A Multi-Threaded Approach to Real Time Face Tracking

UCSD MPLab TR 2000.02

Boris E. Shpungin
Department of Cognitive Science
University of California San Diego

Javier R. Movellan*
Institute for Neural Computation
Department of Cognitive Science
University of California San Diego

Abstract

We present a multithreaded approach to real-time face tracking. The system consists of a set of modules running on different processing threads and coordinated by a mini operating system. The first module uses color, shape, and face dynamics to determine a 2-D region of interest. The second module is a mixture of linear experts classifier whose task is to help determine whether the region of interest is indeed a face.

We present a fast and reliable 2-D face tracking system consisting of two modules coordinated by a scheduler. The system was inspired on Toyama's Incremental Focus of Attention architecture [6] but instead of using a sequential architecture our system uses a multithreaded approach with modules running in parallel in independent threads coordinated by a scheduler. The first module is very fast, and can operate more than one time per video frame on a mid-end PC workstation. This module uses three sources of information: color, shape, and face dynamics to determine a rectangular region containing a face. The second module is slow and utilizes a mixture of experts neural network [2, 1] to help determine whether the region being tracked by Module I is indeed a face. The two modules are coordinated in real time by a scheduler, which acts as a mini-operating system, distributing processing resources in real time to the two modules.

Throughout the paper we use standard probabilistic notation: Random variables and processes are defined on a probability space $(\Omega, \mathcal{F}, P)$, random variables and processes are denoted using capital letters and Greek-letters. Specific samples of these variables and processes are represented with small letters. $R(t) = r(t)$ is short notation for the event that the random variable $R(t)$ takes the specific value $r(t)$, i.e., the set $\{\omega \in \Omega : R(t, \omega) = r\}$. Estimates are represented using hat-notation, e.g., $\hat{\mu}(t)$ is an estimate of $\mu(t)$.

The goal of the system is to find the smallest vertically oriented rectangle which contains a face of interest (see Figure 2). We denote the region of interest (ROI) at time t as $R(t)$ and parameterize it using two location parameters $\mu(t)$ and two scale parameters $\sigma(t)$.

*This work was partially funded by a Microsoft Research Gift.

1 Module I

This module consists of the following processes: (1) Active sampling; (2) Summary statistics; (3) ROI estimation; (4) Uncertainty estimation; (5) Initialization and disengagement.

Active Sampling: Module I has access to a current image of the world, and to an estimate $\hat{R}(t-1)$ of the region of interest at the previous time the module was called. Based on this information the goal is to estimate as fast and precisely as possible the ROI at time t. To save time, instead of analyzing the entire image, we concentrate on the most promising pixels, a process called active sampling. In the current implementation we assume the most informative pixels are around the previous ROI and sample $n = 1000$ pixels randomly selected with equal probability from a slightly extended version of $\hat{R}(t)$.

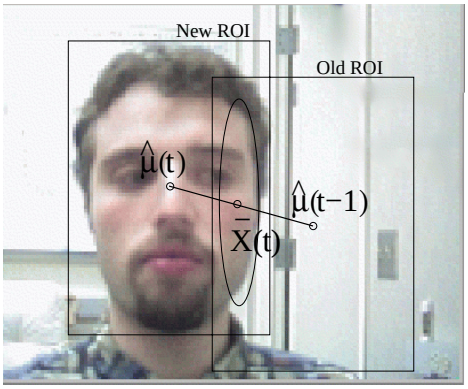

Figure 1: The summary statistic $\bar{X}(t)$ is based on pixels located on the ROI at time $t-1$. Due to the active sampling scheme and the color model, only those pixels inside the old ROI and inside the face have a significant influence on $\bar{X}(t)$. The statistic can be used to estimate the location of the new ROI.

