Towards Real-Time Hand Tracking In Crowded Scenes

Matthew N. Dailey and Nyan Bo Bo
Sirindhorn International Institute of Technology (SIIT)
Thammasat University
Pathumthani, Thailand 12121
Email: {mdailey,nyanbobo}@siit.tu.ac.th

***Abstract*— Being able to detect and track human hands is one of the keys to understanding human goals, intentions, and actions. In this paper, we take the first steps towards real-time detection and tracking of human hands in dynamic crowded or cluttered scenes. We have built a prototype hand detection system based on Viola, Jones, and Snow's dynamic detector, which was originally constructed to detect and track pedestrians in outdoor surveillance imagery. The detector combines motion and appearance information to rapidly classify image sub-windows as either containing or not containing a hand. A preliminary evaluation of the system indicates that it has promise.**

I. Introduction

If we are ever to realize the goal of autonomous mobile robots able to interact with us in everyday life, we will have to overcome many obstacles. One of the most significant is the current lack of technology for perceiving and interpreting the structure of the world and the agents acting in it.

In this paper, we focus on a particular problem relevant to mobile robot applications in personal services, health care, and security: detecting and tracking human hands. A robot able to find and track human hands in real time would be able to accomplish many tasks. It could accept gesture-based commands from humans [1], [2], interact socially with humans, help patients in and out of bed, and so on. It could also detect and/or respond to security incidents, such as shoplifting, pick-pocketing, and assault.

Over the last 15 years or so, a great deal of research has focused on the problem of hand tracking. To date, the vast majority of systems have been aimed at empowering human computer interaction or sign language recognition. The early hand tracking systems relied on uncluttered static backgrounds, high resolution imagery, and manual initialization. These systems could track hands reliably and accurately, so long as the constraining assumptions held.

One of the first such systems was DigitEyes [3], which is able to track a human hand with 27 degrees of freedom at 10 frames per second (fps), given an already-initialized model. DigitEyes predicts the appearance of the hand in two stereo images given a previous state estimate then searches for the nearest matching features in the actual input. The actual measured feature positions are then used to obtain a maximum likelihood estimate of the hidden kinematic state by linearizing the transform from state to image appearance around the state estimate from the previous time. As long as there is no occlusion and the image features do not move too much between frames, the system can track a single hand with amazing accuracy.

Ahmad's tracker [4] was perhaps the first system to perform 3D hand tracking in real time with arbitrary background clutter. The tracker first performs color segmentation then applies a variety of classical computer vision techniques to identify the palm of the hand, its planar orientation, and the orientation of the fingers. It estimates depth changes using changes in the size of the palm. The system only works robustly when the hand is approximately parallel to the image plane.

Segen and Kumar built a more robust system [5] that is capable of tracking a hand in good imaging conditions through four different gestures. It provides a 10 degree of freedom estimate of hand's position: five for the 3D position and orientation of the thumb, and five for the 3D position and orientation of the index finger.

Many more hand tracking systems have appeared in recent years, but nearly all of them rely on fairly detailed models of the hand. Some, such as Triesch and von der Malsburg's [6], take a pattern recognition approach and are quite robust to clutter, but the systems still all assume a fairly high resolution view of the hand. This assumption is fine for "virtual mouse" and sign language applications, but it is unrealistic in the applications we have in mind. When our fictitious security robot spots the "slipping the gold watch into the pocket" gesture, for instance, the hand being observed might only be, say, five pixels wide in the camera image.

Systems like Pfinder [7] are probably more applicable to our task. Pfinder finds and tracks entire human bodies first, then finds the body parts. The body is assumed to be made up of a collection of colored blobs. A human's hands are normally easy to segment from the rest of the body using color, under favorable imaging conditions. We could in principle use a similar approach to first find the humans in the scene then find the hands of those humans.

