Facial Landmark Tracking on a Mobile Device

Y.C. van Wettum
Student Bachelor Electrical Engineering
University of Twente, Enschede, The Netherlands
Supervisor: Dr.Ir. L.J. Spreeuwers

***Abstract*—Face recognition is an important research topic in computer vision and biometrics and has become an important component in society. Face recognition is used in crime fighting and person identification at airports. With the increasing interest in applications of face recognition, the need for real-time face recognition has arisen. In order to perform face recognition, the facial features (e.g. eyes, nose, mouth) have to be localised. Usually, this is done by 'detection' in each individual frame. This is an inefficient method for dynamic footage, since the location information of facial features from previous frames is not used. A more efficient method is 'tracking'.**

This paper is about the feasibility of facial feature localisation in dynamic footage using video tracking algorithms. This research is conducted with the aim of improving the speed of a 2D face recognition system on a smartphone. A C++ framework is realised in order to compare 4 video tracking algorithms and 2 widely used facial landmark localisation software libraries. The region-of-interest of the different video tracking algorithms is set automatically. A database is composed to evaluate the different algorithms on accuracy, precision, robustness and speed. With an ideal landmark detector, 2 of the 4 video tracking algorithms are able to outperform a state-of-the-art facial landmark localisation method on precision and accuracy.

***Index Terms*—Facial landmark tracking, detection, real-time.**

I. Introduction

Face recognition is one of the most studied research topics within the field of computer vision during the last decades. It has a wide range of applications, e.g. surveillance, human-computer interaction and secure access. The accurate localisation of facial features, also referred to as facial fiducial points or facial landmarks [1], is of great importance to the performance of a face recognition system [2]. Examples of such landmarks are eye corners, corners of the mouth and tip of the nose. Facial landmarks are used to calculate the face pose and the face scale that allows for normalisation. The normalisation is needed to make the faces comparable and make face recognition possible. Only a few primary landmarks are needed, the outer corners of the eyes and the tip of the nose are already enough to perform the normalisation [3].

In static footage (images) the facial landmarks are localised by facial landmark 'detection'. In dynamic footage (video) it is unnecessary to perform detection in each frame. When detection is performed in each individual frame, information from preceding frames is not used. The location of facial landmarks in preceding frames can be used to find the facial landmarks in the current frame. Therefore, it is more efficient to perform 'tracking' instead of detection. In this paper, tracking is defined as the process of locating a moving object over time using information of previous frames. An advantage of tracking over detection is that an object tracking method is more robust to changes in pose/viewpoint, which an object detection method may not be trained for. To perform accurate localisation of facial landmarks in dynamic footage an algorithm using both detection and tracking is needed. In this paper the focus is on the tracking part. The two research questions on which this paper is based are:

1) *Which video tracking algorithms are usable for real-time facial landmark tracking on a mobile device?*
2) *What is the performance of the video tracking algorithms used for facial landmark tracking, in comparison with existing facial landmark localisation methods?*

Facial landmark tracking has been proven extremely challenging. This is due to a variety of factors, e.g. the variety in human faces, occlusions, illumination changes, different expressions and pose variations. This led to a whole range of solutions to the tracking problem. Some tracking algorithms claim to be better than state-of-the-art tracking algorithms. Many times the comparison is unfair because the algorithm is trained on the same database as it is tested on [4]. There are a few benchmark papers in which (facial landmark) tracking algorithms are tested on the same database [4, 5]. These benchmarks provide insightful information about the performance of different tracking algorithms. The normalised cumulative error curve is often taken as measure for the performance of a tracking algorithm. To rank the algorithms in order of performance, the area under the normalised cumulative error curve is used. Although the cumulative error plot is a strong measurement tool for the overall precision and accuracy, there are more aspects which have an impact on the performance of an algorithm. An additional measurement tool is used in this paper to evaluate the robustness of the compared algorithms.

The aim of this paper is to explore which video tracking algorithms are suitable for real-time facial landmark tracking. This exploration is conducted with aim of improving the speed of a 2D face recognition system on a smartphone. The current face recognition system on the smartphone uses STASM [6] to localise the facial landmarks. STASM is a C++ software library for finding facial landmarks in faces. STASM is too slow to make real-time facial recognition possible. Furthermore, STASM is not a tracking algorithm, for each frame a face is detected and a model of points is fitted onto the face. As described above, it makes more sense to do tracking instead of detection in dynamic footage.

In order to find an answer to the research question, sub-questions have been formulated and categorised as follows:

1) State-of-the-art
    a) Which algorithms are used for facial landmark localisation?
    b) Which video tracking algorithms perform in real-time and are sufficiently accurate for facial landmark tracking?
2) Performance evaluation
    a) Which databases can be used for evaluation of the tracking algorithms?
    b) How is accuracy and precision defined for tracking algorithms?
3) Parameters
    a) What is the optimal region of interest for video tracking algorithms?
    b) At what intervals should detection of the facial landmarks occur?

An answer to the sub-questions is obtained by doing both literature research and conducting experiments.

The remainder of this paper is organised as follows: In section II, related work on tracking algorithms and facial landmark localisation is discussed. The realised framework is described in section III. Section II and III give an answer to sub-questions: 1a, 1b and 2a. The next section IV describes the experiments that are conducted to compare the performance of the different tracking algorithms. This is followed by section V in which the results are presented. These two sections give an answer to the remaining sub-questions: 2b, 3a and 3b. In section VI the findings are discussed. Finally, in section VII the paper is concluded by answering the two main research questions of this paper and the recommendations for further research.

II. Related Work

A. Tracking algorithms

Object tracking has been investigated extensively during the last decades. There are different approaches and they all have their own advantages and disadvantages. There is no perfect method that handles occlusions, rotations, illumination changes etc. the best. The aim of this section is not to discuss all the existing tracking algorithms but to discuss the methods that are interesting for facial landmark tracking. Also methods that are used for facial landmark localisation are discussed.

It is difficult to categorise object tracking algorithms because tracking algorithms often use a combination of several techniques to locate objects. S. Phimoltares et al. categorise the algorithms based on the type of information used. They define five categories: geometry-based, color-based, appearance-based, edge-based and motion-based landmarking algorithms [7]. O. Çeliktutan et al. use a different categorisation for tracking algorithms, first they make a distinction between, model-based methods and texture-based methods. These main categories are subdivided respectively in two sub categories explicit methods and implicit methods, transform-based methods and template-based methods [1].

In this paper the distinction between *online* and *offline* tracking methods is made. Online methods include the approaches that do not need training in order to perform tracking. These methods use information of previous frames in the tracking process [5]. Consequently, offline tracking methods are the methods that need training in order to be able to track objects. The offline methods often include an object detector and then fit a statistical model on the detected object. These methods visually perform the task of a tracker but actually perform detection in each frame. The interest in this paper is in the online tracking algorithm because they do not need extensive training and are therefore widely applicable.

Popular examples of offline methods use Active Shape Models (ASMs) [8] and Active Appearance Models (AAMs) [9]. Both models require training of a statistical model. Methods that make use of an ASM iteratively deform a statistical shape model, which is represented by a set of model points, to fit to a target object. The ASM algorithm looks also at the image appearance around every model point for the best texture match. The simplest method is to assume model points lie on strong edges. The AAM is a generalisation of the ASM. The ASM approach uses only information near the modelled edges, the AAM makes use of all the information in the image region (i.e. the texture across the target object). The statistical shape model limits the degree of deformation of the object that can be detected. Furthermore, both ASMs and AAMs require a good initialisation, otherwise these methods are prone to getting stuck at local minima [1] (i.e. when the initialisation is too far from the true solution, the model will not converge). A major advantage of the offline approach is that this circumvents drifting [10]. Popular landmark localisation software that uses an ASM is STASM [6].

Recently, a new high-speed landmark localisation algorithm was presented in [11]. An ensemble of regression trees (ERT) is used to estimate the faces landmark positions. An implementation of this algorithm is available in the C++ software library: Dlib [12]. The ERT approach is more accurate than STASM [11], tested on the HELEN database [13]. Another popular C++ library is Deformable Shape Tracking (DEST) [14] which is also based on the ERT approach and makes use of a statistical model. In figure 1 an example of the iterative statistical shape model deformation is shown using DEST. Respectively, from left to right, iterations 1, 6 and 11 (last iteration) are shown. In figure 1, the landmark points of the eyes are too far from the true solution and do not converge.

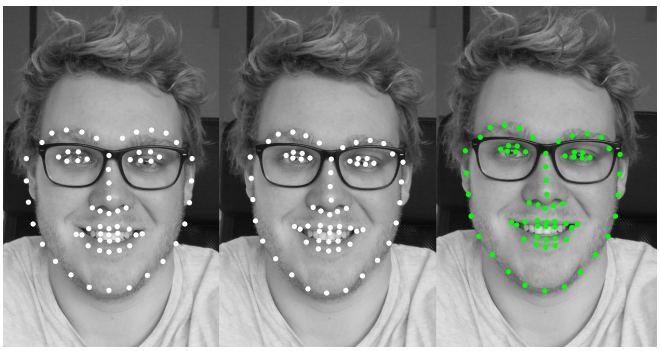


Figure 1: Example of iterative statistical shape model deformation using DEST.

