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

Pihel Saatmann

Hand-tracking in video conversations

Bachelor's Thesis (6 EAP)

Supervisor: Päivi Kristiina Jokinen

Author: .... " ....." May 2013

Supervisor: .... " ....." May 2013

Allowed to defence

Professor: .... " ....." May 2013

TARTU 2013

Table of Contents

Introduction....................3
1 Object tracking....................4
1.1 Object detection....................4
1.2 Tracking methods....................5
1.3 The Viola-Jones framework....................6
2 CAMShift....................8
2.1 The mean-shift algorithm....................9
2.2 CAMShift for tracking colored objects....................10
2.3 Solutions to object tracking problems....................11
2.4 Limitations....................12
3 Implementation....................13
3.1 Technologies used....................13
3.2 Plugin description....................14
3.2.1 Creating the template histogram....................14
3.2.2 Image processing and tracking....................15
3.2.3 Detecting movement....................16
3.3 ANVIL interface....................16
4 Evaluation....................18
4.1 Tracking bare skin and colored objects....................18
4.2 Tracking with different video encodings....................19
5 Discussion and future work....................20
Summary....................21
Käeliigutuste tuvastamine ja jälgimine videosalvestistes....................22
References....................23
Appendix 1 – Starting the plugin....................25
Appendix 2 (on CD)

Introduction

The goal of this thesis is to describe various methods for tracking objects in video sequences and to implement one of the methods for automatic gesture annotation. The implementation has been made as a plugin for the ANVIL annotation tool [1] and can be used for recognizing and annotating hand gestures in recorded video conversations. Hand gesture recognition consists of several elements such as hand detection and tracking and gesture detection. The implementation can track and detect gestures, however initial detection of the hand inside the video frame is left to the user. The plugin has been tested on real dialogue data recorded as part of the Estonian Science Foundation's project MINT (Multimodal INTeraction, ETF8958). The aim of the MINT project is to study signals of multimodal communication (hand gestures, head movements etc) and their relation to speech [2]. Annotation of video data is an important prerequisite for human communication studies, but doing this manually is time and resource consuming. It is thus important to study automatic tools for annotation.

Section 1 gives a general overview of object detection and tracking and describe an object detection method known as the Viola-Jones framework. Section 2 describes the CAMShift algorithm in detail. Section 3 gives an overview of the implementation and Section 4 discusses evaluation on the MINT data. Section 5 outlines the problems encountered during the tracking process and some possible solutions and future improvements. The source code for the implementation is included on a CD.

1 Object tracking

Object tracking is a sub-domain of computer vision that deals with locating the position of an object in each consecutive frame of a video. A tracking algorithm usually consists of detecting the object, tracking it and estimating the trajectory. Additionally information about the object's properties such as orientation and shape can also be provided.[3]

Efficient algorithms have to overcome problems such as image noise, poor or changing lighting, complex object shapes, irregular object motion, occlusion and distractors present in the video and be able to run in real-time. Tracking may be simplified by setting constraints on the motion or appearance of the object (for example only looking at objects of a certain color or shape or assuming the motion is smooth).[4]

1.1 Object detection

An object can be represented by it's shape and appearance. Some examples of shape representation include a set points or single point, primitive geometric shapes and contours that mark the edge separating the object from the rest of the image. Appearance can be represented by probability densities, appearance models or templates that are formed using shapes of silhouettes. The advantage of templates and models is that they also contain information about the shape. Representations are usually chosen according to the object domain.[3]

A relevant piece of information (shape or appearance representation) about an object is called a feature. In order for a detector to be successful a suitable feature or features need to be selected for analysis. The best kind of visual features are ones that enable objects to be more easily distinguished. Mostly features are selected manually depending on the object domain, but automatic feature selection by using supervised learning is also possible. An example of using supervised learning is the Viola-Jones object detection framework that is described in section 2.3.[3]

Object detection is usually based on information from a single frame. In order to reduce the number of false detections, image differencing information which highlights changing regions in a sequence of frames can also be used. The method for detection depends on the set of used features. An example of detection methods is image segmentation that partitions the image into perceptually similar regions in order to find object contours.[3]

1.2 Tracking methods

Detecting an object and matching it's instances across frames can be done separately or jointly. In the first case, detection methods are applied to find possible objects in each frame and then a tracking method tries to match the instances across frames. In the latter case, the location estimation and instance matching are done jointly by iteratively updating the object's location and region information based on previous information. Both of these cases use shape and appearance models to represent objects. Tracking methods can be divided into three categories – point, kernel and silhouette tracking. A detailed taxonomy of tracking methods can be seen in Figure 1.[3]

