Real-time Tracking of Participants in Meeting Video

Michal Hradiš[1] and Roman Juránek[2]

Department of Computer Graphics and Multimedia
Brno University of Technology
Brno, Czech Republic

Abstract

The aim of our work was to develop a robust real-time tracker of body parts in meeting videos. We have focused mainly on tracking heads in meeting video data from the AMI project. The AMI project is concerned with the development of meeting browsers and remote meeting assistants for instrumented meeting rooms. Our system is intended to process video data acquired from camera online. This sets bounds in the terms of computational cost of used algorithms. Therefore we decided first to create a fast solution and analyze its results in order to tell if further improvements are needed.

We have divided tracking task into two separate parts. First skin color model, background subtraction and topological feature based object classification is used to detect heads in video. The detected objects are consequently tracked throughout the video sequence by using KLT feature based tracker. Tracking results are stored in the XML format for evaluation.

Keywords: Tracking, Head detection, Skin color model, KLT tracker

1. Introduction

Detecting and tracking objects in video sequences is very wide and up to now not generally solved topic. In order to achieve practical results it is necessary to constrain this problem and use implicit knowledge as much as possible. Our work focuses narrowly on tracking people in meeting video data. This allows us to assume constant illumination, static background and certain typical object behavior in all video sequences.

Human visual tracking is used in many applications—mainly video surveillance and human-computer interaction. In our case tracking results are planned to be used to extract semantic information about a meeting. This involves meeting participant localization, current speaker identification and keeping track of events like voting etc. It could be also used for gesture recognition in order to allow comfortable control over the recording system. The resulting system is supposed to process all recorded data on the fly. This fact demands all used algorithms to be able to work in real-time.

As the first step we decided to concentrate on head tracking, considering the fact that this task is relatively easy compared to tracking other parts of human body. When tracking heads we can make use of a knowledge about its color, shape, size and movement. Our goal was to present simple multiple head tracker, evaluate its performance and suggest further improvements if needed.

1 *xhradi05@stud.fit.vutbr.cz*
2 *xjuran07@stud.fit.vutbr.cz*

2. Related work

There has been much effort to solve human tracking in video sequences in the last years. Many different methods were created to detect faces in images based on skin color model [1], neural networks, image classifiers, template matching, Gabor wavelet networks etc. There are also many methods for tracking objects in video sequences e.g. particle filters, Kalman filter, condensation algorithm and optical flow estimation [2]. Here are some projects about body parts tracking in videos.

The HandVu is originally a project of University of California. It is targeted to track hands in real time and to control a system by hand gestures. It is based on the work of Kölsch [3] which extends the KLT tracker with feature flocking and color cue behavior.

In [4] an extended KLT tracker of heads is described that uses stochastic skin color model with elipse fitting to init the tracker. The result system achieved real time performance and was able to automatically recover when the face was lost.

The AMI project in which this work takes part is more complex. It is focused on development of meeting browsers and remote meeting assistants for instrumented meeting rooms. Image processing part of the project focuses on automatic annotation of meeting sessions. This involves mainly speaker detection and tracking of participants in order to extract semantic information about session progress. Meeting rooms use either scenario with several video cameras or with one omni directional camera. Within the scope of the AMI project many different approaches are explored to search for body parts and to track them e.g. particle filters [5] or neural network based face detectors [1].

3. Overview

In the present it is very difficult or even impossible to reliably detect faces or heads in all positions and situations that can arise during meeting sessions. On the other hand it is possible to track almost arbitrary objects even in difficult conditions in real-time. Therefore we decided to combine relatively simple head detection with an object tracker and observe the resulting performance. This scheme allows us also to exchange either the detector or the tracker for a different one if necessary.

Our head detector is based on background subtraction and skin color segmentation. We could use these methods due to constrained meeting environment with constant illumination, static camera and background.

We based our tracker mainly on the work by Kölsch and Turk [3]. They combined KLT feature tracker with feature flocking behavior and with color model information to track a hand in video from non-stationary camera. They used this approach mainly for its ability to deal with rapid posture changes, but it is also fast and relatively simple. KLT Tracker is available in public domain and is also part of the OpenCV library which allows fast application development.

4. Head detection

This contains several facilities processing image in order to find participants heads. Skin color detection, background subtraction and connected component analysis. The detection is done in several steps. First a background mask is calculated and then intersected with a mask of skin color. So a result one contains foreground pixels that have color of skin (usually face and hands). The last step is to find components in this mask corresponding to participants faces and drop the others. In most cases faces correspond to compact ellipse-like shapes with distinctive axis aspect ratio in the mask. We use a method of statistical moments to find these components.