Summary Statistics: Let $X_1(t), \cdots, X_n(t)$ be random variables representing the 2-D coordinates of the n randomly selected pixels. Let $C_1, \cdots, C_n(t)$ represent the HSV color coordinates of these pixels. Let $x_i(t) \in \mathbb{R}^2$, $c_i(t)\mathbb{R}^3$ represent specific values taken by these random vectors. Speed is of the essence for Module I and thus the information in the 1000 pixels is first summarized into a few statistics to maximize the processing speed of later stages. Specifically, Module I uses, $\bar{X}(t)$ and S^2, two two dimensional statistics for location and scale of the obtained distribution of pixels:

$$\bar{x}(t) = \frac{\Sigma_{i=1}^{n} x_i(t) P(x_i(t) \in R(t), C_i(t) = c_i(t) \mid \hat{R}(t-1) = r(t-1))}{\sum_{i=1}^{n} P(x_i(t) \in R(t), C_i(t) = c_i(t) \mid \hat{R}(t-1) = r(t-1))}, \tag{1}$$

$$s^2(t) = \frac{\Sigma_{i=1}^{n} (x_i(t) - \hat{\mu}(t-1))^2 P(x_i(t) \in R(t), C_i(t) = c_i(t) \mid \hat{R}(t-1) = r(t-1))}{\sum_{i=1}^{n} P(x_i(t) \in R(t), C_i(t) = c_i \mid \hat{R}(t-1) = r(t-1))}, \tag{2}$$

where $\bar{x}(t) \in \mathbb{R}^2$, $s^2(t)$ represent the specific values taken by $\bar{X}(t), S^2(t)$ when $X(t) = x(t), C(t) = c(t), \hat{R}(t-1) = r(t-1)$. The conditional distribution used

in (1) and (2) encodes color and shape information and is assumed to factorize as follows:

$$P(x_i(t) \in R(t), C_i(t) = c_i(t) \mid \hat{R}(t-1) = r(t-1)) =$$
$$P(x_i(t) \in R(t) \mid \hat{R}(t-1) = r(t-1)) P(C_i(t) = c_i(t) \mid x_i(t) \in R(t)). \quad (3)$$

The flesh-color model $P(C_i(t) = c_i \mid x_i(t) \in R(t))$ was inspired on McKenna's work [3, 5]. The model was constructed using 1230 samples from skin in photographs from an Internet singles site. The samples contained male and female individuals of a wide variety of races and many different skin tonalities. When the results were plotted in Hue-Saturation coordinates (HS), the S values were nearly uniformly distributed across the entire saturation range while the H values formed a Gaussian blob. Since most color information was in the hue, we discarded the H and V coordinates. Hereafter $C_i(t)$ will represent the hue value of pixel $X_i(t)$. The color model was defined follows:

$$P(C_i(t) = c_i \mid X_i(t) \in R(t)) \propto \exp(-\frac{d^2(h, 17.95)}{12.2^2}), \quad (4)$$

where $\mu = 17.95$, $\sigma = 12.2$ and d is the angular distance in degrees, between h and μ.

The shape model $P(x_i(t) \in R(t) \mid \hat{R}(t-1) = r(t-1))$ was constructed as follows: First we determined that the ideal aspect ratio of the ROI should be 1.2, i.e., ideally the height of the ROI should be 1.2 times its width. If $\hat{R}(t)$ is too tall (i.e., $\hat{\sigma}_2(t)/\hat{\sigma}_1(t) > 1.2$) Module I computes the widest rectangle inside $\hat{R}(t)$ centered at $\hat{\mu}(t)$ and with the desired aspect ratio. If $\hat{R}(t)$ is too wide (i.e., $\hat{\sigma}_2(t)/\hat{\sigma}_1(t) < 1.2$) Module I computes the tallest rectangle inside $\hat{R}(t-1)$ centered at $\hat{\mu}(t-1)$ and with the desired aspect ratio. Let $\tilde{\sigma}(t)$ represent the scale parameters of that rectangle. The shape model was a 2-dimensional Gaussian distribution controlled by those parameters:

$$P(x_i(t) \in R(t) \mid \hat{R}(t) = r) \propto \exp\left(\frac{x_{i,1}(t) - \hat{\mu}_1(t-1)}{0.8\tilde{\sigma}_1(t)}\right)^2 + \left(\frac{x_{i,2}(t) - \hat{\mu}_2(t-1)}{0.8\tilde{\sigma}_2(t)}\right)^2. \quad (5)$$

The idea here was that if the estimated ROI was unusually tall or wide it probably was due to the fact that the face was occluded by an arm. This shape model made the system robust to occlusions by flesh and non-flesh colored objects (see Figure 2).