A widely used approach to the tracking problem is optical flow estimation. Optical flow is used to study a large variety of motions. One of the most popular methods is the Lucas-Kanade algorithm. This method is developed by Bruce D. Lucas and T. Kanade and presented in [15]. The algorithm estimates the displacement for the local neighbourhood of a pixel. One pixel does not give enough information for matching with another pixel. It is better to use multiple pixels, i.e. a neighbourhood around a pixel. For every point that is tracked a movement vector is obtained by comparing the pixel intensities of two consecutive images. Many improvements have been made to the Lucas-Kanade algorithm. J. Bouguet reduces the resolution of images first and then applies the Lucas-Kanade method [16], he proposed a pyramidal implementation of the classical Lucas-Kanade algorithm.

An efficient approach to real-time tracking is introduced by D. Comaniciu et al. in [17]. The proposed approach uses an isotropic kernel to spatially mask the target object. Then a smooth similarity function is applied which translate the tracking problem to a maximum similarity search around the previous location. The mean shift procedure, first described in [18] by K. Fukunaga et al., is used to perform optimisation. The kernel-based approach is described in depth by D. Comaniciu et al in [19]. Since the introduction of the kernel-based approach, this has been a widely used approach in object tracking. The top ranking tracker, named Struck [20], in the recent benchmarking paper of Y. Wu et al. [5] makes use of kernels.

Adaptive tracking-by-detection algorithms [5] have shown to be very successful. These methods update an object detector during run-time. Kalal et al. proposed a successful tracker called Predator [21]. This method decomposes the problem into three sub tasks: Tracking, Learning and Detection (TLD). Results from the tracker are used as training data for the detector. The detector uses all appearances of the object that have been observed and can correct the tracker if it fails. Another successful adaptive visual object tracker was presented in [20] by S. Hare et al. The proposed method is called: Structured Output Tracking with Kernels, (Struck). Struck is based on structured output prediction and uses a kernelized structured Support Vector Machine (SVM) which is learned online to provide adaptive tracking. To allow for real-time performance, a budget maintenance mechanism is realised for the online structured output SVMs.

Correlation filters recently obtained considerable attention due to computational efficiency. J. F. Henriques et al. proposed a method, named CSK [22] and uses correlation filters in a kernel space. This method is able to process hundreds of frames per second [5]. KCF (Kernelized Correlation Filter) method is an improvement to CSK and is proposed in [23]. KCF has been shown successful and able to outperform TLD and Struck, while running at hundreds of frames per second [23].

To aid object tracking, feedback systems are used. A popular method used in control theory is the Kalman filter [24]. Another often used method to aid tracking is particle filters [25]. The particle filter approach is inspired by the inability of the Kalman filter to perform object tracking with significant background clutter and noise. The Kalman filter assumes the system is linear and that noise in the system and measurements is white and Gaussian. These assumptions are often invalid [26].

B. Benchmarking

There is a variety of different databases available to evaluate object trackers. For object tracking a popular benchmark is described in [5]. Although, there are a lot of benchmarks available for facial landmark localisation in static images. Limited effort has been made towards benchmarking facial landmark tracking algorithms in videos. In [4] the first comprehensive benchmark for long-term facial landmark tracking is presented. The videos in the corresponding video database, called 300 Videos in the Wild (300-VW) database [4, 10, 27], have been annotated using the mark-up as is shown in figure 2. Another popular annotated video database is the Talking Face [28], which has a simular mark-up as the 300-VW database.

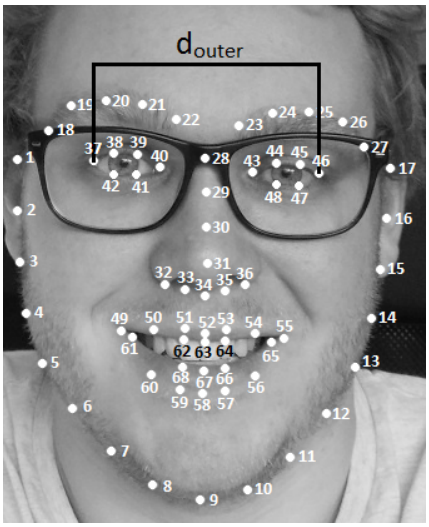


Figure 2: The 68 points mark-up used in the 300-VW database.

III. Implementation

In the exploration towards a suitable method for facial landmark tracking on a mobile device, several algorithms are considered applicable. A C++ framework is realised in Microsoft Visual Studio 2015 in order to compare the different video tracking algorithms. The framework includes 4 video tracking algorithms and 2 state-of-the-art facial landmark localisation software libraries. In this section the framework is described. Furthermore, the properties of the implemented video tracking algorithms are discussed.

A. Framework

In this paper, the focus is on tracking algorithms. A stand-alone facial landmark localisation system needs a landmark detector. The landmark detector is used to initialise the tracker. In order to compare the effect of different tracking algorithms an 'ideal' landmark detector is used. The annotated data files (ground-truth files) of the videos are used as an ideal

landmark detector. The coordinates of 5 landmarks (outer eye corners, nose tip and mouth corners) are passed to the different tracking algorithms. The trackers will track the points for a number of frames. After a certain number of frames, the tracker is re-initialised by the detector. After re-initialisation, the trackers will again track the facial landmarks. A schematic representation of the framework using a detection rate of once every 50 frames is shown in figure 3. In this paper, the detection rate is defined as the number of frames per detection.

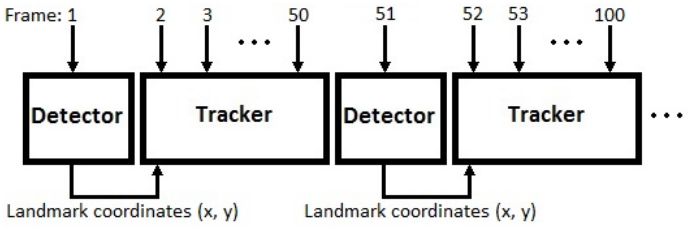


Figure 3: Schematic overview of the framework, detection rate is 50 frames/detection

The outer corners of the eyes, nose tip and mouth corners are chosen as landmarks to track because these landmarks can be used for normalisation, as was mentioned in section I. An example of the video output of the framework is shown in figure 4. The tracked landmarks are represented by (green) dots. Each landmark is in the middle of a (blue) square. This square is called the region-of-interest (ROI). The ROI, also referred to as window size or search area, is different for each video. The measures of the ROI are dependent on the scene in the video and the scale of the face, this will be elaborated in section IV-B. The results of the different trackers are visualised using different symbols and colors. For each tracker a different symbol (triangle, square etc.) and color is used. As a result, the output of the various trackers can be monitored.

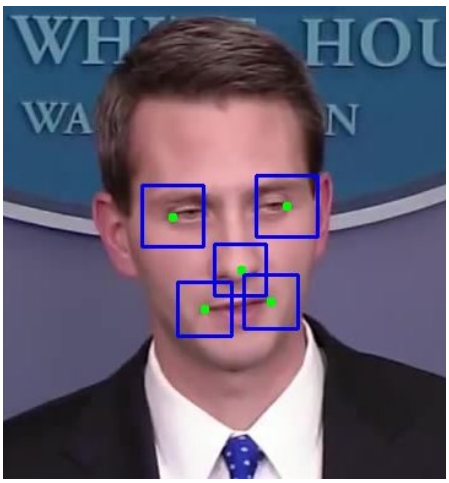


Figure 4: Example of a tracked frame with 5 landmarks and their ROI. The image is obtained from the 300-VW database [4, 10, 27].

In the main function of the framework, the frames of the videos are loaded and stored in a 2-dimensional array. This array is passed to the individual trackers. In the first frame of each video the inter-ocular distance (IOD) is calculated. The IOD is defined as the Euclidean distance between the outer corners of the eyes. In figure 2, the IOD is denoted by d_{outer}. The IOD is calculated using the obtained coordinates from the ideal landmark detector. The IOD is used to set the ROI of the different landmarks. The ROI of each landmark is different for each tracking algorithm. Results of tracking algorithms are stored in separate files. These files are used for testing and evaluation.

B. Lucas-Kanade (LK) Point Tracker

Optical flow is considered suitable for landmark tracking on a mobile device because it can be used to observe a large variety of motions [26], i.e. static observer and moving object, moving observer and static object, moving observer and moving object. This last scenario is most likely on a mobile device. A sparse optical flow method is used because specific points are of interest. The Open Source Computer Vision (OpenCV) library [29] includes an implementation of the Lucas-Kanade (LK) method. This implementation is based on a sparse iterative version of the Lucas-Kanade optical flow in pyramids [16].

The LK tracker from the OpenCV library is implemented in the C++ framework because it is able to run in real-time. The real-time performance is facilitated by the use of a pyramidal representation of the frames. The pyramidal representation allows the tracker to handle large pixel motions, i.e. larger than the used ROI. The ROI can be kept relatively small which is advantageous for the computational load. The pyramid representation is built in a recursive manner, starting from the original frame. Assuming the Full HD camera of the mobile device is used, it would be useless to go above a pyramidal level of 5 (5 lower resolution frame representations). For example, 1920 x 1080 pixels is the resolution of the image at level 0. The image resolutions of the subsequent levels are respectively 960 x 540, 480 x 270, 240 x 135, 120 x 67 and 60 x 33 pixels. In the framework 5 pyramidal levels are used for resolutions of 1280 x 720 pixels and 4 pyramidal levels are used for all resolutions below 1280 x 720 pixels.