But the task of finding humans in video streams is itself an extremely difficult problem, and the technique would preclude finding the hands of people that are mostly occluded by objects or other people. Also, reliable color information may not always be available, especially in low-light or surveillance camera applications. We take an entirely different approach in this paper. The goal is to detect multiple hands in a cluttered, crowded scene, *without first detecting the human bodies*.

Classifying, say, a 5×5 blob in a gray scale image as a hand and not a face or a piece of paper taped to the wall might at first seem to be impossible. Indeed, we do not know of any existing system capable of performing the task. However,

there is some hope: *motion* is an extremely salient source of contextual information that can help us distinguish "hand blobs" from other blobs. The power of motion as a cue to the identity of a moving object is well known, especially for human and animal motion. For instance, Polana and Nelson [8] find that the spatio-temporal profiles of low-level features can be used to automatically classify a scene as a human walking, running, jumping, and so on. Perhaps hands in typical crowded scenes have similarly characteristic motion profiles and can be detected according to their motion.

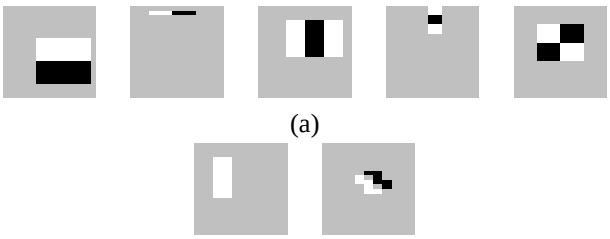
In this paper, we report on preliminary experiments applying Viola, Jones, and Snow's dynamic object detector [9] to the task of hand tracking in crowded scenes potentially containing many humans. The results thus far are promising, but there is much left to be done. First we describe the system, then we present preliminary results on a single video sequence, then we conclude with future directions for the research.

II. The Dynamic Object Detector

Viola and Jones created a great deal of excitement in the computer vision and machine learning communities when they first demonstrated the fastest-ever human face detection system [10]. The idea of the V&J detector, as in many other object detection systems, is to sweep a search window over the input image at multiple scales, and classify each window as either a member of the class or not a member of the class. The difficulty is twofold: the classifier must be extremely robust, to handle noise, partial occlusion, and background clutter, and it must be extremely fast, in order to allow classification of each possible window at frame rate. As an indication of the scale of the problem, for a 640×480 image with a 20×20 window moved two pixels on each test, at 25 different scales increasing by 10% at each step, a system must calculate a function $h : \Re^{400} \mapsto \{0, 1\}$ 349,740 times on every camera frame.

To solve the speed problem, the V&J detector is restricted to a class of filters that can be computed in constant time, regardless of the spatial extent of the filter, and is also arranged in a cascade that rapidly rejects most negative windows without much computation. The filters involve comparing sums of pixel intensities in rectangle-shaped regions of the image (see Figure 1(a) for example filters). To evaluate this filter, the sum of the image pixels falling within the black box is subtracted from the sum of the image pixels falling in the white box. Sums of image pixels within rectangles can be computed in constant time, regardless of the size of the rectangle, if an "integral image" [10] is precomputed, requiring just a few operations per pixel.

Filters like those in Figure 1 can be turned into classifiers if we apply a threshold and parity, for example, "if the difference in the sum of the pixels in the black box and white box is more (alternatively, less, according to the parity) than 20, then the image contains an example of our object." No single such classifier is good enough for any interesting object, but it is possible for multiple such "weak" classifiers to



(a)

(b)

Fig. 1. (a) Example V&J static filters. (b) Example VJ&S dynamic filters.

"vote" for whether the window contains an object. If enough of the weak classifiers vote for presence of the object, the classifier returns the value 1; otherwise it returns 0.