The result of the head detection is set of a detected head centers. These points are used on higher level to initialize a KLT tracker.

The following subsections will describe each method to make functionality of a head detection clear.

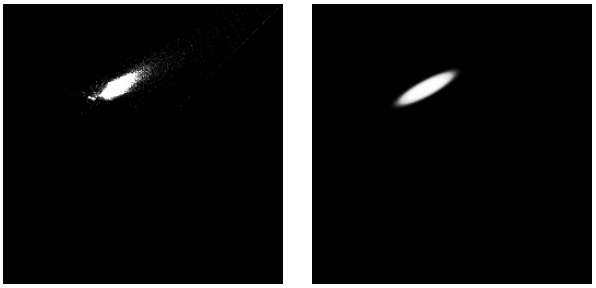
4.1. Skin color model

We use a color model to find areas in image where skin color appear. The input is a RGB image, processing is done in normalized RG space and result is a grayscale skin color likehood image. Algorithm is based on method described in [1].

RG space discards brightness and keeps chrominance and saturation. Therefore skins of many different people are mapped to relative small area, because values vary mostly in brightness. Normalization to RG is described by (1).

$$r=\frac{R}{R+G+B} \quad g=\frac{G}{R+G+B} \tag{1}$$

Where *R*, *G* and *B* are original color values and *r*, *g* are coordinates in *RG* space. Value *b* is redundant due to normalization, where *r* + *g* + *b* = *1*.



(a) (b)

Figure 1.: (a) Distribution of skin samples in RG space. (b) Gaussian model calculated from (a). Images are equalized in order to bring up contrast in dark areas.

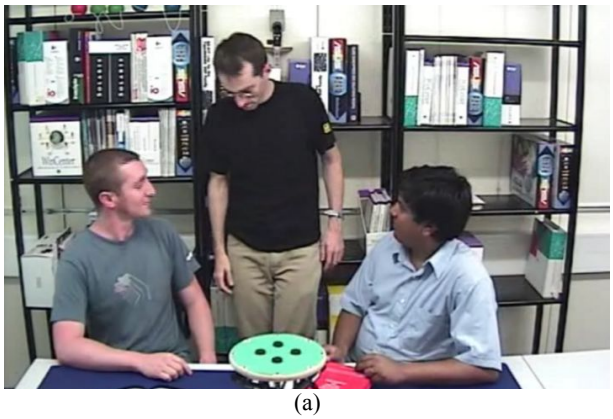
Now we make distribution function from skin samples mapped to RG as shown on figure 1a. This distribution could be either used itself to get probability of particular pixel or we can approximate it using some function. We use approximation by gaussian distribution $N(m,\mu^2)$ where $m=(\bar{r},\bar{g})$ is center of the gaussian function calculated by (2). Parameter N is number of skin samples. Probability $P(x)$ of each pixel $x=(r,g)$ we calculate using the formula (4) where Σ is covariance matrix calculated by (3).

$$\bar{r}=\frac{1}{N}\sum_{i=1}^{N} r_i \quad \bar{g}=\frac{1}{N}\sum_{i=1}^{N} g_i \tag{2}$$

$$\Sigma=\begin{bmatrix}\sigma_{rr} & \sigma_{rg}\\ \sigma_{gr} & \sigma_{gg}\end{bmatrix}=\frac{1}{N}\sum_{i=1}^{N}(x_i-m)(x_i-m)^T \tag{3}$$

$$P(x)=\exp(-0.5(x-m)^T\Sigma^{-1}(x-m)) \tag{4}$$

Using this we transform RGB image to grayscale where value of each pixel corresponds to the skin color likehood. To obtain segmented image we use thresholding.



(a)

(b)

Figure 2.: (a) Original image. (b) Skin color likehood of (a)

4.2. Background subtraction

There are lot of methods extracting foreground parts of image to discard non interesting pixels. Some of them use probabilistic model calculated from certain history of frames (e.g. [6]) or simple methods based on subtraction of background image. Most of them work with static camera because involving motion causes problems. Some problems are also caused by still foreground objects that are evaluated as background; after some time and they disappear. In our case we need an algorithm that will always evaluate objects in foreground even if they are still for an arbitrary time period.

We designed method based on progressive background model improvement. It needs several background—only frames in the beginning of sequence to make initial model. Value of each pixel of a current frame is then compared against value in the model. When difference is higher than a threshold the pixel is evaluated as foreground otherwise it is background. By this we obtain mask of objects in foreground.