OpenCV uses a corner point detector to initialise the LK tracker. The corner point detector is not used in the framework because the framework uses an ideal landmark detector. Furthermore, the corner point detector finds the most prominent corners in the image. Landmarks such as the nose tip are often not defined by prominent corners.

C. Discriminative Scale Space Tracker (DSST)

In the face recognition system on a mobile device the facial image is captured by hand. This will introduce a varying scale of the face in the captured video. The Discriminative Scale Space Tracker (DSST) [30] performs well in image sequences with significant scale variations. Moreover, the DSST is the best performing tracker in the Visual Object Challenge (VOT) 2014 [31]. Therefore, the DSST tracker might be a good solution to the landmark tracking problem. The DSST implementation of the Dlib C++ software library is used.

The DSST is an extension to the Minimum Output Sum of Squared Errors (MOSSE) tracker [32] with robust scale

estimation. The MOSSE tracker is limited to estimating the translation between frames, the DSST tracker adds robust scale estimation. The MOSSE tracker is initialised in the first frame. The object is tracked by correlating the trained filter (which models the appearance of the object) over a search window. The maximum value in the correlation output is the new position of the object. The correlation filter of the tracker is then updated, the filter is trained during run-time. The correlation is computed in the Fourier domain because computing the correlation is an element-wise multiplication in the Fourier domain. The DSST estimates the target size by learning a one-dimensional discriminative scale filter. The scale filter is trained by extracting sample patches of different scales around the current position of the object. Intensity features and the HOG features (histogram of oriented gradients) are used for the translation filter.

D. Kernelized Correlation Filters (KCF)

The high-speed tracking algorithm with Kernelized Correlation Filters (KCF) [23] is also implemented in the framework. This tracking algorithm is considered suitable for landmark tracking because it performs in real-time [31]. A realisation of the KCF algorithm is available in the OpenCV library. The implementation is extended with color features which result in superior performance for visual tracking [33].

E. Structured Output Tracking with Kernels (Struck)

The Structured Output Tracking with Kernels (Struck) method [20] is based on structured output prediction. The method uses a kernelized structured output SVM, which is learned online. This allows for adaptive tracking, which is beneficial for facial landmark tracking. Facial landmarks deform due to facial expressions. As was mentioned in the previous section, this method uses a budget mechanism in order to perform in real-time.

The Struck algorithm is implemented using the code from the authors. The code of the authors is open source and is available on GitHub [34]. The code is modified in order to include it in the framework. No changes have been made to the operation of the algorithm. The default settings are used, this means Haar features and a Gaussian kernel are used.

F. Zero-effort tracker

For reference purposes a zero-effort tracker is realised in the framework. This tracker does nothing, as the name already implies. After detection of the landmarks, the tracker does nothing and stays in the same place. The zero-effort tracker is used as a baseline in the performance experiments. The experiments are described in the next section.

G. Landmark localisation software

For comparison purposes, facial landmark localisation software is included in the framework. For detection of highly non-rigid objects, cascade regressors are currently the state-of-the-art. An ensemble of regression trees (ERT) can be used to localise landmarks and achieve super real-time performance with accurate results [11].

The Dlib software library includes an implementation of the ERT method with a face detector. The Dlib Facial Landmark Detector (DFLD) is included in the framework. Furthermore, the DEST software library is included in the framework. Both software libraries use the ERT method to find facial landmarks. The two software libraries use different face detection methods. The face detector in DFLD is realised using Histogram of Oriented Gradients (HOG) features combined with a linear classifier. Furthermore, an image pyramid and sliding window detection scheme are used. The DEST library uses the face detector of OpenCV. OpenCV uses the Viola and Jones algorithm for face detection. This algorithm uses Haar features and a cascade of classifiers. In comparison with the face detector of DFLD, the face detector of OpenCV does not use pyramids. Instead of pyramids, the features are scaled. Both libraries come with pre-trained classifiers and shape models.

DLFD and DEST perform face detection in each frame. An advantage of the face detection in each frame is that the methods do not drift. However, face detection is computationally heavy and takes the most time. Therefore tracking algorithms will be faster but are prone to drifting.

A disadvantage of DFLD and DEST is that these methods are prone to pose variations. Usually models are trained for frontal faces. If a face in the video turns to the side, the face detector fails to recognise it as a face. The video tracking algorithms are more robust to changes in pose, assuming the video tracker is initialised correctly. In a practical application the face detector in the framework, see figure 3, also needs training. If the tracker is not initialised correctly it will track points without physical meaning. Therefore, a practical application should have a feedback system. The feedback system should indicate whether the detected points are real landmarks.

IV. Experiments

In this section the experiments are described that are conducted to analyze the performances of the different tracking algorithms. The 4 video tracking algorithms (KCF, DSST, LK, Struck) and 2 state-of-the-art landmark localisation software libraries (DEST, DFLD) are compared. In this paper the performance is defined as a combination of accuracy, precision, robustness and speed.

A. Database

Evaluation of the algorithms is done using annotated videos of the 300-VW database [4, 10, 27] and the Talking Face (TF) database [28].

The 300-VW database contains 114 annotated videos. All videos show only one person. The average duration of a video is 64 seconds (25 - 30 frames/second). The videos are annotated semi-automatically, visual inspection was performed to find incorrectly annotated landmarks. All frames have been annotated using the same mark-up (68 landmarks) as is shown in figure 2. The videos vary in pixel resolution and spatial

resolution. The database is divided into 3 categories. The first category contains videos in well-lit conditions and without occlusions of another person or a hand. The second and third category contain respectively videos in unconstrained conditions and videos in completely unconstrained conditions including the illumination.

The TF database consists of one video that contains 5000 frames and shows a person engaged in a conversation. It corresponds to 200 seconds of recording (25 frames/second). The resolution of the video is 720 x 576 pixels. The data is semi-automatically annotated with a 68 point model. The model has corresponding points with the eye corners, nose tip and mouth corners of the 300-VW database mark-up. The frames in the database have been visually checked and the annotation is sufficiently accurate to represent facial movements [28]. Therefore, the database is considered suitable for evaluation of facial landmark tracking algorithms.

To keep the experiments manageable, a small database was composed of 20 videos. The database contains 19 videos of the category one videos of the 300-VW database and the video of the TF database. Visibility of the face is necessary in an application for face recognition. Therefore, the videos in category two and three of the 300-VW database are left out. The videos in the composed database aim to evaluate algorithms that could be used in naturalistic well-lit conditions. The composed database is divided into the following two categories:

- Category one: Contains 14 videos of people recorded in well-lit conditions. The database consists of 13 videos from the 300-VW database and the video from the TF database. This category contains facial videos without occlusions such as glasses, hair and beards. Example frames of category one are shown in figure 5.
- Category two: Contains 6 videos of people recorded in well-lit conditions. The 6 videos are obtained from the 300-VW database. This category contains facial videos with occlusions such as glasses, hair and beards. Example frames of category two are shown in figure 6.

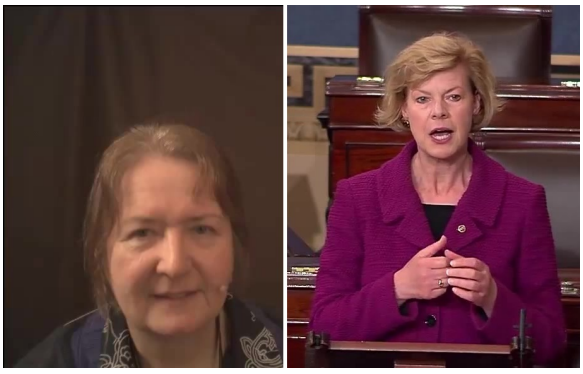


Figure 5: Example frames of category one videos in the composed database. [4, 10, 27]

The videos are saved as image sequences with the name corresponding to the frame number. This is done because the annotation is stored in separate files for each frame. Loading the frames with corresponding annotation file is easier by using

Figure 6: Example frames of category two videos in the composed database. [4, 10, 27]

image sequences. All videos are set to a length of 500 frames with in the first frame a complete view of the frontal face.

B. Region-of-interest optimisation

To allow for an automatic landmark tracking system, the ROI of the tracker has to be determined automatically. The tracker must adjust the ROI according to the scale of the face in the given input footage. The ROI is defined here as a square region with in the middle the corresponding landmark, see figure 4. The size of the face in the video varies with the distance between the face and the camera. When the scale of the face is large, a too small ROI for the facial landmarks will decrease the performance of the tracker and vice versa. The best window size is dependent on different factors, e.g. the speed of the movement and background. This experiment covers sub-question 3a mentioned in section I. The question on which this experiment is based is:

What is the optimal region of interest that must be used for tracking of the individual landmarks and how can this be made generic for any input video?