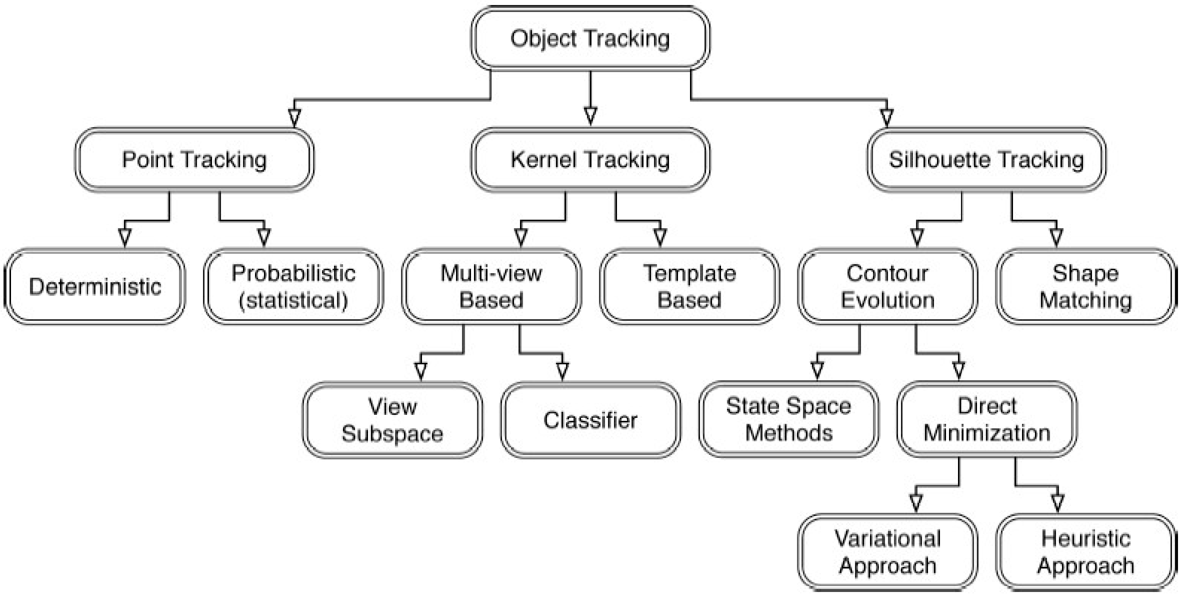


Figure 1 - Taxonomy of object tracking methods. Taken from [3]

Point tracking requires a detection method to find objects in each frame. Objects are represented by sets of points and the tracker attempts to match the features in consecutive frames based on the previous object state (position, motion etc).[4]

Kernel tracking is performed by computing the motion of an object's primitive shape or appearance features in consecutive frames. An example of kernel tracking is template matching which finds the region of interest in a frame and searches the next frame for a match. The CAMShift algorithm described in section 3 uses this method.[4]

Silhouette tracking uses object models such as color histograms or object contours that are calculated in each frame to the find the object region in the next frame. This method can be used for objects that are too complex to be represented by points of primitive shapes.[4]

1.3 The Viola-Jones framework

The Viola-Jones framework is a feature-based method that was published in 2001 by Paul Viola and Michael J. Jones. It was primarily meant as a face detection method although it can also be used to detect other types of objects.[5]

The framework classifies objects based on rectangle features (Haar-like features), which are represented by the sums of pixel values in the rectangular areas. The value of each feature is the sum of pixel values within the white rectangles subtracted from the sum of pixel values in the dark rectangles. The framework uses an image representation called the *integral image.* At a given location the integral image represents the sum of all pixel values above and to the left of the given location. This allows for any feature at any scale and location be evaluated with a few operations. If the difference between the dark and light regions is above a predetermined threshold then a feature is detected. A feature used for face-detection could for example compare the intensity of the eye regions to the

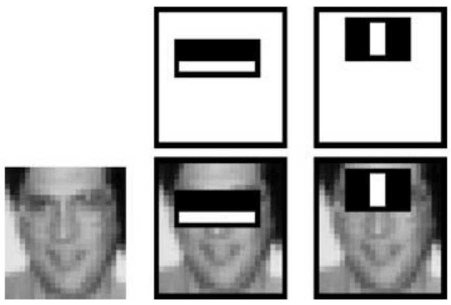


Figure 2 - Viola-Jones feature examples. Taken from [5]

intensity across nose region in a frontal face image (rightmost column in Figure 2).[5]

If all possible combinations of position, scale and feature type in a particular sub-window of an image would be considered then the amount of features to be evaluated each time would be very large. Therefore in order to reduce computing time the framework uses a a cascade of simpler (or weaker) classifiers to form a strong classifier. The detection process works on the principle that if the first classifier gives a positive result then the next one is applied and so forth. A negative result will reject the sub-window. Features are chosen and classifiers are trained by a modified version of a machine learning algorithm called AdaBoost. Each next classifier in the cascade is trained using the examples which pass through the previous classifiers. In this way the majority of the image sub-windows would be rejected in the earlier stages so the more complex classifiers would not have to be applied.[5]

In case of object tracking the detector would be applied to each sequential video frame to attempt to locate the object. The framework would have to be retrained on a selected data-set for it to work on different object types (for example a classifier to be trained on frontal face images would not work for profiled faces).

2 CAMShift

The CAMShift (Continuously Adaptive Mean-Shift) algorithm was developed by computer vision researcher Gary R. Bradski in 1998. It was intended to be used as a face-tracking algorithm for a computer vision interface for controlling video games and exploring 3D virtual environments. In this chapter I will review the algorithm. The presentation is based on the references [6], [7] and [8].

