A MODULAR ARCHITECTURE FOR REAL–TIME FEATURE-BASED TRACKING

Benjamín Castañeda, Yuriy Luzanov and Juan C. Cockburn

Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
Email: {bxc0593 , yxl4459 , jcceec}@rit.edu

ABSTRACT

A modular architecture for real–time feature–based tracking is presented. This architecture takes advantage of temporal and spatial information contained in a video stream, combining robust classifiers with motion estimation to achieve real–time performance. The relationship among features is exploited to obtain a robust detection and a stable tracking. The effectiveness of this architecture is demonstrated in a face tracking system using eyes and lips as features. A pre-processing stage based on skin color segmentation, density maps and low intensity characteristic of facial features reduces the number of image regions that are candidates for eyes and lips. Support Vector Machines are then used in the classification process, whereas a combination of Kalman filters and template matching is used for tracking.

1. INTRODUCTION

One of the main capabilities of Smart Cameras is their ability of adjusting their position in a purposive manner. For example a smart room could have smart cameras in charge of tracking a person (the user) so his/her gestures and movements are evaluated as input commands. It is apparent that face tracking should be fundamental capability of the system, and since the interaction between the room and the user is required, this should be accomplished in real-time. There is a vast literature on face tracking and face detection [1]. However, doing it in real-time is still a challenge.

An effective approach to face tracking combining appearance based classifiers and motion estimators has been reported in [2, 3]. In these approaches either the face is detected as a whole or only one feature (eyes) is used for tracking. Recent results reported in [4] show that face detection with a component-based approach is superior. Therefore, tracking using the combined information of several facial features should improve the robustness of the algorithm. This gave the inspiration for our modular feature-based architecture for real–time visual tracking where we combine not only spatial but temporal information from several features via data fusion.

A flow diagram of this architecture is shown in Figure 1. It consists of modules for feature extraction, tracking and data fusion.

This modular architecture allows to use state-of-the art classifiers to improve the robustness of detection and tracking. A few years ago, achieving real-time performance with such a system would have required the development of a custom designed chip at a high cost. Moreover, it would have been impossible to keep up with newer algorithms and techniques. However, technology has progressed to the point where reconfigurable computing [5] is a viable and cost effective alternative. The proposed architecture has been designed to take advantage of this new technology.

The paper is organized as follows. Section 2 described the proposed architecture. Then, sections 3 to 5 describe in detail each stage of the feature extraction modules as applied to a face tracking problem. Data fusion is described in section 6. Finally, section 7 summarizes the contributions of the paper and indicates future research directions.

2. FEATURE-BASED TRACKING ARCHITECTURE

The proposed architecture has been designed on the premise that the particular object to be tracked has distinctive features (e.g. eyes, lips, eyebrows in a face). A module, composed by pre-processing, classification, motion estimation and verification stages, is instantiated for each of the features to track (see Figure 1) and it has two operational modes: Detection and Tracking. In detection mode, a classifier is used to obtain an initial position of the feature of interest with high confidence. Since there is a trade-off between confidence and computation time, a pre-processing stage is needed to decrease the time spent in classification, reducing as much as possible the number of candidates for the particular feature. It is important to note that the accuracy of the detection mode is due to pre-processing as much as to the classifier.

In general, the detection mode cannot be used for every single frame because the real-time constraint would be violated. Instead, the module switches to tracking mode, where a motion estimator is designed to track the feature detected by the classifier. After instantiated with the initial position, the motion estimator will predict the position of the feature in the next frame and a verifier will be used to corroborate its location.

Both modes of operation give as a result a list of vectors (f_i, s_i, $\mathbf{x_i}$) containing a candidate identifier (f_i), a score indicating the likeliness of the candidate to be the actual feature (s_i) and its position ($\mathbf{x_i}$). The information, presented by each module, is combined in the data fusion stage, which determines the final position of the individual features. These feature positions are used to update the motion estimators and verifiers from the different modules, and also are used by a mode selector to switch any of the modules from tracking to detection mode. This event is triggered periodically to reset any tracking offset or to reacquire the feature when the track

Sponsored by The Gleason Research Funding.