From the detected landmarks the IOD can be calculated. The IOD does not differ much among humans and can therefore be used in the determination of the ROI. To find an answer to the question in this experiment, first the relation between ROI and IOD is determined. The relation between the IOD and the measures of the ROI is expected to be linear. The hypothesis can be supported by an example. When a facial video is downsampled by a factor of 2, the maximum amount of movement (in pixels) between frames is also halved. Hence, the dimensions of the ROI can be half the size of the ROI in the original image. In order to determine the relation between IOD and ROI, the following was done:

The ROI is dependent on the scene in the video. In order to keep the scene constant while varying the IOD, a video is down sampled in 4 steps. Each step, the resolution of the video is reduced by 20% in both coordinate directions relative to the original resolution. For example, when the resolution of the original frame is 1280 x 720 pixels, the resolution of the video in the first step is 1024 x 576 pixels. The re-sampling is performed using 'pixel area relation' because this avoids aliasing [35]. After down-sampling the video by 4 different

factors, there are 5 instances of the same video including the original video. The resolution is different for every instance of the video and therefore the IOD is different in each video. Next, the optimal ROI is determined for the different tracking algorithms (KCF, DSST, Struck, LK) in every instance of the video. Then, the ROIs are plotted against the different IODs. This is done separately for the 3 different types of landmarks (the outer eye corners, tip of the nose and the mouth corners). This gives the relation between ROI and IOD for each landmark. The experiment is repeated for 2 other videos (different scenes). The obtained relation between ROI and IOD for the different algorithms is assumed to be true for the rest of the videos. In order to make the estimation complete the optimal ROI is determined for the other 11 videos in category one of the composed database.

As a result of the varying sizes of the faces in the videos, a normalised error measure must be used in order to compare results from different videos. The normalised root-mean-square error (NRMSE) is used as an error measure, normalised with the IOD. This error measure gives a normalised distance to the ground-truth and is given as a fraction of the IOD. The NRMSE is computed as:

$$\text{NRMSE} = \frac{\sum_{i=1}^{N} \sqrt{(x_i^t - x_i^g)^2 + (y_i^t - y_i^g)^2}}{d_{outer} N} \quad (1)$$

In equation 1, the summation is used to obtain an average for both the eye corner landmarks and the mouth corner landmarks. The t and g subscripts define respectively **t**racker coordinates and **g**round-truth coordinates. d_{outer} is the IOD which is defined as the Euclidean distance between the outer corner of the eyes, as is shown in figure 2. Therefore d_{outer} is computed as:

$$d_{outer} = \sqrt{\left(x_{r.eye}^g - x_{l.eye}^g\right)^2 + \left(y_{r.eye}^g - y_{l.eye}^g\right)^2} \quad (2)$$

The optimal ROI is defined as the window size that results in the best performance of the tracker, i.e. the smallest deviation from the annotated data. A normalised threshold error is set at 0.08 of the IOD. Above a normalised error of 0.08 the tracking is too far off [4]. The threshold error is used to establish whether landmarks have been found with sufficient accuracy. A Cumulative Error Distribution (CED) curve can be used to indicate the accuracy and precision of an algorithm. The CED curve shows the percentage of frames within a certain normalised error. The slope of the curve is a measure for the precision of the tracking algorithm and the value at the threshold error is a measure for the accuracy of the tracking algorithm. The Area Under the Curve (AUC) of the CED curve is used to rank the tracking algorithms with respect to precision and accuracy [4, 5]. When the AUC is plotted against the different measures of ROIs, the optimal ROI correspond to the highest AUC value in the graph. A deviation of 2% from the optimal AUC is used to define a left an right boundary for optimal ROI. This margin is used, because in many cases the curve is flat and has no clear optimum. The 2% margin resulted in a better visualisation of the relation between ROI and IOD. The steps to obtain the relation between ROI and IOD are shown with intermediate results in Appendix A.

Finally, an average of all the optimal window sizes is calculated. This results in the ROI as fraction of the IOD, for each landmark and algorithm. The obtained relations are used in the framework to facilitate the automatic landmark tracking system. Only the category one videos from the composed database are used in the determination of the optimal ROI. The category two videos can give a wrong indication of the ROI as a result of occlusions. The obtained values for the optimal ROI as fraction of the IOD are also used for the category two videos. This is valid because the category two videos have similar scenes as the category one videos.

C. Frame rate

The algorithms that are implemented in the framework are selected because of their speed and accuracy. In this experiment the frame rate of the algorithms is determined. The tracking algorithms and both DEST and DFLD are compared with respect to speed. The question in this experiment is:

Which tracking algorithms will perform in real-time on a mobile device?

During run-time of the framework the average frame rate of the different algorithms is calculated and stored in separate files. This is done for all 20 videos of the composed database. The average frame rate is calculated by using timers. Before the start of a tracking algorithm a timer is started. Directly after the tracker is finished the timer is stopped. The timer is not re-initialised between frames. The reason for this is that the average frame rate can directly be calculated. The total value of the timer is divided by the number of frames. This directly results in an average frame rate for the video.

The best way to obtain the frame rates on a mobile device is to run the framework on a mobile device. Due to time limitations, the framework is not realised on a mobile device. However, the framework is not realised on a mobile device an estimation is made of the frame rates. Since the architecture of a laptop and a mobile device are very different you can not simply compare hardware. Therefore, an educated guess is made of the frame rate based on the time that is needed to do face detection on a mobile device [36]. The estimation is based on the amount of time an iPhone 5S needs to complete face detection (using OpenCV). According to [36] an iPhone 5S needs 235ms to complete face detection on an image with a resolution of 463 x 397 pixels. This is the performance on a single core of the iPhone. On average the face detection (also using OpenCV) on a (4GB RAM, 2 x 2.1 GHz CPU, 64 bit Architecture) laptop takes 526ms on an image of 1280 x 720 pixels. The laptop uses approximately 22% of both CPUs which equals one core utilised at 88%. The resolution of the image on which the laptop is performing face detection is approximately 5 times larger than the resolution of the image on which the iPhone 5S is performing face detection. If it is assumed that the duration of face detection scales linear with the resolution of an image, then an iPhone 5S needs 1175ms to complete face detection on an image of 1280 x 720 pixels. An iPhone 5S will perform face detection approximately 2.2 times

slower than the laptop that is used. Therefore, to estimate the frame rates on a mobile device the frame rates obtained on the laptop are divided by a factor of 2.2.

D. Detection rate

An important part of the facial landmark localisation process is the detection phase, see figure 3. In the detection phase a minimum error is introduced, i.e. this is the lowest error the tracking algorithm can obtain. In this paper the detection rate is defined as the frames per detection. The detection rate is an important trade off that has to be made. A lower detection rate, means more detections per video which increases the computational load. A higher detection rate means the tracker is synced less frequently. If the tracker is not robust, a higher detection rate will increase the error by a significant amount. Therefore, it is important to know what the influence is of different detection intervals on the performance of the video tracking algorithms. The overall performance is used in this experiment. The overall performance is defined as the average of the accuracy and precision performances for the 3 different landmark types. The question in this experiment is:

What is the influence on the overall performance of a video tracking algorithm for different detection rates?

For this experiment only the TF database is used, all 5000 frames of the video are used. The optimal ROI of each individual video tracking algorithm is manually determined beforehand by analyzing the first 500 frames. This is done in the same way as as is described in section IV-B. In the determination for the optimal ROI a detection rate of once every 50 frames is used. This correspond to once every 2 seconds, this detection rate is chosen because it is a realistic detection rate for facial landmark detectors.

E. Accuracy, precision and robustness

To compare the different algorithms on accuracy, precision and robustness, all videos of the composed database are processed by each algorithm. The optimal ROIs as fraction of the IOD, obtained in the first experiment, are used to set the window size. The ROI is set in the first frame using the IOD. Each video has a frontal view of the face in the first frame. A detection rate of 25 frames/detection is used. The results of the frame rate experiment show that a detection rate of 25 frames/detection is a realistic detection rate for landmark detectors. CED curves are plotted to determine the accuracy and precision. In order to evaluate the robustness, the normalised point-to-point error is plotted against the frame numbers. This way, the normalised error over time can be shown.

V. Results

A. Region-of-interest optimisation

In order to obtain the relation between ROI and IOD, 5 videos with different resolutions are created, this is described in the previous section. For each landmark, the relation between ROI and IOD is linear. This holds for the 4 different video tracking algorithms. To illustrate the obtained relation, one result is shown in figure 7. The relation between ROI and IOD is also linear for the tip of the nose and the mouth corners. It is superfluous to show all the individual results. The interest is in the average ROI as fraction of the IOD for all videos, this is shown in table 1.

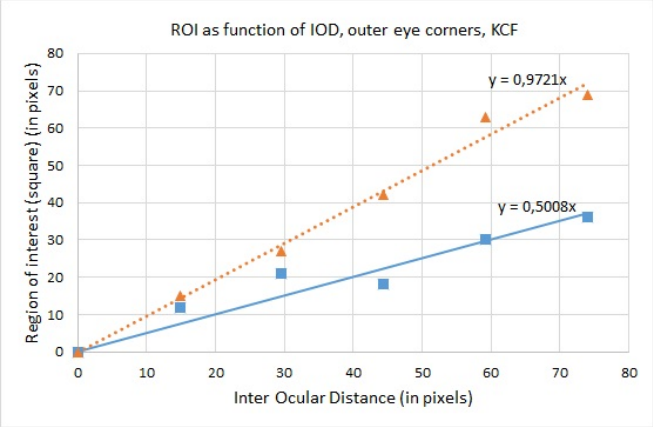


Figure 7: ROI as function of different IODs for the outer corner of the eyes of the KCF algorithm. A video of category one of the composed database is used. The formulas correspond to the lines in the graph. y is the ROI and x is the IOD. The (red) dotted line indicates the right boundary of the optimal ROI an the (blue) solid line indicates the left boundary of the optimal ROI. There is no significant performance change for the tracker in between these lines.