Model improvement is done by accumulation of RGB pixel values of current frame in model buffers (every color channel has buffer of its own). Updated are only those pixels evaluated as background. Updating foreground pixels would cause model degradation. And still objects in foreground might then disappear after some time. Accumulation is important to make model adaptive to lighting conditions.

This model is perfectly suitable for our purposes due to the stable light conditions and controlled environment with no extra background movement.

4.3. Component analysis

We need to find out what parts in the mask of relevant pixels have shape of human head. There are many different approaches to do this. We use a spatial component analysis by statistical moment calculation. Note that results are used for tracker initialization, not tracking itself, though it may be suitable even to do tracking.

The analysis is the key step in our approach. Input is a intersection of masks obtained by methods described earlier in sections 4.1 and 4.2. This is processed by median filter to drop out noise pixels and fill small gaps in the mask and even connect components that were not connected before. This step might be seen useless because of computational time it takes. But we figured out that it brings advantage in better detection results. Algorithm then finds all connected parts within mask by flood fill and for each of them calculates its moments using equations (5) and (7). Where x_t and y_t are component center and N number of pixels.

$$V = \frac{\sum_{i=1}^{N} (x_t - x_i)^2}{\sum_{i=1}^{N} (y_t - y_i)^2} \tag{5}$$

$$x_t = \frac{1}{N}\sum_{i=1}^{N} x_i \quad y_t = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{6}$$

$$M_1 = \frac{1}{N}\max^2(X, Y) \tag{7}$$

where

$$X = \max(x_i) - \min(x_i)$$
$$Y = \max(y_i) - \min(y_i)$$
$$i \in \langle 1..N \rangle$$

The x_t and y_t are component center, N number of pixels and x_i, y_i are coordinates of all pixels belonging to the component. Moment V (variance) represents an ellipse aspect ratio. Moment M_1 represents coverage ratio of component over its bounding square. It is used for discarding of components with low compactness ($M_1 < 0.5$). We accept ellipses within range $V \in \langle 0.3; 1.2 \rangle$.

5. Head tracking

We based our tracker on the public domain KLT feature tracker. It provides means to select features in image that are suitable for the tracking and to find most likely location of these features in another image.

The KLT tracker uses an image pyramid (a series of progressively smaller-resolution interpolations of the original image) [7] in combination with Newton-Raphson style minimalization to efficiently find a most likely position of features in new image. Feature displacement is first roughly estimated at coarsest pyramid resolution and this estimation is then refined on more detailed levels of the pyramid. A feature window size determines how much context information is used for tracking. Although larger window sizes may provide better resistance against image ambiguities, it comes with higher computation cost and problems near object silhouettes. In most cases it is appropriate to choose window size 7×7.

We embedded both flocking behavior and color cue suggested in [3] into our tracking system.

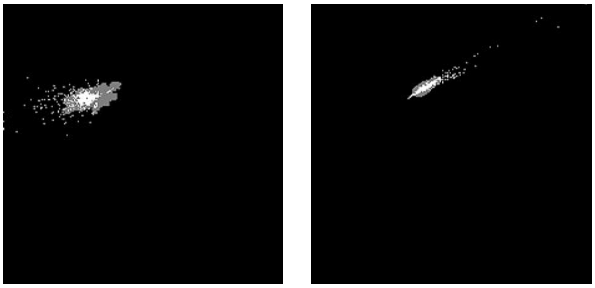
We use the maximum distance from mean position constraint. This means that after processing each frame we discard all features with distance from flock center larger then defined threshold. This is appropriate because we do not expect larger size changes of tracked objects. On the other hand we do not use minimum distance between features constraint, because we have not observed significant local feature cumulation and so we do not think this would bring much performance improvement. However this constraint could easily be included later.

As color cue we use object RG color model that could be either predefined or trained when the tracker is placed on an object. We use this model to discard all features whose color does not match expected object color. The predefined model is the same as we use for skin segmentation during head detection. The learned model is constructed from RG color samples under the tracker position. First we compute the 256×256 2D histogram in RG color space. Then we smooth this histogram to compensate the small number of samples and finally we threshold it. When determining if feature is on correct color we check 4 neighboring pixels. If any of them contains color matching the model we keep the feature otherwise we discard it.

This color cue in combination with the flock compactness criterion almost eliminates feature drift to background and non-stationary objects in the scene. Tracker is also resistant to partial occlusions. Although normalized RG color representation is quite insensitive to light intensity changes, it is sensitive to chromatic changes. While using this color cue, this results in almost certain object loss when the tracked object is illuminated by chromatic light source (e.g. data projector). The object can be also lost as a result of large occlusion or unfortunate combination of background color, object rotation and motion. In our case it is necessary to detect these events.