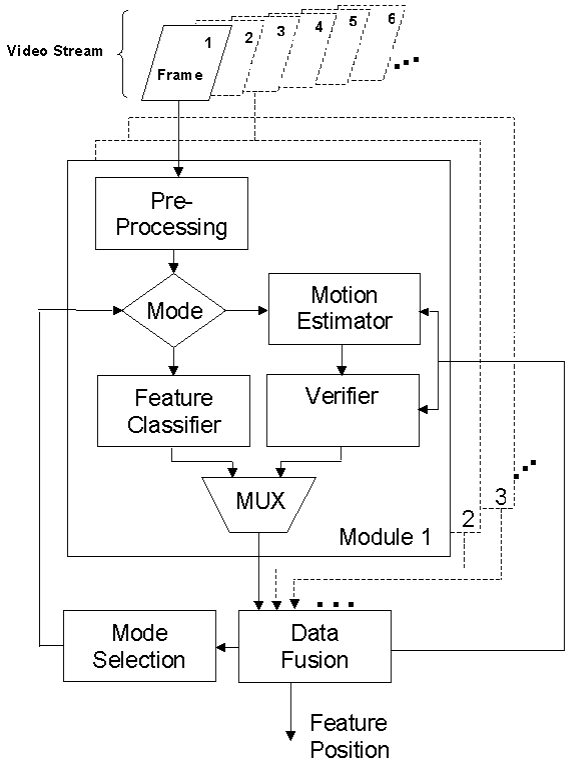


Fig. 1. Tracking Architecture

is lost.

This architecture presents the advantage of using the temporal information available in a video stream to reduce the computationally expensive classification process. It can also take advantage of data fusion algorithms to exploit the relationships among different features to improve the overall (face) detection, tracking and reacquisition. Its parallel design can take advantage of the parallel processing capability of current multiprocessor computers and reconfigurable devices. Depending on the task, it may be sufficient to process only a subset of frames (frame drop) or use an "interlacing" policy to process different features in different frames. This architecture can be implemented with off-the-shelf components allowing rapid and cost-effective prototyping. FPGA and DSP boards configured to speed-up bottleneck processes, specially the classification stage that tends to be computationally expensive.

The effectiveness of this architecture is demonstrated in an experimental face tracking system. Eyes and lips are the features selected for face tracking. According to [6], hand motion can be estimated at 15 frames per second, and since naturally hand motion is faster than head movements, face motion can also be estimated at this rate. Consequently, the interlaced approach will be used: odd frames are evaluated by the lip tracking module while even ones are evaluated by the eye tracking module. The experimental setup is composed of the following hardware: a Dual Xeon Processor 2.66 GHz, a pan/tilt/zoom VC-C4 Cannon Camera and an Osprey 100 frame grabber.

3. PRE-PROCESSING

The pre-processing stage can be divided into two steps. The first step uses a combination of skin color segmentation, density maps and geometric filtering to extract face candidate regions from a frame. The second step extracts the facial features candidates from each face candidate region.

3.1. Skin Color Segmentation

Our approach to skin color segmentation is based on a skin color distribution model. The most common way to represent a color pixel is by encoding the red (R), green (G) and blue (B) values. This color space is in general not suitable for computer vision applications since it correlates brightness ("intensity") and chrominance ("color"). Other color spaces such as HSV, YUV, YCrBr and CIE Lab separate the chrominance information and often involve nonlinear transformations from the RGB space. This additional computation can not be afforded given the real–time constraint. To address this issue, a skin model was developed based on a study of skin color in the HSV colorspace. Then, to reduce the computation time this model was mapped to a normalized rgb space which is related to the RGB space by the following equations:

$$r = R/(R+G+B), g = G/(R+G+B), b = B/(R+G+B) \quad (1)$$

This normalization is equivalent to projecting one particular color over the plane $R + G + B = 1$. Therefore, given the values of r and g, b is fixed ($b = 1 - r - g$). The normalized color can thus be represented by only r and g values. Finally, to increase the execution speed, a look-up table (LUT) was used to encode the skin color information. The current LUT has a size of (100x100). This approach has a modularity advantage. The skin color model can be developed in any other colorspace and then mapped back into the r, g LUT.

3.2. Density Mapping and Region Filtering

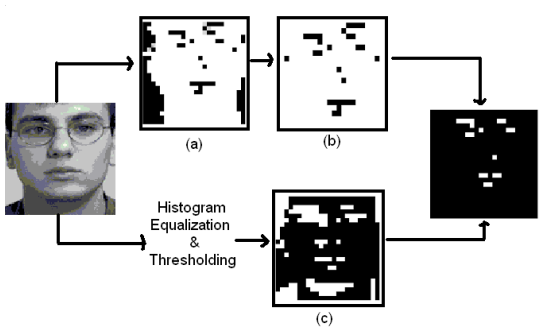
After skin segmentation the image is converted to a density map (down scaling) [7]. In this process, blocks of pixels are mapped to a single pixel with value equal to the number of skin pixels in the block.