The ROI for the different landmarks in the framework is a square region. The ROI is calculated as a fraction of the IOD, the optimal ROIs can be found in table 1 for each landmark. Multiplying the IOD of the person in the video with the factors in table 1 results in the ROI that must be used for the corresponding landmark.

	KCF	DSST	Struck	LK	LK*
Outer Eye Corners	0.543	0.557	0.581	0.438	0.326
Tip of the Nose	0.455	0.375	0.395	0.219	0.326
Mouth Corners	0.477	0.511	0.711	0.322	0.326

Table 1: The optimal ROI as a fraction of the IOD. In order to find the optimal ROI for a certain landmark and tracker, the IOD should be multiplied with the value in the table.
* The LK algorithm in the framework uses the same ROI for all different landmarks. The average of the ROIs in LK column is used to set the ROI of the LK algorithm in the framework.

B. Frame rate

The frame rates for an iPhone 5S are estimated using the estimation described in section IV and are shown in table 2. The composed database contains 20 videos, 10 videos have resolution 1280 x 720 pixels and 6 videos have resolution 780 x 580 pixels. The rest of the videos have different resolutions and are therefore left out of the experiment. During the frame rate measurements neither custom parallelisation nor GPU ported calls are used.

The frame rate of DFLD and DEST decrease when the image resolution is larger. This is as expected because the algorithm has to detect a face in a larger search area. The frame

Resolution (pixels)	Average IOD (pixels)	KCF (fps)	DSST (fps)	LK (fps)	Struck (fps)	DFLD (fps)	DEST (fps)
780 x 580	115.9	4.7	3.6	73.0	1.4	2.1	1.6
1280 x 720	74.5	9.3	3.6	43.0	1.3	1.1	0.8

Table 2: Average frame rates on a iPhone 5S. These frame rates are obtained by dividing the average frame rates on the used laptop by a factor of 2.2. The table is used to give an indication of the possible frame rates on a mobile device.

rate of the KCF algorithm depends on the size of the ROI, i.e. the frame rate increases for a shorter IOD (smaller ROI). The frame rate of the LK tracker decreases with an increasing image resolution and decreasing windows size (smaller ROI). For the LK tracker the same behaviour is expected as is shown for the KCF tracker. An explanation for difference in behaviour is the different number of pyramidal levels the LK tracker uses for different image resolutions. The DSST and Struck tracker show unexpected behaviour, their frame rate remains constant. The fact that the implementations of the DSST and the Struck tracker are optimised for single object tracking, might be a reason for the constant frame rate. Multiple instances of the trackers are started to track all landmarks. The KCF tracker and LK tracker are able to handle multiple objects and show different behaviour compared to DSST and Struck.

C. Detection rate

The best ROIs for each landmark are determined beforehand, as is described in section IV-D. The used window sizes can be found in table 3. For the Lucas-Kanade implementation the same window size is used for the different landmarks.

	KCF	DSST	Struck	LK*
Outer Eye Corners	49 x 49	41 x 41	51 x 51	31 x 31
Tip of the Nose	45 x 45	37 x 37	49 x 49	31 x 31
Mouth Corners	33 x 33	35 x 35	43 x 43	31 x 31

Table 3: ROIs in pixels used during the detection rate experiment. * The implementation of the LK algorithm uses for all landmarks the same window size.

The AUC of the CED curves for the different algorithms is calculated for different detection rates. The AUC is plotted as function of the detection rate. The results are shown for the detection rate (frames/detection) ranging from 5 to 50 frames/detection, in figure 8. The results of a wider range, ranging from 25 to 250 frames/detection are shown in figure 9.

D. Accuracy and precision

The accuracy and precision is visualised using CED curves. The CED curve of each tracker is visualised in figure 10 and 11. The performance is visualised for the 3 different types of landmarks. Figure 10 shows the average performance of the trackers in the category one videos. In figure 11, the average performance of the trackers in the category two videos is shown. The results are shown on a separate page in order to compare the results. The results are shown on page 11.

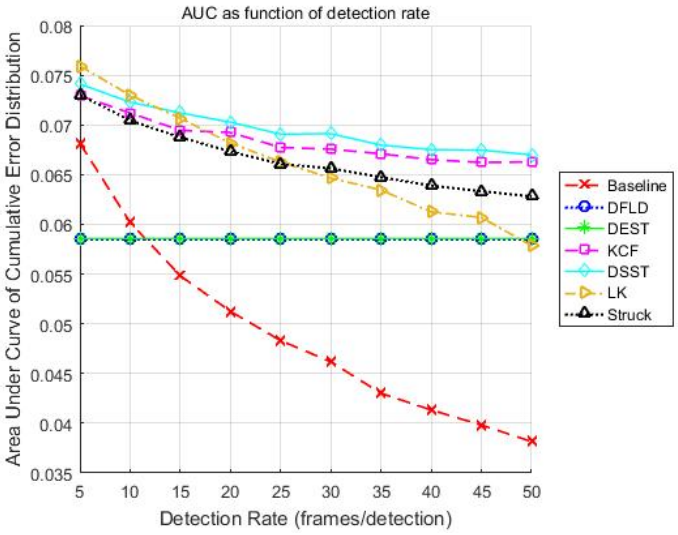


Figure 8: AUC as function of the detection rate. Range of detection rate is 5 - 50 (frames/detection). Note: the y-axis range is different from figure 9.

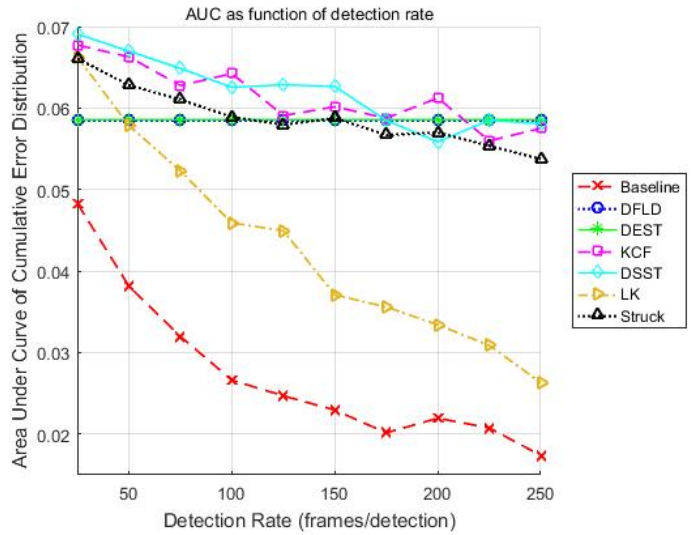


Figure 9: AUC as function of the detection rate. Range of detection rate is 25 - 250 (frames/detection). Note: the y-axis range is different from figure 8.

DEST has a poor performance, due to failures of the face detector. Two examples of face detection failures are shown in Appendix B. Faces are detected in places where no faces are, this results in a large normalised error. When face detection failures happen frequently, this results in a poor accuracy as is the case for DEST.

Note that the video tracking algorithms have an offset in the y-direction. Every 25 frames the video trackers are synced with the ground-truth. Therefore, 20 points of the the 500 tracked points have an normalised point-to-point error of 0. This corresponds with 4% or 0.04 on the y-axis in the CED plots.

E. Robustness

The robustness of the different algorithms is plotted in figure 12 and figure 13 for respectively category one videos and category two videos of the composed database. The results are shown on a separate page in order to compare them. The results are shown on page 12. For readability the normalised error is averaged by the number of detection cycles. A detection rate of 25 frames/detection is used, there are 20 detection cycles. Therefore, the values at frame number 1 correspond to the average normalised error of frames 1 + 25n (for n = 0,1,2 ... 19). The frame number on the x-axis indicates the frame number between two landmark detections. At the first frame the algorithms are set equal to the ground-truth points. The (blue) dotted line is DFLD and does not have a normalised error of 0 at frame 1 because it does face detection in each frame and is not synced with the ground-truth points.

As a measure of deviation, the standard deviation is used and indicated by error bars. In order to compare the scale of the error bars in the robustness plots, the scale of the axis is the same for all robustness plots.

Both the baseline and DEST graph are not included in the robustness plots. These graphs are left out because they resulted in unreadable graphs. The baseline graph rises very quickly, to show the full baseline graph the scale of the y-axis is to large to see differences between the video tracking algorithms. The graph of DEST is excluded because the face detector of DEST detects face at places where there are no faces. This results in a rapidly jumping graph and makes the other results unreadable.