Since the CAMShift tracker was intended to be used in real time and as part of a larger user interface, it needed to be fast and computationally efficient. Complex methods such as feature matching were not efficient enough nor necessary for the kind of basic object tracking the algorithm was meant for. Therefore a color-based tracking method was chosen.

Steps of CAMShift (Figure 3):

1. The image information is converted into a probability distribution.
2. An initial search window size and location for mean-shifting are chosen.
3. The mean-shift algorithm is applied to the distribution data.
4. The next window size is calculated based on the distribution inside the search window.
5. Steps 3 and 4 are repeated until the window doesn't move anymore or moves less than a predefined threshold.

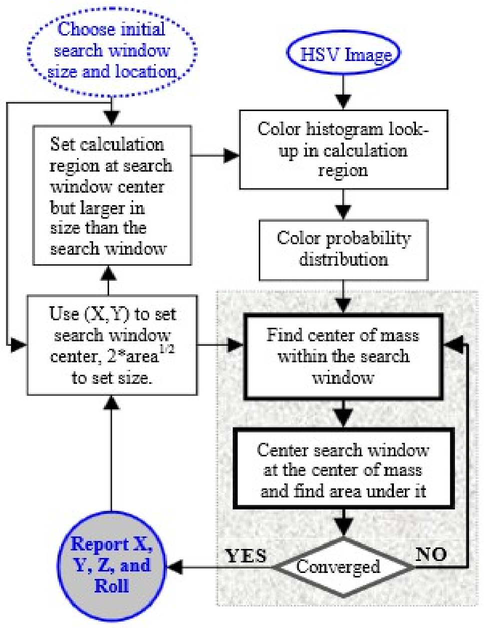

Choose initial search window size and location
HSV Image
Set calculation region at search window center but larger in size than the search window
Color histogram look-up in calculation region
Color probability distribution
Use (X,Y) to set search window center, 2*area$^{1/2}$ to set size.
Find center of mass within the search window
Center search window at the center of mass and find area under it
Report X, Y, Z, and Roll
YES
Converged
NO

Figure 3 - CAMShift for colored object tracking. Taken from [6]

For video sequences the algorithm is applied to each frame to find the new location of the tracked object.

2.1 The mean-shift algorithm

CAMShift uses the mean-shift algorithm which is a „robust non-parametric technique for climbing density gradients to find the mode (peak) of the probability distribution“ [6]. This means that if given a set of data-points the algorithm associates each point with a nearby maxima of the underlying probability density function and iterates a predefined search window over the data. After each iteration the search window moves to a more dense region until it is centered over the maxima. The mean-shift algorithm is represented by the gray box in Figure 2.

How mean-shift is calculated:

1. Search window size and location are selected.
2. The mean of the data inside the search window is calculated.
3. The window is centered at the location of the mean value.
4. Steps 3 and 4 are repeated until the window doesn't move anymore or moves less than a predefined threshold.

The robustness of the algorithm means that it ignores outliers in the data - in case of mean-shift data-points far away from the search window are ignored [8]. This helps eliminate distractors in a video sequence that are not in direct contact with the tracked object (for example faces in case of tracking a hand).

The mean-shift algorithm can be used for finding the location of the object in static images, however in a video recording objects can move away or towards the camera or rotate so they change in size. In those cases the size of the search window would also have to be changed so using mean-shift would fail. As a solution the CAMShift algorithm has added a dynamic search window that adjusts itself to the changes in object size.

2.2 CAMShift for tracking colored objects

CAMShift uses the HSV (Hue Saturation Value) color system (Figure 4), which separates the color (hue) of a pixel, concentration of the color (saturation) and brightness (value) into three channels. The values taken from the hue channel are stored in a 1D histogram, which is later used as a color model to convert the pixels in a video frame to a color probability distribution.

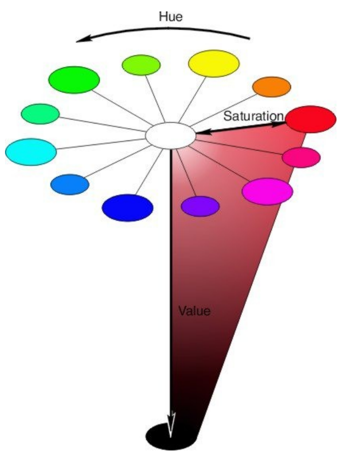


Figure 4 - HSV color model. Taken from [9]

A histogram is a table of possible hue values. Each entry in the histogram represents how many pixels in an image have that specific hue value. If the histogram is normalized so that the sum of the entries equals 1 then each entry represents the percentage of pixels of the given value in an image. In other words a normalized histogram can be used to represent a probability distribution of the color data.[8, p.90]

If we want to track an object, we create the histogram of all the pixels representing the given object and normalize it. Then for the next video frame the probability of each pixel belonging to the tracked object is calculated (the probability distribution of the image). For example if 40% of the pixels in the tracked object have a specific hue value then all the pixels in the next video frame with the same hue value have a 40% probability of belonging to the tracked object.