ROI Estimation: We use the statistic of location to estimate the velocity of the ROI and update the ROI parameters (see Figure 1). We found the following equation worked well for this purpose:

$$\hat{\mu}(t) = \hat{\mu}(t) + 2(\bar{X}(t) - \hat{\mu}(t-1)). \quad (6)$$

As illustrated in Figure 1 we expect the statistics of scale to be most reliable when the velocity of the ROI is small. Based on this idea, our estimate of scale relies more on the current summary statistics if the velocity is small and more on previous estimates if the velocity is large. In practice we used the following formula:

$$\hat{\sigma}(t) = 2.5(\beta(t)S(t) + (1 - \beta(t))\sigma(t-1) + \epsilon), \quad (7)$$

where $\epsilon = (1.5, 1.5)^T$ prevents the scale estimate from being zero, and $\beta(t)$ is an estimate of the reliability of the current scale statistic based on the speed of the ROI

$$\beta(t) = 1 - \frac{\|\bar{X}(t) - \hat{\mu}(t-1)\|}{\|\sigma(t-1)\|}. \quad (8)$$

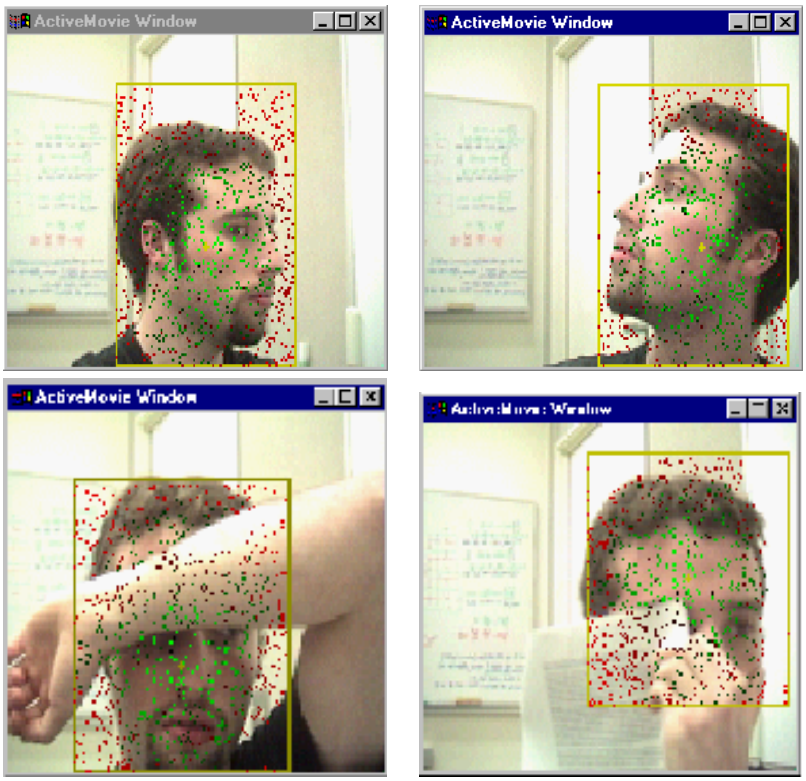

Figure 2: Module I at work. Note its resistance to occlusion by flesh and non-flesh colored objects.