VI. Discussion

Based on the CED curves, it can be said that the LK tracker is the best performing tracker for the nose tip in both the category one and category two videos. Furthermore, the LK tracker performs best for the outer eye corners and the mouth corners in the category two videos. DSST is the best performing algorithm for the outer eye corners and the mouth corners in the category one videos. DEST has a poor performance, this is due to failures of the face detection. The KCF tracker and Struck tracker have similar performances. The CED curves show that if the trackers (using an ideal detector and a detection rate of 25 frames/detection) are initialised correctly, at least 90% of the tracked points is below the normalised threshold error of 0.08. For the category one videos, the trackers: KCF, Struck and DSST, have less accurate results for the nose tip compared to their performance for other landmarks. For the LK tracker it is exactly the opposite. The LK tracker and DSST are the most accurate and precise video tracking algorithms of the 4 compared video tracking algorithms.

From the results of the detection rate experiment, it can be concluded that the LK tracker is the worst performing tracker for long term tracking. In the detection range of 5 to 25 frames/detection, the difference between performance of the different tracking algorithms is small. In the range of 25 to 250 frames/detection the performance of the LK tracker decreases rapidly. Interesting to see is that Struck, KCF and DSST (with an ideal landmark detector) have a better performance compared to DEST and DFLD at a detection rate of 150 frames/detection. This correspond to a detection once every 6 seconds. Important to notice is that a facial landmark localisation system using the structure as is shown in figure 3, with DFLD used to initialise for example a LK tracker, will never outperform the stand-alone DFLD. This results in an increase in speed at the cost of accuracy. The tracker cannot compensate for the errors introduced by the detector.

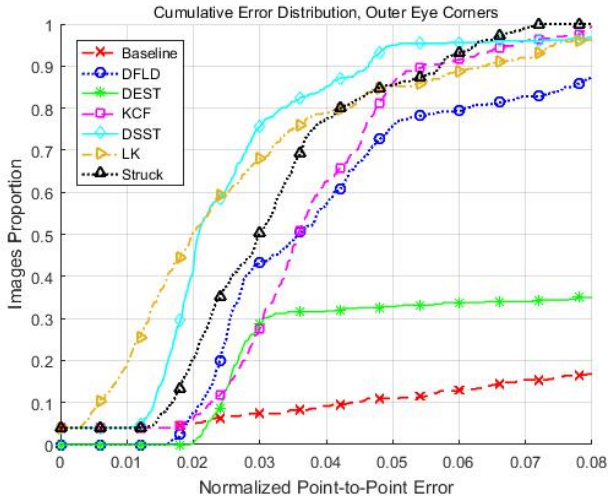
In the robustness plots, the LK tracker is significantly more robust during tracking of the nose tip, compared to the other tracking algorithms. The standard deviation of the LK tracker for the nose tip is approximately 3 times lower compared to the other tracking algorithms in the category one videos. For the category two videos the standard deviation of DSST and the LK tracker for the nose tip is significantly lower compared to the other tracking algorithms. The DSST is on average more robust while tracking outer eye corners and mouth corners than the LK tracker. The absolute values of the curve of DSST and DFLD in the robustness plots is approximately identical independent of the landmark type.

Based on the results for the frame rates, the LK tracker is the fastest tracking algorithm and is able to perform in real-time ($\pm$ 20 frames/second). The second fastest tracker is KCF, followed by DSST and Struck. The implementations of DSST and Struck, used in the framework, are not optimised for multiple objects. Although, no custom parallelisation is used, multi-threading might be implemented in the different libraries. Therefore, in order to make a fair comparison between speed of the different algorithms the experiment must be conducted on the same number of cores with the same number of threads.

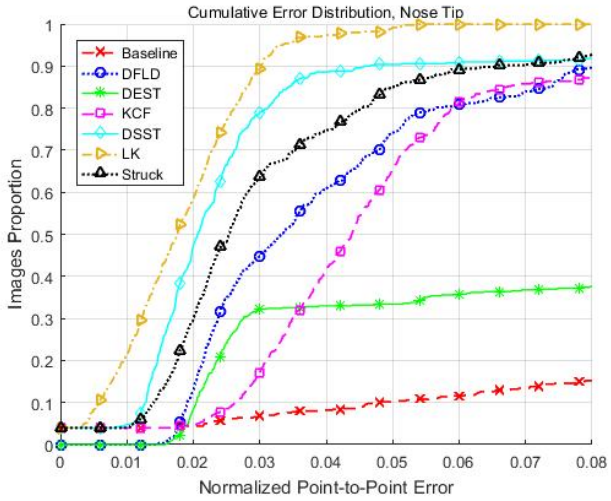
VII. Conclusion

In this paper, the performance of 4 different video tracking algorithms (LK, Struck, DSST, KCF) in facial landmark tracking is compared using a C++ framework. In order to compare the video tracking algorithms with state-of-the-art facial landmark localisation software, 2 facial landmark localisation software libraries (DFLD, DEST) are included in the framework. The ROI of the video tracking algorithms is made generic. The ROI is set as a fraction of the IOD in the first frame. The algorithms are compared on accuracy, precision, robustness and speed.

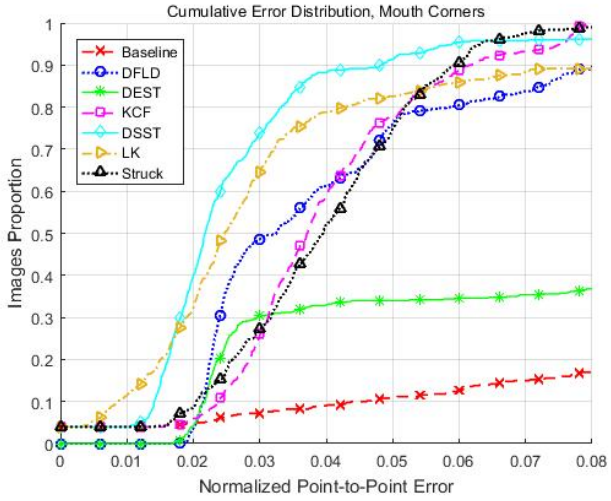
To conclude the paper, the research questions, stated in section I, are answered. The LK tracker and DSST (both using an ideal landmark detector) outperform DFLD on accuracy, precision and speed for the 3 different facial landmarks in both video category one and two. DEST is not used in the performance comparison, because the results are invalid due to face detection failures. The KCF tracker and Struck (both using an ideal detector) do not outperform DFLD on accuracy, precision and robustness. The LK tracker and DSST are usable for facial landmark tracking, but the LK tracker can perform in real-time, which DSST cannot. Therefore, based on the results the LK is the most usable tracking algorithm for real-time facial landmark tracking on a mobile device.



(a) CED curve of the outer corner of the eyes

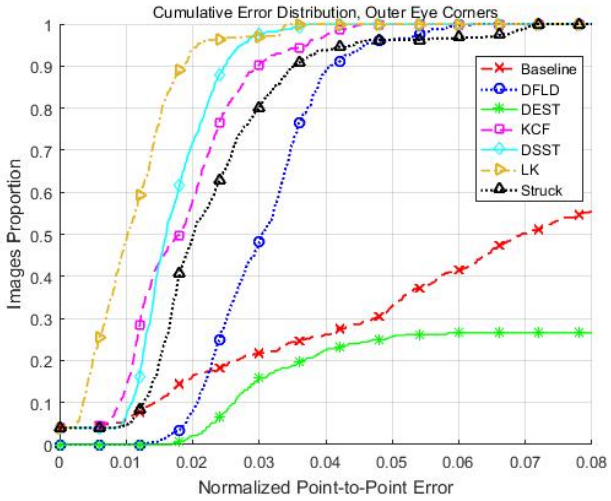


(b) CED curve of the tip of the nose

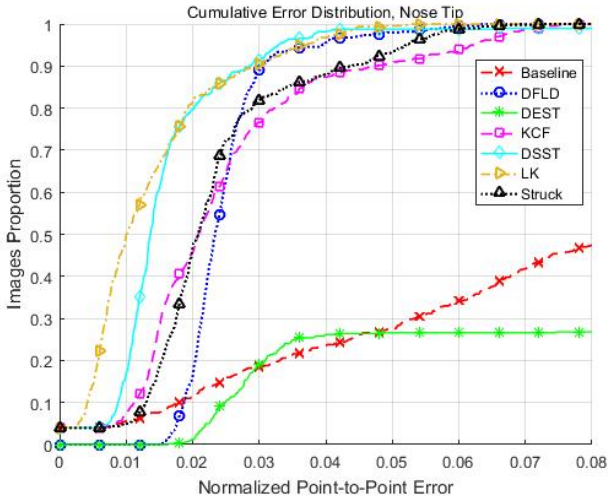


(c) CED curve of the mouth corners

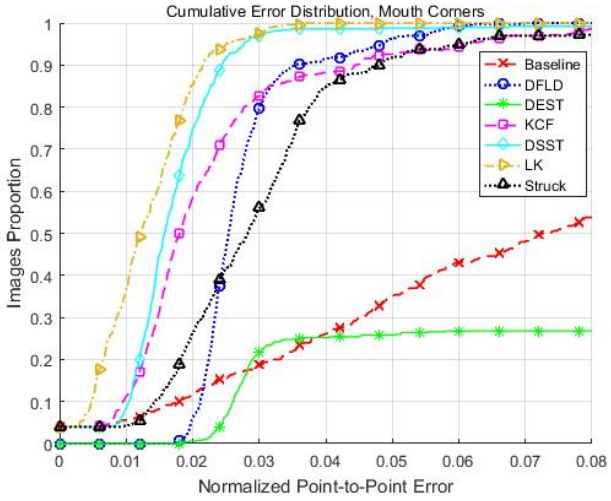
Figure 10: Cumulative Error Distribution curves of the category one videos