In the Viola and Jones scheme, many classifiers based on rectangle-feature filters are learned from a large set of examples using the AdaBoost algorithm [11]. AdaBoost is a greedy method for finding a good combination of weak classifiers. First one applies the weak learning algorithm to find the optimal filter for the set of training examples, using brute-force search over all possible filters if necessary. Then the training set is reweighted such that misclassified examples get bigger weights. Then a new classifier is trained on the reweighted examples. The final classifier is a weighted combination of the votes of the weak classifiers. Viola and Jones found that with a large training set (thousands of face and non-face images), AdaBoost could learn a very good detector containing 200 filters similar to those shown in Figure 1(a). But to speed up the amortized time to classify each image subwindow, they instead split the ensemble into several stages, rejecting windows early if they are extremely unlikely to contain the object of interest. This system has generated enormous interest in the the computer vision community. In fact, it has already been applied to hand gesture detection in high-resolution imagery [12].

Unfortunately, the V&J detector alone is not good enough for our particular application of detecting and tracking multiple hands in crowded scenes. Within the range of resolutions we are interested in, a hand could be simply a blob, five pixels or so wide. A local, static detector would produce far too many false positives to be of direct use. To find and track hands reliably with a low false positive rate, we need more context, either spatial or temporal. Viola, Jones, and Snow [9] have recently extended the V&J detector into the time domain, and we thought the new algorithm was a promising candidate for our hand detector.

The VJ&S dynamic detector uses the same techniques as the V&J detector, but in addition to static "appearance" filters applied directly to an image, a few *difference images* are also considered as candidates for filtering. The AdaBoost-based training procedure is now allowed to choose not only filters applied directly to $\mathcal{I}_t$ (the original image at time t) but also to five difference images (as listed in [9]):

$$\begin{aligned} \Delta_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1}) \\ U_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \uparrow) \end{aligned}$$

$$
\begin{aligned}
L_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \leftarrow) \\
R_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \rightarrow) \\
D_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \downarrow)
\end{aligned}
$$

where the arrows indicate a shift of one pixel up, down, left, or right. This means a classifier could be built with, say, a rectangle feature in the difference image Δ to indicate a region of motion, or a rectangle feature with a low threshold specifying a lack of a response in the U image to specify an object apparently moving upwards. To further improve the ability of the weak learners to utilize these difference images, Viola, Jones, and Snow add filters like the ones shown in Figure 1(b) to the set of candidates for the weak learning algorithm. The additional motion information is sufficient to learn an extremely high-accuracy detector for pedestrians in video sequences [9].

Now that we have described the VJ&S dynamic detector, in the next section we describe its application to human hand detection.

III. Hand Detection With VJ&S

We have built a prototype hand detection system based on the VJ&S dynamic detector and performed a preliminary study of its efficacy on a real-world data set. The system is comprised of two main software modules: the *training* system, which does its work off-line, and the *run-time* system, which attempts to detect hands in a live video stream using a classifier produced by the training system.

A. Training

The first step in training is to collect a large set of example image pairs containing hands in natural poses. We take a straightforward approach: simply capture video of everyday human activity with a digital camera, ideally in several locations under various lighting conditions. Once the video data is collected, a human operator must manually select regions containing human hands from the image sequence. As this is a labor-intensive process, and a large number of examples are needed for training, we have created a simple graphical tool to automate as much of the process as possible. The main criterion is that the exact same window in both images of the pair must contain the hand. Subject to this constraint, we then try to ensure that the hands are roughly centered in the selection window, and that the hand should take up around 60% of the pixel area within the selection window. Sometimes these constraints conflict, as when there is a large amount of motion between two successive frames; then the selection window tends to contain the hand at one extreme end in the first image, and the other extreme end in the second image.[1] We repeat this process for every visible hand approximately 5 or more pixels in width, for every image pair in the sequence. We do not allow image pairs to overlap.

The learning algorithm also requires a set of negative examples, i.e., windows in image pairs that do not contain hands. We create an initial set of negative examples by simply choosing random window locations and sizes from the training video sequence then manually verifying that randomly-chosen windows do not happen to contain hands.

[1]We plan to correct this problem with a faster frame rate in the future.