Uncertainty estimation The interaction of Module I with the Scheduler was modulated via estimates of confidence. The confidence for the current ROI estimates was defined as follows:

$$\kappa(t) = \frac{W(t)}{(1+\delta(t))\tilde{W}(t)}, \tag{9}$$

where $\delta(t) = \|\hat{R}(t) - \hat{R}(t-1)\|$, is an estimate of the ROI velocity,

$$W(t) = \frac{1}{10n}\sum_{\tau=0}^{9}\sum_{i=1}^{n} P(x_i(t-\tau) \in R(t-\tau), C_i(t-\tau) = c_i(t-\tau) \mid \hat{R}(t-1) = r(t-\tau)), \tag{10}$$

and

$$\tilde{W}(t) = \frac{1}{10n}\sum_{\tau=0}^{9}\sum_{i=1}^{n} P(\hat{\mu}(t-\tau) \in R(t-\tau), C_i(t-\tau) = c_i(t-\tau) \mid \hat{R}(t-1) = r(t-\tau)), \tag{11}$$

are running averages of the color/shape and color-only likelihoods in the estimated ROIs. The priority value sent to the Scheduler decreased with the confidence of the current ROI and with the number of times the module had been called on the

current video frame (Module I is very fast and can be called more than once within a video frame)

$$\gamma(t) = \frac{1 - \kappa(t)}{1 + N}, \tag{12}$$

where N represents the number of times Module I operated on the current video frame.

Initialization and disengagement At the start of the tracking process, the first module simply produces random initial estimates for each of the ROI parameters, and proceeds with tracking using these estimates. Failure detection and disengagement was controlled by $W(t)$, the moving average of the likelihood values obtained in the past 10 time steps. Whenever $W(t)$ drops below a given threshold, it is taken as an indication that tracking has failed and Module I is reinitialized to new random ROI parameters.

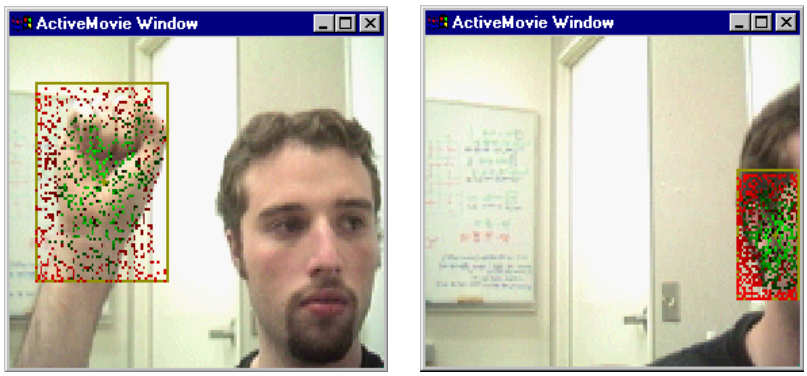

Figure 3: The first module provided good candidates for faces, but at times it would mistakenly track non-face objects. Module II provided additional intelligence to solve this problem.

2 Module II

The primary task of Module I was to narrow down the input to a small ROI. The module worked very well but occasionally, due to its lack of knowledge about the high order statistics of face shape, it would mistakenly track flesh-like objects with face-like aspect ratios (see Figure 3). Module II was designed to address this problem.

Module II operated on the ROI produced by Module I. First, the estimated ROI was cropped to a square around its center. The obtained image was scaled to a 40x40 bitmap. The flesh color model was applied at every pixel and the resulting pixel values were linearly scaled to range between 0 and 255. Working with flesh color likelihoods instead of gray-level values increased greatly the accuracy of Module II by providing a representation which was resistant to changes in illumination and background (see Figure 4). The obtained image was convolved with a bank of Gabor energy filters of different scales and orientations. As suggested by previous work, for each complex Gabor filter we computed the magnitude and discarded the phase [1].