The probability distribution is then given to the CAMShift algorithm, which iterates the search window over the image and finds the location where the probability inside the window is maximized. To make tracking more efficient the color probability distribution doesn't have to be calculated for the entire image, but can be restricted to a region surrounding the current search window.

The quality of the tracking depends on the quality of the probability distribution. A problem with using the HSV color system is that if the brightness value is very low then saturation is also low and hue can not accurately represent the color differences between pixels. This also applies to pixels that have very high brightness (white pixels) or just very low saturation (neutral gray pixels). These pixels contribute to noise and therefore have to be filtered out by saturation and brightness thresholds.

2.3 Solutions to object tracking problems

Most of image noise is filtered out by using the HSV color model and by correctly applying thresholds. CAMShift has been shown to track the object's X and Y position and angle quite well in up to 30% of white noise [6]. Use of the HSV color model and thresholds also gives the tracker a wide lightning tolerance.

The robustness of the algorithm allows it to ignore distractors that are outside of the search window. Distractors inside the search window, but not in direct contact with the tracked object tend to be ignored as well. Occlusion is handled by the algorithms tendency to move to the nearest dominant peak of the distribution. This means that if a distractor passes in front of the tracked object the tracker should stay on the tracked object as long as it's not completely occluded or the probability distribution of the distractor isn't “more suitable”. In case of partial occlusion the tracker tends to “stick to the mode of the color distribution that remains” [6].

Objects changing in size as they change position and rotation during a video sequence is not a problem since CAMShift's search window adjusts its size to the object's color probability distribution area. This also solves the problem of irregular object motion as objects closer to the camera appear to move faster than those further away.

CAMShift has the tendency to grow its search window to contain the connected probable pixels of a tracked area. This means that the search window doesn't stick to just a small part of the object, but grows to track the entire object (in case of tracking a hand if the search window is initially set on the fingers, it moves to encompass the entire hand). Thus if the initial selected part of the object is occluded, the algorithm is still able to track the remainder.

2.4 Limitations

CAMShift tends to fail as a tracker if the image has a large amount of probable pixels (if the background has the same hue as the object). Too bright, dim or colored lightning will also cause tracking errors.

If the tracked object is lost or the tracker moves to another object, some other method of object recognition has to be used or the search window needs to be manually relocated.

The search window's tendency to encompass the connected probable pixels can also be a limitation, since it doesn't allow more specified tracking (like tracking only part of the face).

3 Implementation

CAMShift was chosen for the implementation, because the main purpose of the plugin is to detect movement and knowing the precise hand position in the 3D-space is not required. Color-based tracking allows to estimate the approximate position of the object with enough precision to detect movement between frames. Unlike the Viola-Jones tracker CAMShift does not require prior training or a database of features to work and could also be used to track various other colored objects besides hands.

Currently ANVIL has a face-tracking plugin that creates automatic annotations for head movements. It uses the face recognition algorithms implemented in OpenCV that are an improved version of the Viola-Jones face detector. The tracker can also be used for other body parts, however the available settings do not include hands.[10]

3.1 Technologies used

The plugin is uses Java version 7.

OpenCV (Open Source Computer Vision) is a C and C++ library of algorithms for computer vision, video analysis and image manipulation. It has the implementations of many object recognition and tracking algorithms (including CAMShift).[11] The plugin uses version 2.4.3.

JavaCV provides wrappers to the OpenCV and other libraries used in computer vision. It wraps C API wherever possible, and C++ API when necessary. It also includes helper classes and methods on top of OpenCV to facilitate its integration to the Java platform.[12] The plugin uses version 0.3.

ANVIL is a free video annotation tool that allows the user to create multi-layered color-coded annotations. ANVIL also features several tools that allow the annotators to analyze and manage the annotations, for example calculate the intercoder agreement (kappa score) and association tables for annotation categories. The annotations can be written out in XML-format and so be easily used for further statistical analysis.[1] ANVIL is available at http://www.anvil-software.org/.

3.2 Plugin description

The tracker's work can be divided into three segments – creating a histogram to be used as the object template, converting the frames to a appropriate representation and applying CAMShift, detecting object movement and writing movement data to an output.

The OpenCV library contains an implementation of the CAMShift algorithm. Before an image can be used as input for the CAMShift algorithm it has to be converted into a suitable format that represents it's color probability distribution (a probability map). OpenCV also provides functions for image processing and histogram creation.

OpenCV methods use a special image structure called the IplImage, which has multiple formats for representing different pixel depths, number of color channels and other image parameters [13]. For using CAMShift all video frames have to converted to the HSV color space before information is extracted from them.

3.2.1 Creating the template histogram

Before tracking can start, the program needs to create a hue histogram of the tracked object to be used later as template for color probability calculations. This is done simply by having the user mark a rectangular region that contains the object and then calculating the histogram for the marked region. The marked region can be smaller than the object as CAMShift will automatically expand the window to the largest possible colored area that matches the color information of the marked region. If the region is bigger than the object then color information from the background objects may impair tracking.