To detect the object loss we use the same trained object RG color model as described earlier. Percentage of area

beneath the tracker matching the color model is computed each frame. If the percentage drops below given threshold (e.g. 35 %) tracked object is considered lost. Further we compute sum of mean feature velocities over short history (e.g. 10 frames). If this velocity drops too low we assume tracker has drifted to background object and we also discard it. This leads to falsely discarding trackers over temporary stationary objects, but this isn't such a problem because lost objects are in most cases soon detected again. On the other hand the tracker attached to background would otherwise remain there until it is occluded.



(a) (b)

Figure 3.: Gray area represents trained color model. White dots correspond to current color distribution under tracker.

Even when applying these two rules the tracker can still drift to other non-stationary objects with similar color as the original object. In our case these objects are hands and other heads. We have not observed any case when tracker drifted from a head to a hand even when the hand occluded the face. This is due to hand's small size and fast movement which causes that trackers tend to stick preferably to faces that provide a texture rich areas for features to lock on.

To avoid tracker drift to other faces in the image we compute a distance between all trackers in the image. If this distance is smaller then assumed head diameter we discard one of the trackers. In order to choose the occluded tracker we compute for each tracker a sum of lost features over recent history and discard the one with more lost features.

The KLT tracker is able to track features only up to certain maximum displacement which is derived from the number of image pyramid levels, sub sampling between the levels and feature window size. Increasing this maximum distance results in higher computational cost and/or worse tracker performance. In order to increase maximum displacement we predict future tracker position based on current mean feature velocity and acceleration. This also provides little speed up due to the fact that the search for feature correspondence starts at more probable position resulting in less search iteration cycles at coarsest pyramid level.

6. Results

To evaluate tracker performance we have employed common evaluation scheme used in the AMI project [8]. This scheme consists of recorded meeting video sequences with corresponding ground truth annotations as well as methods to evaluate multiple object trackers.

For evaluation we have chosen 8 video sequences with total length of 13 minutes. These sequences represent a set of typical meeting situations and were recorded under constant illumination conditions. The sample from these sequences can be seen on figure 2a.

To measure tracking performance over these sequences we used following measures based on tracker estimation E and annotated ground truth GT:

- FP - False positive. There is E indicating an object, where no GT is.
- FN - False negative. A GT is not tracked by an E.

Decision about correspondence between estimated object E_i and ground truth object GT GT_j is based on F-measure F_{ij} (10).

$$Recall\ \alpha_{ij} = \frac{|E_i \cap GT_j|}{|GT_j|} \tag{8}$$

$$Precision\ \beta_{ij} = \frac{|E_i \cap GT_j|}{|E_i|} \tag{9}$$

$$F_{ij} = \frac{2\ \alpha_{ij}\beta_{ij}}{\alpha_{ij} + \beta_{ij}} \tag{10}$$

If F_{ij} is greater then certain threshold (we use 0.33) objects are considered to be corresponding.

We have evaluated the tracking system as a whole because detection and tracking are meant to complement each other. Where one fails the other should succeed. It would be possible to evaluate head detection and tracking separately, but that wouldn't give us much idea about the resulting performance.

We have compared the performance of the tracker using skin color model trained specifically for each video sequence, with the tracker using general skin color model for all sequences (table 1). Since there were people of distinct skin color in different video sequences, these results confirm that the RG color model is general enough to represent a skin color.

	Universal Model	Specific Model
FP	0,320	0,357
FN	0,085	0,081

Table 1.: Results of tracking using color segmentation with universal color model and model specific to concrete sequence.

Further we have inspected the benefit of the background subtraction. The difference in performance between the case when the background subtraction is used and when it is not is strictly dependent on the skin color threshold. The lower the threshold is the more background objects are considered to have skin color and are classified as heads. Generally the use of background subtraction is beneficial in environment where there are many skin-like colored objects in the background.