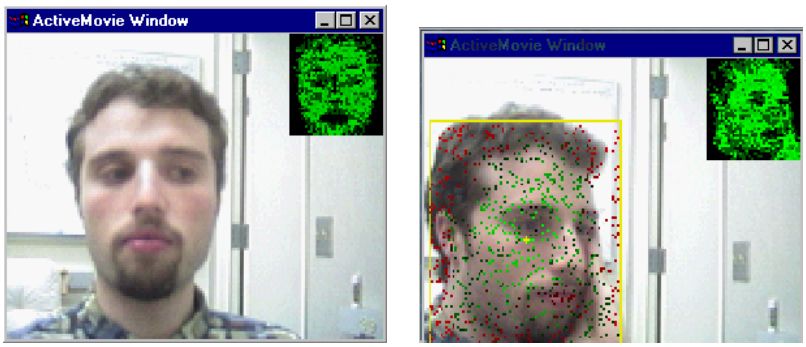

Figure 4: Raw images and color-likelihood images produced by Module I (top right corner of each image).

Due to the fact that the ROI is scaled to dimensions of 40x40, it is convenient to define Gabor kernels that have dimensions of 10x10 and 20x20, because such kernels can be laid out within the image in a regular grid (which makes them easier to apply). Figure 5 shows the locations within the normalized image at which the Gabor kernels were applied. Nine different 20x20 Gabor kernels were applied at each of the 9 points shown in black in Figure 5. The 9 kernels had orientations of 26, 77, and 129 degrees, and spatial frequencies of 5, 10, and 20 pixels per cycle. Eight 10x10 kernels were applied at each of the 16 points shown in Figure 5. They had orientations of 0, 51, 103, and 154 degrees, factorially combined with two spatial frequencies of 10 pixels/wavelength and 5 pixels/wavelength. The envelope for the Gabor kernels had standard deviation of 2.5 pixels for the 10x10 patches, and 5 pixels for the 20x20 patches. The coefficients that result from applying the Gabor kernels were linearly scaled to range between [-1,1] based on the maximum and minimum possible coefficient values that can in principle be obtained with a particular kernel. This processing of the ROI results in 209 real-valued magnitudes between -1 and 1, the input to our face recognition classifier.

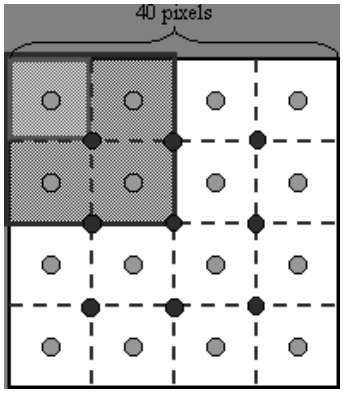

Figure 5: Locations of Gabor kernels within normalized image. Dark dots are positions of 20x20 kernels; light dots are 10x10 kernels. As examples, the left upper corner shows areas covered by a 10x10 patch (light square), and a 20x20 patch (dark square).

2.1 The Classifier

The face recognition system was a Mixture of Linear Experts [2, 1]. The classifier was trained with a database of face and non-face flesh images of the first author. There were 306 images of the face under taken across 7 different angles of roll, pitch, and yaw (ranging from -90 to +90, through 0), as well as 30 random combined orientations with occasional extreme facial expressions (for a total of 51 pictures) against 6 different backgrounds. The database contained 400 images of hands, fists, arms, elbows, upper arms, and lower legs in various positions and against the 6 different backgrounds. In addition images of non-face non-flesh objects were also included. In total, the database contained 906 images.

Images containing the face and non-face body parts were run through the first module to capture their ROIs. For the non-flesh images the entire image was treated as the ROI. Each of the three sets of images in the database was randomly split into three roughly equal parts used for training, validation and test. The network never saw the test set during training, and this set of images was used purely to evaluate classification accuracy and generalization. The validation set was used to stop training at the first signs of over-learning (i.e. decreased performance on the validation set). Best generalization performance was obtained with a network of 5 experts. Adding more experts did not improve performance, while having fewer experts hurt performance. The error rate of this network on the test database was 10 %.