The image has to be converted to the HSV color space and split into three separate channels for hue, saturation and value. The histogram is one dimensional and has 256 bins (entries), OpenCV uses hue values ranging from 0 to 180. It is computed from the hue channel of the image. If a saturation threshold is set then pixels with low saturation are ignored by applying a saturation mask on hue channel, which removes all the pixels with a value below the threshold. The histogram is then normalized so that the values range from 0 to 1 to make probability calculations simpler.[9]

3.2.2 Image processing and tracking

After the template histogram is created, the program will take the next frame from the video and process it. For each frame the tracker has to:

1. Convert the image to HSV color space.
2. Calculate probability map for the image.
3. Eliminate pixels with low saturation from the map.
4. Apply CAMShift to the probability map.
5. See if the object has moved a distance that is bigger than threshold.

After converting the image to HSV it is separated into three channels. The saturation channel is used to create a saturation mask for the image which is later used to eliminate pixels that have a value below the saturation threshold.

Then the probability map is created by applying a technique called histogram back-projection, which uses the the hue histogram created at the beginning of tracking. This creates an image where regions with highest probability are marked with white color and regions with lowest probability with black, grayish pixels have a probability somewhere between those. The saturation mask is applied to the back-projection image to filter out pixels with low saturation (Figure 5).[9]

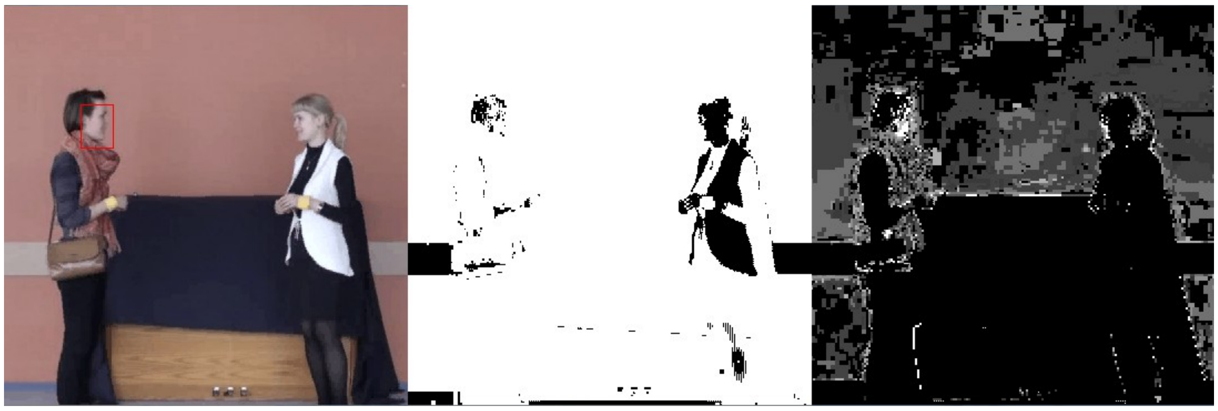


Figure 5 - normal view, saturation mask and back-projection for skin hue.

OpenCV's implementation of CAMShift takes the back-projection, location of the search window in the previous frame and search termination criteria as input and outputs the window's new location. It also returns the number of iterations it took for the tracker to find the window location. The implementation has two termination criteria – the maximum number of iterations and the window movement distance below which the window is considered to be centered on the new location.[13]

3.2.3 Detecting movement

The search window's location is compared to it's location in the previous frame. If the distance between the window center points is above a predetermined threshold then movement is detected. The start of the movement is marked by the first following frame that has a distance value above the threshold and the end of the movement by the next frame that has a distance value below the threshold.

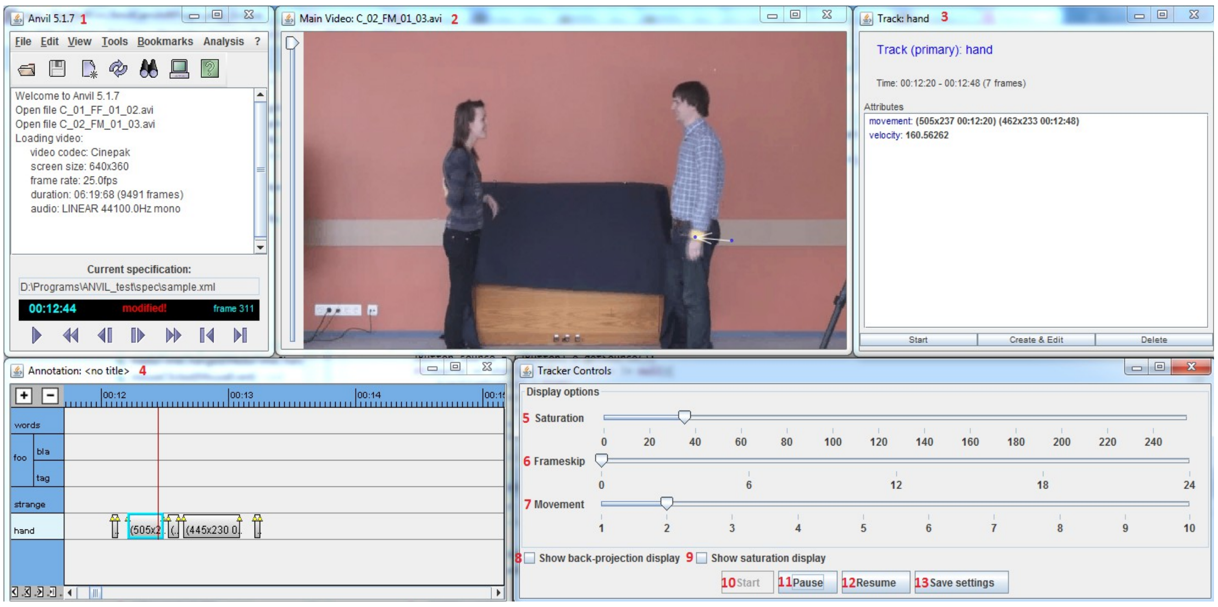
The start and end positions and total movement distance and duration are saved and written to the selected annotation track as a new interval. A vector connecting the start and end points of the movement is displayed on the main video window during each interval.