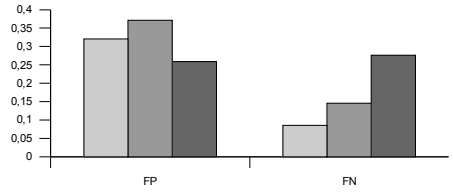
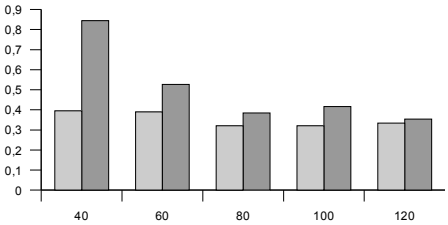
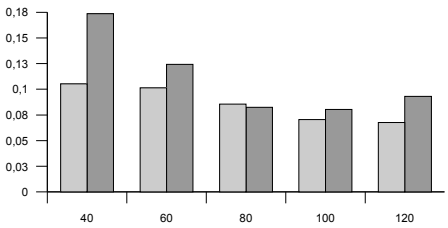


Figure 4.: FP and FN dependence on used color cue. Bright column – object color model, middle – skin color model and dark – no color model

We have also evaluated the benefit of a using color cue in combination with the KLT tracker (figure 4). The color cue brings significant performance gain and better tracker behavior. Training special color model for each tracked object surprisingly proved to be better than using the feature discarding based on skin color model. There is an unexpected decrease in FP rate when no color cue is used, but this is not significant as we plan to reduce FP rate in the future by using additional knowledge about the scene.



(a)



(b)

Figure 5.: (a) FP and (b) FN dependency on threshold for skin color segmentation. Bright column – with background subtraction, dark - without background subtraction

7. Conclusion and future work

We have created the head tracker that tracks heads correctly in more then 90 % of video sequences. On the other hand the FP rate is quite high – around 30 % because hands are often misinterpreted as heads. The FP rate could be reduced using additional topological knowledge about the scene and temporal correspondence, or by using some face detector. It seems promising to use a face detection method described in [9] based on skin color model and several assumptions that verifies face candidates. Otherwise it si possible to use an ellipse fitting presented in [4].

Compared to [4] this paper focuses on inspection of a benefit of different configurations of the head detection to FP and FN rate as shown on figure 4 in chapter 6. Further we evaluated FP and FN dependency on used color cue.

The result implementation achieves approximately 17 frames per second on Athlon 64 3500+ processor. We assume that the speed could be increased at least by factor of two.

Main drawback of the presented tracking system is that the results depend upon many parameters that have to be set manually for different meeting environments. This fact very much limits the usability of the tracker. These parameters should be set as generally as possible or there should be a simple procedure to calibrate the system. Another limiting factor is that the skin color model is sensitive to illumination changes. But this could be avoided by using head detector which is not color dependent, because the skin color model is not necessary needed for tracking.

The main conclusion of this work is that the concept of tracker interconnected with detector is well suited for tracking heads. Also the KLT tracker proved as promising method to track head sized objects.

In the future we plan to use some statistical-based method like a condensation algorithm [10] or Kalman filter to interpret optical flow estimation generated by the KLT feature tracker. This should allow us to track even very small and fast moving objects like hands.

Acknowledgment

Thanks to Adam Herout, Igor Potúček and Standa Sumec for guidance and to Víťa Beran for tracking evaluation software.

References

[1] F. Wallhoff, M. Zobl, G. Rigoll, I. Potucek, *Face Tracking In Meeting Room Scenarios Using Omnidirectional Views*, 2004

[2] J. Shi, C. Thomasi, *Good Features to Track*, IEEE Conference on Computer Vision and Pattern Recognition, 1994

[3] Mathias Kölsch, Matthew Turk, *Fast 2D Hand Tracking With Flocks and Multi Cue Integration*, Department of Computer Science, University of California, 2005

[4] P. Gejguš, M. Šperka, *Face Tracking in Color Video Sequences*, Proc. of SCCG 2003, Budmerice, Slovakia, 2003

[5] O. Sileye, J. Odobez, *A Rao-Blackwellized Mixed State Particle Filter for Head Pose Tracking in Meetings*, International ACM-ICMI Workshop on Multimodal Multiparty Meeting Processing, 2005

[6] A. Elgammal, D. Harrwood, L. Davis, *Non-Parametric Model for Background Subtraction*, 6th European Conference on Computer Vision, Dublin, 2000

[7] J. Bouguet, *Pyramidal Implementation of Lucas Kanade Feature Tracker - Description of the Algorithm*, Microprocessor Research Lab, Intel Corporation

[8] S. Schreiber, D. Gatica-Perez, *Evaluation Scheme for Tracking in AMI*, 2006

[9] M. Sedláček, *Evaluation of RGB and HSV Models in Human Faces Detection*, Proc. of CESCG 2004, Budmerice, Slovakia, 2004

[10] M. Isard, A. Blake, *Conditional Density for Visual Tracking*, International Journal on Computer Vision 29 (1), 1998

[11] B. D. Lucas,T. Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision*, Proc. of Image Understanding Workshop, 1981