This operation reduces the size of the image significantly (e.g., by 1/16 if 4x4 blocks are used) making it possible to eliminate spurious skin color pixels (due to variations in lighting conditions and other interferences) without compromising the real–time performance of the system. Figure 3 shows an image and its density map in the bottom left corner.

At this point thresholding followed by connected components analysis (connectivity four) is applied to reduce the number of skin regions. In our system blocks containing more than half skin color pixels are kept. The resulting skin regions are further processed using the following scale and biometric heuristics:

1. The aspect ratio of a region should be between 1 and 1.4.
2. The number of pixels in a region should be bigger than a predefined minimum.
3. The ratio of region area to bounding box area should be bigger than a prescribed minimum.

The regions that satisfy all these conditions become face candidates to be analyzed further.



(a) Skin Mask (b) Extended Skin Mask, (c) Low intensity mask

Fig. 2. Facial Feature candidates

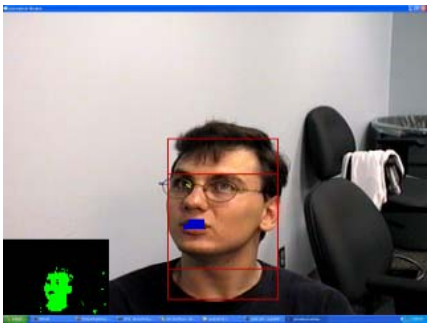


Fig. 3. Example of lip classification

3.3. Feature Candidates Selection

A simple and effective way to identify face features is to analyze the points considered non-skin inside a skin region [8]. One drawback of this approach is that illumination conditions can create false non-skin candidates (e.g. light reflected in the forehead is not considered skin). To solve this problem, the fact that facial features have lower intensity than the rest of the face is used.

For each region considered a face candidate, an intensity mask over the area of its bounding box is created by averaging every 4x4 block to match the size of the density map for the same region (skin mask). The information of both masks is combined to obtain possible facial features. As shown in Figure 2, the following procedure is applied:

1. Histogram equalization of the intensity mask.
2. A threshold is applied to the equalized intensity mask. Everything below an intensity value of 51 is considered as low intensity.
3. The skin mask is extended to fill the bounding box.
4. The extended skin mask is reversed to obtain non-skin "holes" inside the face candidate region.
5. All points considered low intensity and non-skin are kept as candidates for facial features to be analyzed by the classifier.

Up to this point, a frame from the video stream has been analyzed to obtain facial feature candidates in possible face regions. These candidates will be classified in the next stage of the algorithm.

4. CLASSIFICATION

In this system Support Vector Machines (SVM) are used for feature detection. SVM have been applied successfully in face detection [9, 2] and facial feature extraction [4]. They are based on machine learning principles [10] and therefore must be trained. In this section, the training process and the classification results are presented.

Building a training set for a SVM is a very delicate and tedious task. It is important for the SVM to be trained with data as close as possible to what they are going to classify. Two programs were written to perform the training process. The first one follows the algorithm described in the previous section. For each facial feature candidate, a 10x20 image was extracted and classified by a human operator. The second program was designed to guarantee the integrity of the training and data sets; the already classified features grouped by class are shown to an operator who verifies the classification. 13 subjects from different ethnicity went through this process, in which they were asked to move while looking at a video camera. Only small movements such as yaw, roll and pitch of +/- 10 degrees were allowed. The training set contained 15308 images and the test set 5347 images of eyes, eyebrows, nostrils, lips and hair. Two classifiers (for eyes and for lips) were trained with the training set changing the positive samples according to the class in turn. The SVM were trained using the Matlab toolbox [11] which is based on Platt's SMO Algorithm [12].

To choose the machine Kernel and the cost of misclassification parameter (C), experiments were performed with linear, polynomial and radial basis function (RBF) kernels using a smaller training set (5000 samples) and varying C from 0.01 to 100. The best performance was obtained using a polynomial kernel of second degree and $C = 0.01$.

The following table summarizes the performance of the classifier:

	# SV	Accuracy Training Set	Accuracy Test Set	Min(ms)	Max(ms)	Avg(ms)
Eyes	2415	98.80%	91.10%	47	177	109.03
Lips	2261	98.65%	91.10%	37	209	77.3

Table 1. Performance and Accuracy of the SVM Classifiers

The above table shows robust results for the SVM classification at the expense of real–time performance. Even if SVM are only used for obtaining the initial position of a feature, the first frame will have a significant delay. In Figure 3, an example of lip detection is shown. The face region is highlighted by a bounding box and the lips by square dots. The classifier generates a list of candidate features, scores and positions that feed the data fusion stage.