Algorithm TRAIN-CASCADE
- Given: $\mathcal{P}$, a set of positive examples
  - $\mathcal{N}_0$, an initial set of negative examples
- Returns: C, a cascade of ensemble classifiers
- $C \leftarrow H_0 \leftarrow$ ADABOOST($\mathcal{P}, \mathcal{N}_0$)
- $i \leftarrow 0$
- Repeat:
  - Test C on new, known-to-be-negative examples
  - $\mathcal{N}_i \leftarrow$ The top k false positives from test
  - $H_i \leftarrow$ ADABOOST($\mathcal{P}, \mathcal{N}_i$)
  - $C \leftarrow C$ with H_i appended to the cascade
  - $i \leftarrow i + 1$
- Until performance is "good enough"
- Return C

Fig. 2. Cascade training algorithm.

Algorithm ADABOOST
- Given: $\mathcal{P}$, a set of positive examples
  - $\mathcal{N}$, a set of negative examples
  - WEAKLEARN, a weak learning algorithm
- Returns: H, an ensemble of weak classifiers
- Initialize weights w_i for each example i uniformly
- For $t \leftarrow 1$ to T:
  - Normalize weights to sum to 0.5 for $\mathcal{P}$ and $\mathcal{N}$
  - $h_t \leftarrow$ WEAKLEARN($\mathcal{P}, \mathcal{N}, \mathbf{w}$)
  - $\epsilon_t \leftarrow$ the weighted error for h_t given $\mathbf{w}$
  - $\beta_t \leftarrow \epsilon_t / (1 - \epsilon_t)$
  - For correctly classified examples i,
    - $w_i \leftarrow \beta_t w_i$
- Return H :

$$
H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}
$$

Fig. 3. AdaBoost learning algorithm [11].

Towards constructing an extremely efficient classifier that can be quickly evaluated on a given window in an image pair, following Viola, Jones, and Snow [9], we cascade several ensembles as described in Figure 2.

The result is a cascade C of ensemble classifiers H_i, each consisting of several weak classifiers based on rectangle filters. The training method means that we train a classifier on the initial training set, get a set of false positives from the trained classifier, then use those false positives to train the next stage of the cascade. Given a set of positive and negative examples, we train h_i using the AdaBoost algorithm, described in Figure 3.

The weights $\mathbf{w}$ indicate the importance of each training example. Early on in training (small t), simple individual weak classifiers will have fairly low error, making β_t small,

so that examples correctly classified early on will have less weight later, and therefore have less influence on subsequent calls to the weak learner. This means the later weak learners focus on the "hard" examples that are not correctly classified early on. The final result, H, is simply a weighted majority vote by each of the weak classifiers.

The weak learning algorithm, as mentioned before, is simply to try all possible rectangle filters and thresholds, and determine which combination of filter and threshold minimizes the weighted error.

B. Run-time

The run-time system is responsible for evaluating a classifier cascade on every possible window in a given image pair. We begin with the image pair at its original resolution, calculate the needed difference images and integral images, then slide the search window over the image. We first classify a given window according to the first ensemble in the classifier cascade, H_0. This entails calculating the prediction of each of the weak classifiers h_i in H_0 and weighting their votes.

Rather than use the vote threshold recommended by AdaBoost directly, however, following [9], we lower the threshold to meet a specific target false negative rate at each stage, thus ensuring that very few windows containing hands will be rejected prematurely, at the cost of accepting more potential false positives. If the candidate window location passes stage 0 of the cascade, we then evaluate ensemble $H_1, H_2, \ldots$ on the window until it is either rejected or accepted by every stage of the cascade. The run-time system records each location in the image predicted to be positive by the classifier cascade, eliminating any weak positives that are overlapped by stronger positives. Finally, when every possible window location at the current scale has been tested, we down-scale the entire image and sweep the window over the image again. Though the window is still the same size, this corresponds to a larger window in the original image pair.

Once every possible window location in the input image pair has been classified, the run-time system returns the list of potential hand locations to the running application.