(a) CED curve of the outer corner of the eyes

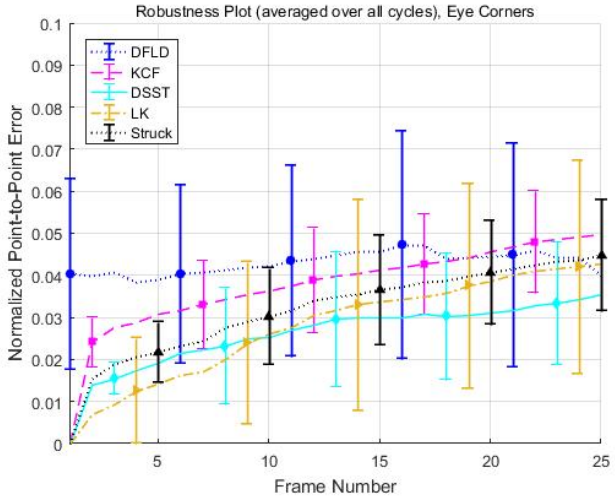


(b) CED curve of the tip of the nose

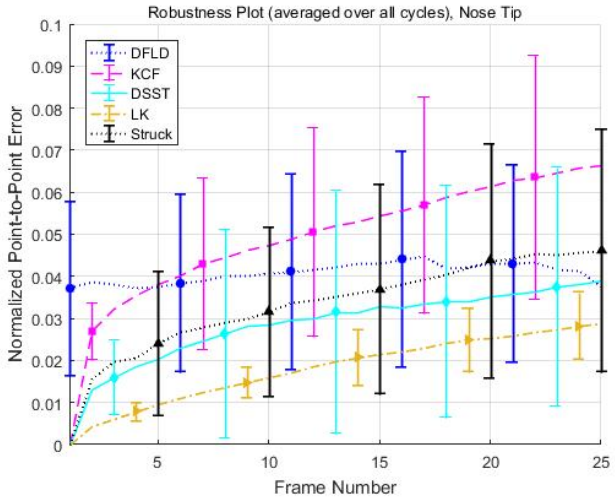


(c) CED curve of the mouth corners

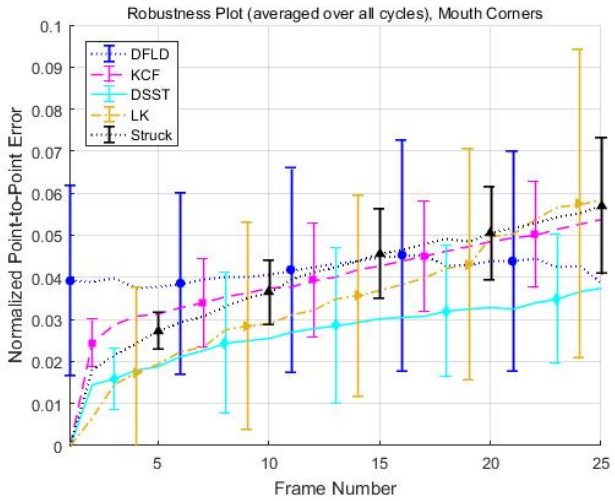
Figure 11: Cumulative Error Distribution curves of the category two videos



(a) Robustness plot of the outer eye corners, category one videos

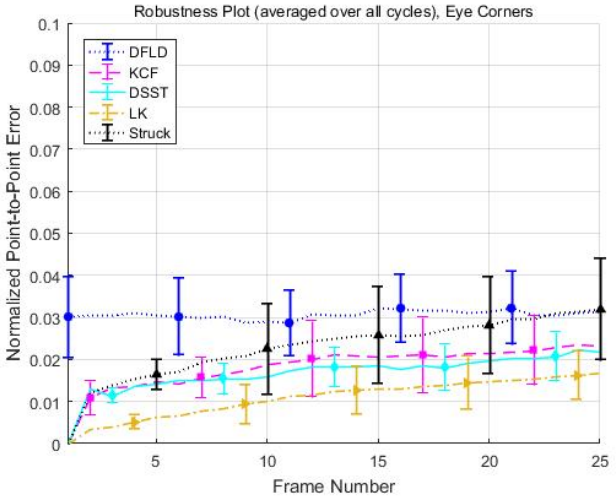


(b) Robustness plot of the nose tip, category one videos

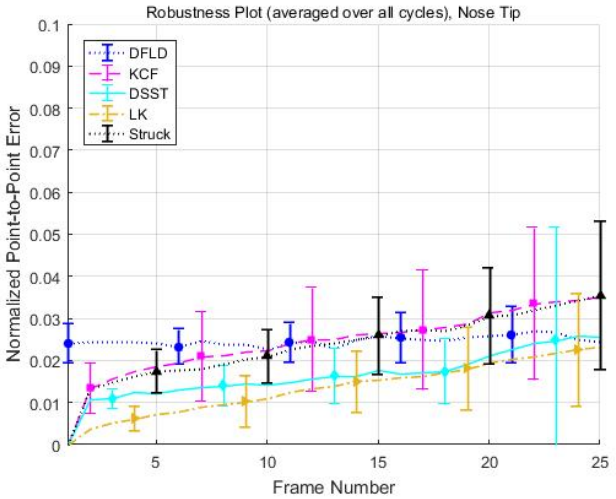


(c) Robustness plot of the mouth corners, category one videos

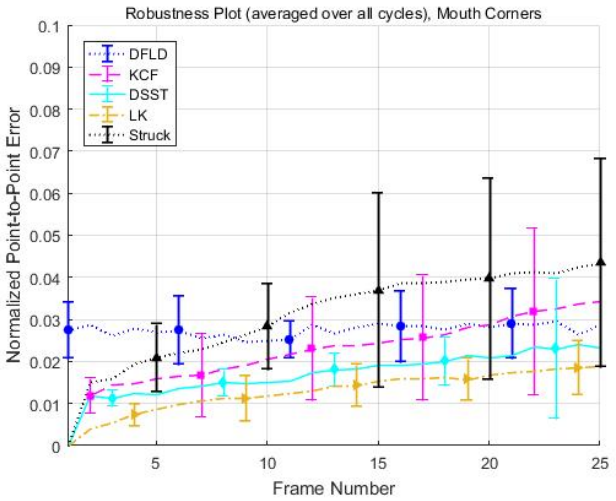
Figure 12: Average normalised error plots of the 3 different landmarks in the category one videos. For readability the average of the 500 frames is taken with intervals of 25 frames (one detection cycle). The standard deviation is indicated by the error bars. Also for readability reasons, the error bar of only one algorithm is shown at each data point. At each data point, the same symbols as in figure 10 and 11 are used to indicate the different algorithms.



(a) Robustness plot of the outer eye corners, category two videos



(b) Robustness plot of the nose tip, category two videos



(c) Robustness plot of the mouth corners, category two videos

Figure 13: Average normalised error plots of the 3 different landmarks in the category two videos. For readability the average of the 500 frames is taken with intervals of 25 frames (one detection cycle). The standard deviation is indicated by the error bars. Also for readability reasons, the error bar of only one algorithm is shown at each data point. At each data point, the same symbols as in figure 10 and 11 are used to indicate the different algorithms.

A. Recommendations

For future work, it is recommended to perform a fair comparison between the frame rates of the different tracking algorithms. The algorithms have to be tested in the same conditions in order to compare their speed.

Furthermore, the CED curves presented in this paper show the accuracy and precision of the video tracking algorithms using an ideal detector with a detection rate of 25 frames/detection. When a non-ideal detector is used, the CED curves will shift to the right. Therefore, the tracking algorithms need to be tested using a non-ideal detector.

Finally, if a non-ideal landmark detector is used, the detector will not have negligible operating time. The framework as is shown in figure 3 cannot be used anymore. This is due to the fact that a detector is slower than a tracker. If the structure as is shown in figure 3 is used, the output frame rate is still limited by the landmark detector. When the output frame rate is set to the frame rate of the tracker, the detector will always initialize the tracker to coordinates of n frames ago. Where n is the number of frames it takes to detect the landmarks. Therefore, a solution to the interaction between landmark detector and landmark tracker has to be found when a non-ideal landmark detector is used.

REFERENCES

[1] Oya Çeliktutan, Sezer Ulukaya, and Bülent Sankur. "A comparative study of face landmarking techniques". In: *EURASIP Journal on Image and Video Processing* 2013.1 (2013), p. 13.

[2] David Cristinacce, Tim Cootes, and Ian Scott. "A multi-stage approach to facial feature detection". In: *In British Machine Vision Conference*. 2004, pp. 231–240.

[3] S. Arca, P. Campadelli, and R. Lanzarotti. "A face recognition system based on automatically determined facial fiducial points". In: *Pattern Recognition* 39.3 (Mar. 2006), pp. 432–443.

[4] J. Shen et al. "The First Facial Landmark Tracking in-the-Wild Challenge: Benchmark and Results". In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. Dec. 2015, pp. 1003–1011.