5. MOTION ESTIMATION AND VERIFICATION

A Kalman Filter was designed for motion estimation and was combined with a template matching technique for verification purposes. The parameters of the Kalman Filter are initialized following the criteria described in [6].

The SVM classifier initializes the first measurement and the first (10x20) template. In subsequent frames, the Kalman Filter estimates the next position of the feature and this estimate is used to define a search window for template matching. In its current implementation this windows is defined as a neighborhood of 9x9 pixels around the estimated position. All the points evaluated are presented to the Data Fusion stage (See Section 2). The Data

Fusion stage will return the candidate identifier recognized as the feature of interest. Then its position is used to update the Kalman Filter and update the template for the next evaluation.

Table 2 shows the performance of the detection and tracking modes of the lips and eye modules independently without the Data Fusion Stage. The performance is measured by the minimum, maximum and average time for the module to process a frame. Table 2 also shows the maximum and average number of frames spent in the tracking mode of the module. These results indicate clearly the advantage of switching to a tracking mode after a detection step.

	Min(ms)	Max(ms)	Avg(ms)	Max # frames on track	Avg # frames on track
Eyes	9	178	33.3	10	3
Lips	9	164	18	11	4

Table 2. Performance of SVM Classifier combined with Kalman filter and template matching

6. DATA FUSION AND MODE SELECTOR

The Data Fusion stage combines the information from different modules (e.g., eyes and lips) to give a better estimation of the face position, in time and space, and help reacquiring lost features using geometric information about relative position between lips and eyes. The data fusion stage performs the following tasks:

1. It decides with the combined information if one of the modules is out of track.
2. It returns to each module the identifier of the candidate to be considered the feature of interest.
3. If both features are out of track, it first runs the lip classifier, and in the next frame it combines this information with the result of the eye classifier.
4. If one feature is out of track, it weights the result of the classifier according to the euclidian distance to the region where the feature is expected.
5. If one of the features has been on track for more than 11 consecutive frames, the classifier is called to evaluate the next corresponding frame, and its result is weighted according to the last position known from the motion estimator.

If the Data Fusion stage decides that one module is out of track, it will alert the mode selector stage, which in turn will send a message to the specific module to change to detection mode.

This stage is yet to be added to the current implementation of face tracking. It is expected that it will improve the reacquisition of lost features and the real–time performance of the system.

7. CONCLUSION AND FUTURE WORK

A modular feature–based tracking architecture suitable for real-time visual tracking algorithms has been proposed and applied to an experimental face tracking system. Two independent tracking modules, one for eyes and one for lips, have been implemented using SVM classifiers combined with Kalman Filters and template matching. Experimental results demonstrate that a real–time performance can be achieved. The SVM classifier proves to be effective. However, the number of support vectors for each classifier is too large and may introduce a noticeable delay. This problem can be alleviated by reducing the number of Support Vectors while minimizing the degradation of the classifier [13, 9]. Furthermore, the implementation of the data fusion stage is almost complete and the results will be reported shortly.

8. REFERENCES

[1] D. J. Kriegman M. Yang and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[2] V. Kumar and T. Poggio, "Learning-based approach to realtime tracking analysis of faces," *Technical Report 1672 - MIT*, 1998.

[3] S. Spors and R. Rabenstein, "A real-time facetracker for color video," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1493–1496, 2001.

[4] J. Wu B. Heisele, P. Ho and T. Poggio, "Face recognition: component-based versus global approaches," *Computer Vision and Image Understanding*, vol. 91, pp. 6–21, 2003.

[5] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, June 2002.

[6] M. Kohler, "Using the kalman filter to track human interactive motion - modelling and initialization of the kalman fitler for translational motion," Tech. Rep. 629, University of Dormund, January 1997.

[7] V. Kravtchenko, "Tracking color objects in real time," M.S. thesis, The University of British Columbia, 1999.

[8] E. Saber and A. M. Tekalp, "Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost-functions," *Pattern Recognition Letters*, vol. 19, no. 8, pp. 669–680, 1998.

[9] B. Scholkopf S. Romdhani, P. Torr and A. Blake, "Computationally efficient face detection," in *8th International Conference on Computer Vision*, 2001, vol. 2, pp. 695–700.

[10] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[11] G. C. Cawley, "MATLAB support vector machine toolbox (v0.50β) [http://theoval.sys.uea.ac.uk/˜gcc/svm/toolbox]," University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ, 2000.

[12] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Microsoft Research, 1998.

[13] C. J. C. Burges, "Simplified support vector decision rules," *International Conference on Machine Learning*, pp. 71–77, 1996.