using optical flow. The algorithm detects when a person is smiling based on the motion of the reference points.

The algorithm works in real-time speed. The detection rate depends on how to adjust the thresholds, it is possible to detect nearly all smiles, but this also results in a higher false positive rate. Reasonably chosen thresholds give a low misdetection rate while most of the smiles are detected.

The algorithm was also used for an interesting application: a smile detector lock. It is a system where our smile detection algorithm controls an electric lock, the lock is opened when a person smiles. The application is supposed to be installed to control access to a seminar room, as it does not need any security, but it is good to entertain people who enter the room.

References

[1] R. Gonzales, R. Woods, "Digital Image Processing," 3rd Ed, Prentice Hall, 2007.

[2] G. Thomas, M. Weir, J. Hass, "Thomas' Calculus," 12th Ed, Pearson, 2009.

[3] C. Darwin, "The expression of the emotions in man and animals," Oxford University Press, 1998.

[4] P. Ekman and W. Friesen, "Constants across cultures in the face and emotion," *Journal of personality and social psychology,* vol. 17, no. 2, 1971, pp. 124-129.

[5] M. Suwa, N. Sugie and K. Fujimora, "A preliminary note on pattern recognition of human emotional expression," *International Joint Conference on Pattern Recognition,* 1978, pp. 408-410.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 2001., vol. 1, pp. 511-518.

[7] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," *In Proceedings IEEE Conf. on Image Processing*, vol. 1, 2002., pp. 900-903.

[8] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, 55, no. 1, 1997., pp. 119-139.

[9] B. Horn, B. Schunck, "Determining Optical Flow," *Artificial Intelligence,* vol 17, 1981, pp. 185-203.

[10] B. Lucas, "Generalized image matching by the method of differences," *Ph.D. dissertation*, Robotics Inst, Carnegie Mellon Univ, July 1984.

[11] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proc. of Imaging Understanding Workshop*, 1981., pp. 121-130.

[12] J. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[13] BioID AG, "BioID face database," http://www.bioid.com/downloads/software/bioid-face-database.html, 2014.

[14] C. Harris and M. Stephens, "A combined corner and edge detector," *Proc. Alvey Vision Conference,* 1988, pp. 147-151.

[15] H. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," *Tech Report CMU-RI-TR-3,* Carnegie-Mellon Univ, Robotics Inst, 1980.

[16] J. Shi and C. Tomasi, "Good Features to Track," *IEEE Conf. Computer Vision and Pattern Recognition,* June 1994, pp. 593-600.

[17] S. Smith and J. Brady, "SUSAN - A New Approach to Low Level Image Processing," *International Journal of Computer Vision,* vol. 23, May 1997, pp. 45-78.

[18] Y. Huang and C. Fuh, "Face Detection and Smile Detection," *Proc. of IPPR Conference on Computer Vision, Graphics and Image Porcessing*, Shitou, Taiwan, 2009, A5-6, p. 108.

[19] K. Srinivasa, I. Shivalingaiah, L. Gracias, N. Ranganath, "Facial Expression Recognition System Using Weight-Based Approach," *Ubiquitous Computing and Communications Journal*, vol. 5, 2010.

[20] Y. Yacoob, L. Davis, "Recognizing Human Facial Expressions From Long Image Sequences Using Optical Flow," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 6, June 1996, pp. 636-642.

[21] K. Mase, "An Application of Optical Flow-Extraction of Facial Expression-," *Proc. Machine Vision Applications,* 1990, pp. 195-198.

[22] L. Wiskott, J. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition and gender determination," *Proc. Int. Workshop on Automatic Face Gesture Recognition,* 1995, pp. 92-97.

[23] Y. Shinohara, N. Otsu, "Facial expression recognition using fisher weight maps," *IEEE Conf on Automatic Face and Gesture Recognition,* May 2004, pp. 499-504.

[24] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience,* vol. 3, no. 1, 1991, pp. 71-86.

[25] P. Belhumeur, J. Hespanha and D. Kriegman, "Eigenfaces vs. Fisherface: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, July 1997, pp. 711-720.

[26] M. Lyons, J. Budynek and S. Akamatsu, "Automatic classification of single facial images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 12, Dec. 1999, pp. 1357-1362.

[27] H. Soyel and H. Demirel, "Facial expression recognition based on discriminative scale invariant feature transform," *Electronics letters,* vol. 46, no. 5, March 2010, pp. 343-345.

[28] H. Soyel and H. Demirel, "3D facial expression recognition with geometrically localized facial features," *23rd International Symposium on Computer and Information Sciences,* Oct. 2008, pp. 1-4.

[29] P. Rai and M. Dixit, "Smile Detection via Bezier Curve of Mouth Interest Points," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 3, no. 7, July 2013, pp. 1802-1806.

[30] Arduino, "ArduinoBoardLeonardo," http://arduino.cc/en/Main/ArduinoBoardLeonardo, 2014.

[31] "Arch Linux," https://www.archlinux.org, 2014.

[32] LinuxTVWiki, "Video4Linux APIs," http://www.linuxtv.org/wiki/index.php/Development:_Video4Linux_APIs, 2014.

[33] OpenCV, "The OpenCV Library," http://opencv.org, 2014.

[34] Qt Project, "The Qt Library," http://qt-project.org, 2014.

[35] Qwt User's Guide, "Qwt - Qt Widgets for Technical Applications," http://qwt.sourceforge.net, 2014.

Non-exclusive licence to reproduce thesis and make thesis public

I, Andres Traumann (date of birth: 28th of March 1992),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Smile Detector Based on the Motion of Face Reference Points

supervised by Gholamreza Anbarjafari

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 12.05.2014