[5] Y. Wu, J. Lim, and M. H. Yang. "Object Tracking Benchmark". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (Sept. 2015), pp. 1834–1848.

[6] Stephen Milborrow and Fred Nicolls. "Active Shape Models with SIFT Descriptors and MARS". In: *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications, Volume 2*. 2014, pp. 380–387.

[7] S. Phimoltares, C. Lursinsap, and K. Chamnongthai. "Face detection and facial feature localization without considering the appearance of image context". In: *Image and Vision Computing* 25.5 (2007), pp. 741 –753.

[8] T.F. Cootes et al. "Active Shape Models-Their Training and Application". In: *Computer Vision and Image Understanding* 61.1 (1995), pp. 38 –59.

[9] T. F. Cootes, G. J. Edwards, and C. J. Taylor. "Active appearance models". In: *Computer Vision — ECCV'98: 5th European Conference on Computer Vision Freiburg, Germany, June 2–6, 1998 Proceedings, Volume II*. 1998, pp. 484–498.

[10] G. S. Chrysos et al. "Offline Deformable Face Tracking in Arbitrary Videos". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. Dec. 2015, pp. 1–9.

[11] Vahid Kazemi and Josephine Sullivan. "One Millisecond Face Alignment with an Ensemble of Regression Trees". In: *CVPR*. 2014, pp. 1867–1874.

[12] Davis E. King. "Dlib-ml: A Machine Learning Toolkit". In: *Journal of Machine Learning Research* 10 (2009), pp. 1755–1758.

[13] Vuong Le et al. "Interactive Facial Feature Localization". In: *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*. ECCV'12. 2012, pp. 679–692.

[14] *DEST v.08 software library*. https://github.com/cheind/dest/releases (visited on 15-12-16). Feb. 2016.

[15] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'81. 1981, pp. 674–679.

[16] Jean-Yves Bouguet. "Pyramidal implementation of the Lucas Kanade feature tracker". In: *Intel Corporation, Microprocessor Research Labs* (2000), p. 9.

[17] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. "Real-time tracking of non-rigid objects using mean shift". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*. Vol. 2. IEEE. 2000, pp. 142–149.

[18] K. Fukunaga and L. Hostetler. "The estimation of the gradient of a density function, with applications in pattern recognition". In: *IEEE Transactions on Information Theory* 21.1 (Jan. 1975), pp. 32–40.

[19] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. "Kernel-Based Object Tracking". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 25.5 (May 2003), pp. 564–575.

[20] S. Hare et al. "Struck: Structured Output Tracking with Kernels". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.10 (Oct. 2016), pp. 2096–2109.

[21] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Tracking-Learning-Detection". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.7 (July 2012), pp. 1409–1422.

[22] João F. Henriques et al. "Exploiting the Circulant Structure of Tracking-by-detection with Kernels". In: *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*. 2012, pp. 702–715.

[23] João F. Henriques et al. "High-Speed Tracking with Kernelized Correlation Filters". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.3 (2015), pp. 583–596.

[24] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82 (1960), pp. 35–45.

[25] Michael Isard and Andrew Blake. "CONDENSATION—Conditional Density Propagation for Visual Tracking". In: *International Journal of Computer Vision* 29.1 (1998), pp. 5–28.

[26] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007, pp. 787–846. ISBN: 049508252X.

[27] G. Tzimiropoulos. "Project-Out Cascaded Regression With an Application to Face Alignment". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2015, pp. 3659–3667.

[28] FGNet. *Talking Face database*. http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html (visited on 11-11-16).

[29] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[30] Martin Danelljan et al. "Accurate Scale Estimation for Robust Visual Tracking". In: *Proceedings of the British Machine Vision Conference 2014 :* 2014, p. 11.

[31] Matej Kristan et al. "The Visual Object Tracking VOT2014 Challenge Results". In: *Computer Vision - ECCV 2014 Workshops: Zurich, Switzerland, September*

6-7 and 12, 2014, Proceedings, Part II. 2015, pp. 191–217.

[32] David S. Bolme et al. "Visual object tracking using adaptive correlation filters." In: *CVPR*. IEEE Computer Society, 2010, pp. 2544–2550.

[33] Martin Danelljan et al. "Adaptive Color Attributes for Real-Time Visual Tracking". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*. June 2014, pp. 1090–1097.

[34] S. Hare. *Struck C++ software*. https://github.com/samhare/struck (visited on 11-11-16).

[35] Anthony Tanbakuchi. *Comparison of OpenCV Interpolation Algorithms*. http://tanbakuchi.com/posts/comparison-of-openv-interpolation-algorithms/ (visited on 03-12-16). Feb. 2016.

[36] Marco A. Hudelist, Claudiu Cobârzan, and Klaus Schoeffmann. "OpenCV Performance Measurements on Mobile Devices". In: *Proceedings of International Conference on Multimedia Retrieval*. ICMR '14. 2014, pp. 479–482.

APPENDIX A

In this appendix the steps of the window size experiment are shown with intermediate results. This is done for clarification reasons. For each downsampling step the CED curves for different ROIs are plotted. In figure 14 the CED curves are plotted for the outer eye corners in a certain video. The first down sampled instance is used. The resolution is therefore 80% of the original image resolution in each coordinate direction.

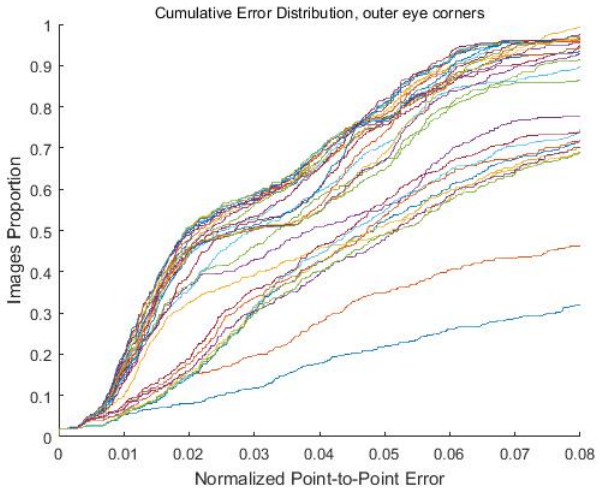


Figure 14: CED curves of the outer eye corners for different window sizes. This is the CED curve of the the first down sampled video instance of a certain video from category one.

The plot in figure 14 is unreadable and therefore the AUC of the different CED curves is plotted against the different ROIs, this is shown in figure 15.

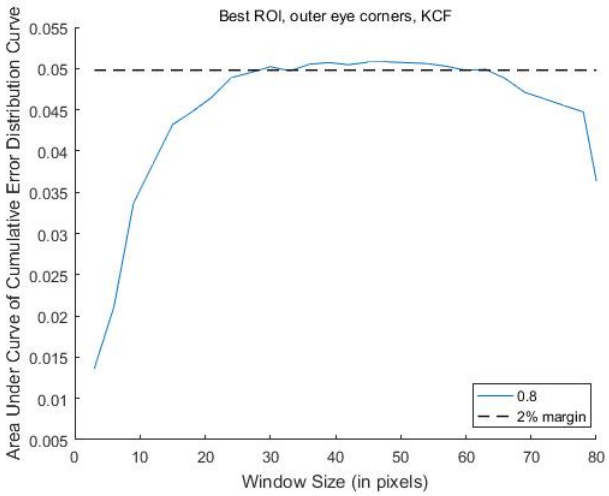


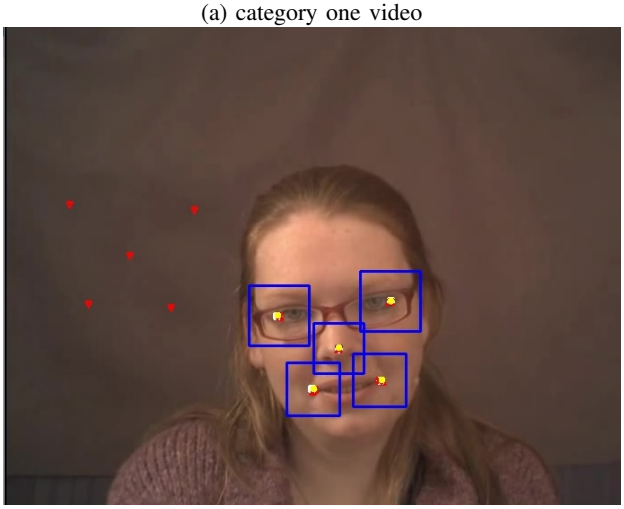
Figure 15: AUC plotted as function of different window sizes.

The points at the intersections between the 2% margin (dotted line) and the AUC curve, as function of the window size (solid line), are used to find the relation between the ROI and IOD. The steps described in this appendix are performed for all 5 instances of the same video with different resolutions. This gives the relation between ROI and IOD for one specific landmark and one specific algorithm. The relation that is obtained is shown in section V in figure 7. The steps described in this appendix are performed for all the different landmarks and all the different algorithms for 3 different videos. The obtained relation between ROI and IOD is assumed to be the same for the other videos in the composed database.

APPENDIX B
DEST FACE DETECTION FAILURES

In figure 16, two examples of the face detection failure of DEST are shown for a category one video and a category two video.

(a) category one video



(b) category two video

Figure 16: DEST face detection failures