3 The Scheduler

The tracker is currently written in Microsoft Visual C++ 6.0, and utilizes the Microsoft DirectX Media API under Windows NT 4.0 for handling the tasks of capturing and rendering video. The modular design of the DirectX Media filters enables us to literally hook up a video capture filter to our face tracker, and direct the output from the face tracker to a video rendering engine.

Modules run in their own processing threads, and the modules are managed by yet another thread in charge of scheduling. The Scheduler can be seen as a mini-operating system in charge of distributing resources to the different modules.

The multithreaded design, at least in principle, enables the system to scale well on multiprocessor systems. Also, the priority-based scheduling scheme enables the system to potentially invoke several modules per frame, or indeed a single module several times per frame - if there are enough computing resources available. Since video frames come in at regular intervals, the Scheduler can dynamically track such intervals and, by timing the execution of modules, it can determine how much time remains in the current time slice before a new video frame comes in. If there is enough time to invoke another module, the Scheduler promptly does so. Indeed, in our case the first module is so efficient that upon initialization the Scheduler can invoke it up to 4 times per frame. The second module is much more computationally intensive, and can only be invoked once per frame - although, sometimes there is enough time left in the very same frame to also invoke Module I.

Priority is inversely proportional to the confidence given by the module to its latest batch of computed parameters. For example, in our case Module I does very primitive motion tracking, and needs to be invoked on practically every single video frame. Module II, on the other hand, does not care when it is invoked, because it does not rely on any information obtained from its previous execution. The Scheduler sorts the currently active modules by priority, and always invokes the module

with the highest priority first. Module II is initially inactive, and becomes activated only when the first module's priority drops below a certain threshold (indicating that the first module is fairly confident about its estimated ROI.) If Module II detects a string of 3 consecutive failures, it raises a warning flag and requests to be deactivated, at which point all processing is once again devoted to Module I. When Module I detects the warning flag over a sufficient number of invocations, it disengages tracking and reinitizializes its face detection process.

Module II does not benefit from being executed more than once on the same image; hence its priority routine is designed to output 0 after the first invocation, and then produce gradually increasing numbers as the frame index changes. Module I, on the other hand, does benefit from multiple invocations due to its tracker component - by using Module I several times on a single video frame, the tracker can both probe the image more extensively during initialization, and track the face more accurately later on.

4 Conclusions

We are currently in the process of evaluating more quantitatively the success and failures of the approach. Pending a more formal evaluation, the system performed extremely well with a very wide variety of people that visited our lab. One of its most impressive aspects is the speed with which it finds and locks onto faces as people enter the lab. Its resistance to occlusions is also very impressive. On the negative side, unusual illumination conditions, like the blue lighting employed by a TV crew, proved too difficult to handle. The use of adaptive color models may help solve this problem. The system also fails if the background is flesh-colored. However, when this happens the system knows that it is failing and alerts the Scheduler of the problem. New modules will be constructed in the future, based on motion or other approaches to take care of this problem. In addition we are working on a more theoretical model for the scheduler, based on the ideas of active sampling and sequential statistics [4].

References

[1] M. N. Dailey and G. W. Cottrell. Organization of face and object recognition in modular neural network models. *Neural Networks*, 12(7):1053–1073, 1999.

[2] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[3] S. J. McKenna, Y. Raja, and S. Gong. Object tracking using adaptive colour mixture models. *Lecture notes in computer science*, 1(1351):615–622, 1998.

[4] J. D. Nelson and J. R. Movellan. Active inference in concept induction. In *Proceedings of the 7th Symposium on Neural Computation*. UCSD's Institute for Neural Computation, 2000.

[5] Y. Raja, S. J. McKenna, and S. Gong. Segmentation and tracking using colour mixture models. *Lecture notes in computer science*, 1(1351):607–614, 1998.

[6] K. Toyama. Prolegomena for robust face tracking. Technical Report MSR-TR-98-65, Microsoft Research Technical Report, 1998.