3.3 ANVIL interface

Figure 6 shows the user interface for ANVIL and the controls for the hand-tracker. The user can specify multiple settings to increase the efficiency of the tracker.

1 – Main ANVIL window.
2 – Main video window.
3 – Annotation interval info.
4 – Annotation tracks.

Tracker controls
5 – Minimum saturation threshold (see section 3.2.2).
6 – The number of frames to skip on each iteration.
7 – Movement threshold in pixels (see section 3.2.2).
8 – Toggles the back-projection display.
9 – Toggles the saturation display.
10 – Starts the tracking.
11 – Pauses the tracking.
12 – Resumes the tracking.
13 – Saves tracker settings to the selected track.



Welcome to Anvil 5.1.7
Open file C_01_FF_01_02.avi
Open file C_02_FM_01_03.avi
Loading video:
    video codec: Cinepak
    screen size: 640x360
    frame rate: 25.0fps
    duration: 06:19:68 (9491 frames)
    audio: LINEAR 44100.0Hz mono

Current specification:
D:\Programs\ANVIL_test\spec\sample.xml
00:12:44 modified! frame 311

Track (primary): hand
Time: 00:12:20 - 00:12:48 (7 frames)
Attributes
movement: (505x237 00:12:20) (462x233 00:12:48)
velocity: 160.56262

Figure 6 - ANVIL interface and tracker controls

The tracking cannot start until a template histogram is obtained so if the user hasn't marked an object region in the video and presses the 'Start' button, they are prompted to do so. The user is also allowed to re-mark the object region both before and after tracking has started.

The default value of the saturation threshold is set to 65. Higher values of the saturation threshold help to filter out noise, but may also cause the pixels the user wants to track be ignored if they have low saturation. The saturation and back-projection displays are meant to help the user decide on an appropriate saturation threshold and also to see if the background or other moving objects may be interfering with the tracking.

The default number of frames to skip is 0, which means that the tracking algorithm will be applied to every frame in the video. Increasing this value will lower the computation cost, but also increase the probability that the search window is lost or jumps to another object if the tracked object moves too much between frames.

4 Evaluation

The tracker was tested on videos recorded during the MINT project, the recorded participants have agreed on the use of the videos for research purposes and being shown to third parties. In each video two people standing face to face are filmed from the side so that their full body apart from the feet is in the frame. The raw videos recorded for the project were converted to two different formats (Xvid and Cinepak) and resized to 640x320 pixels. A stand-alone version of the tracker was tested on both of these formats since ANVIL does not support Xvid. Videos with Cinepak encoding are of a lower quality than Xvid so the tracking window tended to be lost more frequently and tracking was not as precise. Precise results of using the tracker on a 40-second sequence one of the test files are provided in section 4.2.

4.1 Tracking bare skin and colored objects

In some videos the recorded people have yellow bracelets around their hands, the tracker was tested on both these bracelets and on bare hands. When tracking the yellow bracelets the tracker was working as expected without any errors. The tracking window didn't move to the wrong object was lost only if the bracelet was completely occluded. Movement was detected relatively accurately.

Tracking bare hands proved to be more problematic. The tracker would jump to other hands or faces in the frame if the regions overlapped (for example if the people shook hands) or get lost if the hand was completely occluded (for example if the person put their hands in their pockets. Since the recording was made from a side angle then the hand further away from the camera would often interfere with tracking the other hand. Movement was not accurately detected and often small movement was detected when there actually was none.

Figure 7 shows comparison of the two tracking conditions. On the left is the color probability image based on the bracelet and on the right based on the bare skin area of the hand. Tracking the bracelets proved more accurate, because the bracelets have a hue that is easily distinguishable from the background. In most cases it was the only object in the video of that particular color. Looking at the probability images also shows that edges belonging to the background have a somewhat similar hue to the skin hue. This causes the tracker to sometimes get stuck to a part of the background if the hand becomes occluded. Things are made even worse if the person's clothes also have a similar hue. In those cases the tracker usually moved to the body region. Since the two bracelets in the video very rarely came into direct contact, the search window usually stayed on the correct object.