IV. Experimental Results

We have implemented the system described in the previous section and performed a preliminary evaluation using a single training video sequence captured in a cafeteria at SIIT. Using an inexpensive IEEE 1394 digital camera, we captured 8-bit 640×480 grayscale video at 4 fps for 10 minutes. We paired consecutive frames of the sequence (1198 pairs of images) and manually selected the image regions containing suitable human hands as described in the previous section. We cropped the hand regions from each image pair and scaled the resulting small pairs to a size of 20×20 pixels. After exhaustively scanning the data set, we had 1060 pairs of 20×20 images containing human hands (many pairs contained several suitable locations, but many others contained no suitable locations). For the initial set of negative examples $\mathcal{N}_0$, we selected 1060 random patches random from the original video sequence, with width varying uniformly between 20 and 80 pixels.

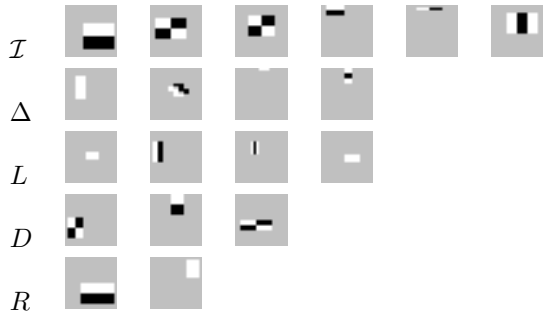


Fig. 4. Filters in the first-stage hand classifier.

From the 1060 negative and 1060 positive training examples, we reserved 10% (212 image pairs) for testing. The remaining 90% of the data was fed to the cascade learning algorithm (Figure 2). Our current cascade has depth three; for each stage, we trained an ensemble consisting of 20 weak classifiers using AdaBoost (Figure 3, $T = 20$), but then used the number of classifiers that produced the lowest ensemble error on the training set. We set the vote threshold for the first two stages to achieve a false negative rate of 0.05 on that stage's training set. The final classifier cascade achieves an accuracy of 92.1% correct on the original training set, 89.5% on the training set for stage 2, and 92.2% correct on stage 3's training set. The same classifier achieves an accuracy of 90.1% correct on the 212 test pairs we held out for testing. Figure 4 shows the filters selected by AdaBoost for the first-stage of the cascade.

After training the three-stage cascade, we evaluated it more thoroughly on ten pairs from the test set. These image pairs had never been seen during training (though of course the background was the same). The current implementation of the run-time system evaluates the classifier cascade C on 349,740 windows, from a size of 20×20 to (nearly) the entire image. Unfortunately, since our current classifier is only approximately 90% accurate on its test set, we can guess that it will have a high false positive rate (indeed we can guess that if the test set is representative of all windows in the image, we should get on the order of 35,000 false positives). The situation is not that severe, however, because, as described earlier, when two overlapping regions at the same scale are positive according to the classifier, we only retain the window with the highest confidence. Figure 5 shows our preliminary results from the run-time system on two image pairs in the test sequence. We found that the detection rate for the current cascade is too high, so for purposes of display, for each test pair, we simply selected the top 10 positive regions. Clearly, the false positive rate is too high for the current system to be used directly, but even so, it is usually apparent why the system makes the mistakes it does. In the next section, we detail how we plan to improve the system in future research.

Fig. 5. Preliminary results on test data from the hand detector. Each row shows a consecutive pair of test images, with the top 10 most likely hands highlighted with boxes.

V. Conclusion

We have only begun to tackle a very difficult computer vision problem. Though these preliminary results are encouraging, currently, the hand detection system makes far too many mistakes to be of practical use. There are four main factors contributing to the mistakes:

1) Our current cascade contains fewer than 60 very simplistic filters and is being asked to perform an extremely difficult discrimination task. The detector simply needs to be trained much longer to achieve better training set performance with a richer representation of the appearance and motion of human hands.
2) At four fps, we observe too much movement of the hands in consecutive frames. Selecting a large enough region to encompass both appearances of the hand, followed by down-scaling to 20×20, often shrinks the hands to unnecessarily small sizes and introduces too much variability, since the hands end up at opposite extremes of the crop window.
3) A first-rate detector will require a much larger training set. Face detectors like the V&J detector require thousands of examples to robustly detect human faces, which have a much more regular structured than human hands in arbitrary poses. Compared to the standards of face detection research, then, our training set is rather small.
4) In many cases, it might be practically impossible to distinguish a 20×20 grayscale image containing a human hand from other similarly blob-shaped, moving objects. Particularly in light of factor 2 above, we believe a human observer would also have a great deal of difficulty with the classification task.