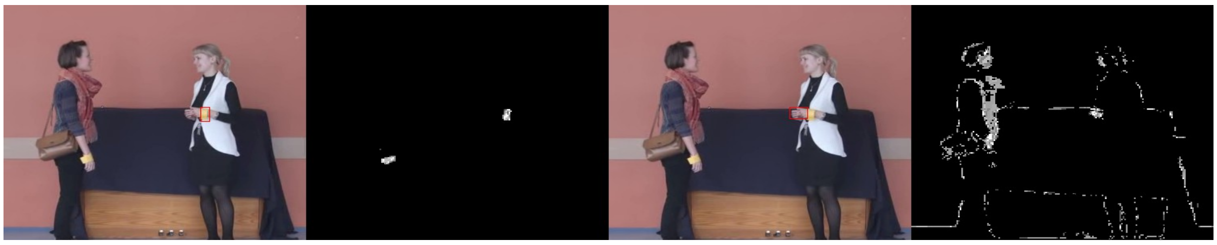


Figure 7 - Normal and probability map views for bracelet and bare skin tracking

4.2 Tracking with different video encodings

Table 1 shows the result of testing the tracker on a video sequence that had the Xvid encoding and a duration of 40 seconds (0:10 – 1:20) (file 'C_01_FF_01_02_Xvid.avi', see Appendix 2). The search window was set to the left person's bracelet. During the tracker's work the search window was never lost.

Threshold	**Movements detected**	**Average movement distance (pixels)**	**Average movement velocity (pixels/frame)**
1	208	7,5	3
3	45	18,8	6
5	25	20,7	9

Table 1 - tracking with different movement thresholds with Xvid

Table 2 shows the the results of testing the tracker on the Cinepak encoded version of the video (file 'C_01_FF_01_02_Cinepak.avi', see Appendix 2). With low thresholds the tracker often detected movement when there actually was none. The search window was lost 11 times during tracking (with each threshold) and was not manually reset.

Threshold	**Movements detected**	**Average movement distance (pixels)**	**Average movement velocity (pixels/frame)**
1	209	19,5	3,6
3	272	10,7	5,3
5	143	11	7,4

Table 2 - tracking with different movement thresholds with Cinepak

5 Discussion and future work

One of the biggest problems using the tracker is that it detects movement when there is none. Looking at the back projection images for the video shows that the probability values for static background objects change during the video. For non-static objects the changes are even larger, which makes the tracker move the search window and give the impression of movement. This is probably caused by changes in lighting or the use of the lower-quality video for testing. Increasing the detection threshold would solve this problem, but then actual movements that are small (for example very slow movements) would also not be detected.

As mentioned in the previous section a big problem with tracking bare hands was that other objects in the video had a similar hue. This could be somewhat improved by increasing the saturation threshold. Unfortunately it appears that the skin hue saturation in the videos is also quite low so increasing the threshold would not solve the problem.

The hand regions in the videos used for testing are quite small and not rectangular, which makes it more difficult to select a 'good' region for the hue template. If a very small region is selected then the range of possible skin hue pixel representations is very small so it doesn't accurately represent the entire hand. If the region selected is bigger than the hand then background hue information will make the template inaccurate. A possible solution would be to allow the user to freely select the initial tracking region instead of using a rectangle. Obviously having a larger hand region would also be a solution.

In summary most of the problems could be solved by using full high quality videos. Currently the pixel rate of the high quality raw MINT videos is impractical for the gesture tracking, and so a trade-off is needed between the tracking accuracy and video quality. In the future a hand detection method could be added to the plugin to reduce the chance of losing the search window during tracking and to remove the need for the user to manually locate the hand region. An automatic gesture recognition system could also be based on the current implementation.

Summary

The goal of this thesis was to describe various object tracking methods and to create a tool for automatic gesture annotation. Annotation of video data is an important prerequisite for human communication studies, but doing this manually is time and resource consuming. It is thus important to study automatic tools for annotation.

The tool implements an object tracking algorithm known as CAMShift and is used as a plugin for the ANVIL annotation software. The tool is able to track hands and other colored objects in a video and detect movements, the inital detection of the hand is left to the user. The movements are automatically annotated by writing the start and end point of the movement and average velocity to a specified annotation track in ANVIL.

The tool was tested on recordings of actual dialogues and used to track both bare hands and colored objects. The tracking and movement detection precision depends on the quality of the video being used and on the user specified settings.

All in all the created tool meets the goal of the thesis as it is able to automatically track and annotate gestures in recorded video conversations. However, adding functionality to automatically detect hands in a video frame without user intervention and classify gestures based on collected movement data would further reduce the need for user input during the annotation process.

Käeliigutuste tuvastamine ja jälgimine video-salvestistes

Bakalaureusetöö

Pihel Saatmann

Resümee

Antud lõputöö eesmärgiks oli kirjeldada erinevaid meetodeid objektide jälgimiseks videosalvestistes ning luua tööriist automaatseks käeliigutuste annoteerimiseks. Videosalvestiste annoteerimine on oluline vahend inimestevahelise suhtluse uurimiseks. Käsitsi annotatsioonide tegemine on aeganõudev ning seega on oluline uurida võimalusi automatiseeritud vahendite loomiseks.

Loodud tööriist kasutab CAMShift jälgimisalgoritmi ning on realiseeritud lisamoodulina programmile ANVIL. ANVIL on vabavaraline vahend annotatsioonide loomiseks. Tööriist suudab jälgida käsi ja värvilisi objekte ning tuvastada liigutusi videovestlustes. Algne käepiirkonna või muu objekti videost ülesleidmine ja ära märkimine on jäetud kasutaja hooleks. Liigutused annoteeritakse automaatselt ning info liigutuse alg- ning lõpppunkti ja keskmise kiiruse kohta kirjutatakse ANVIL'i annotatsioonifaili.

Loodud tööriista testiti videosalvestiste peal kahe inimese vahelisest suhtlusest ning kasutati käte ning muude värviliste objektide jälgimiseks. Jälgimise ja liigutuste tuvastamise täpsus sõltus videokvaliteedist ning kasutaja poolt määratud sätetest (näiteks minimaalne arvestatav värviküllastus).

Kokkuvõtteks võib öelda, et loodud tööriist täidab seatud eesmärke, kuid on ruumi täiendusteks. Näiteks võiks lisada võimalused automaatseks käepiirkondade leidmiseks videos ning liigutuste analüüsimiseks ja eri kategooriatesse jagamiseks. Sellised täiendused vähendaksid veel rohkem kasutaja tööd.

References

[1] Kipp, M. (2001) Anvil - A Generic Annotation Tool for Multimodal Dialogue. Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech), pp. 1367-1370.

[2] Jokinen, K. and S. Tenjes (2012). 'Investigating Engagement - intercultural and technological aspects of the collection, analysis, and use of the Estonian Multiparty Conversational video data'. In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), 23-25 May 2012, Istanbul, Turkey

[3] Yilmaz, A., Javed, O., and Shah, M. (2006). 'Object tracking: A survey'. *ACM Computing Surveys (CSUR)*. 38 (4), Article 13. Available at: *http://crcv.ucf.edu/papers/Object%20Tracking.pdf* (Accessed: 9th May 2013)

[4] Han, Bing; Paulson, Christopher; Lu, Taoran; Wu, Dapeng; Li, Jian (2009). 'Tracking of Multiple Objects under Partial Occlusion'. *Automatic target Recognition XIX,* 7335. Available at: *http://www.wu.ece.ufl.edu/mypapers/trackingSPIE09.pdf* (Accessed: 9rd May 2013).

[5] Viola, P., Jones, M. J. (2004). 'Robust Real-Time Face Detection'. *International Journal of Computer Vision*. 57 (2), pp.137–154.

[6] Bradski, G. R. (1998). Computer video face tracking for use in a perceptual user interface. *Intel Technology Journal*. Q2, pp.705-740.

[7] Hewitt, R. (2007) 'Seeing With OpenCV, Part 3: Follow that Face!', *SERVO Magazine.* [Online]. Available at: *http://www.cognotics.com/opencv/servo_2007_series/part_3/index.html* (Accessed: 5th May 2013).

[8] Laganière, R (2011). *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing Ltd, Olton Birmingham, GBR. pp.90-94. [Online] Available at: *http://my.safaribooksonline.com/9781849513241/91* (Accessed: 9th May 2013).

[9] HSV color model. [Online] Available at: *http://www.ncsu.edu/scivis/lessons/colormodels/color_models2.html* (Accessed: 11th May 2013).

[10] Bart Jongejan (2012). Automatic annotation of head velocity and acceleration in Anvil. In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), 23-25 May 2012, Istanbul, Turkey.

[11] OpenCV (Open Source Computer Vision) Wiki. Available at: *http://code.opencv.org/projects/opencv/wiki* (Accessed: 9th May 2013).
[12] Audet, S. (2013). JavaCV. Available at: *http://code.google.com/p/javacv/* (Accessed: 9th May 2013).
[13] Bradski, G.R and Kaehler, A. (2008) *Learning OpenCV.* O'Reilly Media, Sebastopol. pp.42, 341 [Online]. Available at: *http://my.safaribooksonline.com/9780596516130/341* (Accessed: 9th May 2013).

Appendix 1 – Starting the plugin

1. Add the plugin to ANVIL:
   - Edit → Options → Plug-ins → add.
     - The title can be whatever you want to call the plugin.
     - The class has to be AnvilHandTracker.

2. Create a specification:
   - Edit → Edit specification.
   - Either create a new one or load an existing one.
   - Add track/group → Add track.
     - Name can be whatever you want, type should be primary.
   - Add attribute.
     - Name should be "movement", type TimeStamped Point.
   - Add attribute.
     - Name should be "velocity", type String.

Steps 3 and 4 may be done in any order.

3. Start the plugin:
   - Tools → [Name of the plugin]

4. Load a video:
   - File → Open → [Video] → Browse specification → [The specification you created earlier].

5. Select the correct track in the annotation window by clicking on it.

6. Start the tracker by selecting an initial search window in the main video window and clicking Start in the tracker controls.