The first problem is a simple matter to remedy; we are currently optimizing the training process and will explore the possibility of parallelizing the training software to run on a workstation cluster. We will also automate the selection of false positives for training the successive stages of the classifier; currently the false positive selection process is manual, making human intervention necessary between the training of each stage.

The second problem is also simple to solve with a faster frame rate. We need not run the full detection algorithm at high frame rates (4–10 Hz should be sufficient for most applications), but a 30 fps capture rate would mean less hand travel between frames and make the training system's job easier.

The third problem, likewise, can be solved with some effort. We plan to acquire several video sequences under

a variety of conditions in order to improve the classifier's robustness to noise. It may also be possible to generate new synthetic training data from the existing data by adding small random rotation, translation, and scale transforms to the data and including the transformed items in the training set.

The only potentially serious problem is the last problem. To overcome any inherent ambiguity in the stimulus, we may have to add additional information, such as color, or make the set of possible filters richer, or use higher resolution imagery, or even try to find the humans *before* finding their hands. These solutions, however, will all increase the system's training time, run time, and/or complexity.

Our ultimate long-term goal is to deploy the system as part of a mobile robotics application. Once we have the system performing robustly with a fixed camera, we plan to explore ways to achieve similar results with a moving platform. The current approach obviously will not work, since the difference images rely on a non-moving background. It might be possible to correct for camera motion prior to detection, but a more practical approach would be to have the robot stay still long enough for an initial detection of the hands in the scene, then use more dynamic techniques like tracking and active vision while moving.

ACKNOWLEDGMENTS

We thank the students of SIIT for being our test subjects. Cristopher Voldum and Yang-Cheng Fan helped with the software development for this project. MND is supported by Thailand Research Fund grant MRG4780209.

REFERENCES

[1] M. Becker, E. Kefalea, E. Maël, C. von der Malsburg, M. Pagel, J. Triesch, J. C. Vorbrüggen, R. P. Würtz, and S. Zadel, "GripSee: A gesture-controlled robot for object perception and manipulation," *Autonomous Robots*, vol. 6, pp. 203–221, 1999.

[2] C. Shan, Y. Wei, T. Tan, and F. Ojardias, "Real time hand tracking by combining particle filtering and mean shift," in *IEEE International Conference on Face and Gesture Recognition*, 2004, pp. 669–674.

[3] J. M. Rehg and T. Kanade, "Visual tracking of high DOF articulated structures: An application to human hand tracking," in *Third European Conference on Computer Vision*, 1994, pp. 35–46.

[4] S. Ahmad, "A usable real-time 3D hand tracker," in *Proceedings of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, 1995, pp. 1257–1261.

[5] J. Segen and S. Kumar, "Human-computer interaction using gesture recognition and 3D hand tracking," in *Proceedings of the IEEE International Conference on Image Processing*, 1998, pp. 188–192.

[6] J. Triesch and C. von der Malsburg, "A system for person-independent hand posture recognition against complex backgrounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 12, 2001.

[7] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[8] R. Polana and R. Nelson, "Low level recognition of human motion (Or how to get your man without finding his body parts)," in *Proceedings of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, 1994, pp. 77–82.

[9] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 2003, pp. 734–741.

[10] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[11] Y. Freund and R. E. Shapire, "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 5, no. 1, pp. 119–139, 1997.

[12] M. Kölsch and M. Turk, "Robust hand detection," in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.