Abstract— in this article I present a method to use Viola and Jones algorithm for hand gesture recognition. A training method and an algorithm to choose the most likely gesture have been suggested. Tests under low cluttered environment show good detection rate.

Keywords-; Hand Gesture, Viola and Jones

I. INTRODUCTION

Computer recognition of hand gestures can provide more intuitive user machine interface and can be useful for wide range of applications [1]. Figure 1.A, illustrates such a system, a laptop PC with a built in camera that points to the left side of the keyboard enables the user to communicate through hand gestures. In this article I suggest a method to detect four static postures (see Figure 1.B) using a modified version of the popular face detection algorithm introduced by Viola and Jones. [2]

Mapping Viola and Jones face detector to hand gestures has introduced a few challenges. (1) Hand gestures are more difficult to characterize than face due to finer grain details. (2) A set of two gestures has greater similarity than a face and non-face object (3) A single hand gesture may have many different postures that can be perceived as the same gesture, unlike Viola and Jones face decoder that is limited to face looking.
straight ahead. To simplify, I assumed that only the four hand postures I aimed to detect can appear on the scene. This simplification can be justified by the close environment in which the detection will take place i.e. hands in front of a computer, and by the fact that detecting a hand object from non hand object can be solved orthogonally. In section II, Training, I elaborate on challenges (1), (2) and (3), and proposed a method to mitigate them.

II. Training

A. Posture (shapes) Selection

Four postures have been selected (Figure 1.B). Note that each posture can be satisfied by different hand positions, i.e. right posture can be obtained with tilting the hand in different angels, while spread posture can be obtained with slight different distance between the fingers. Selection of the posture was based partially on work done by [6] which shows that Flat and Spread postures has better suitability for classification.

B. Setup

All hand images were taken from “Cambridge Hand Gesture Data set” [3]. This database provided a generous hand posture collection taken in controlled manner. Eight pictures from each posture used as an input to train the decoders (8x4=32), and eight different pictures from each posture were used to test the decoders (32).

C. Image Preperation

The number of features, hence the training time, is quadratic with the size of the window. Figure 3-left shows a very poor quality image when a window on 20x20 is taken. This is in contrast to face detection in which 20x20 window gives sufficient quality for detection. As mentioned in the introduction this is a result of the finer grain details that are needed to classify the hand. I have chosen to crop the images first, and increase the window size from 20x20 to 39x39, Figure 3-middle. I compensated for the increase in the window size by skipping every other pixel when sliding the specified features on the y direction as the gradient on the Y is lower.

D. Training Scheme

As mentioned in the introduction one of the main challenges for accurate detection is the similarity between the postures and the large variation within a posture. Three images from the test set1 demonstrate the large variation issue Figure 4. On the left a Spread image that is slightly tilted to the right. On the middle another Spread hand only tilted to the left. At the right side there is a left posture with one of the finger open like spread posture.

In the proposed method each posture is trained separately “against” only one posture at a time. For example the Spread posture can be trained separately “against” flat hand, right hand and left hand. The following is explicit list of all 12 decoder and their short notation.

<table>
<thead>
<tr>
<th>Decoder Name</th>
<th>Positive Images</th>
<th>Negative Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>Flat</td>
<td>Spread</td>
</tr>
<tr>
<td>FL</td>
<td>Flat</td>
<td>Left</td>
</tr>
<tr>
<td>FR</td>
<td>Flat</td>
<td>Right</td>
</tr>
<tr>
<td>SF</td>
<td>Spread</td>
<td>Flat</td>
</tr>
<tr>
<td>SL</td>
<td>Spread</td>
<td>Left</td>
</tr>
<tr>
<td>SR</td>
<td>Spread</td>
<td>Right</td>
</tr>
<tr>
<td>LF</td>
<td>Left</td>
<td>Flat</td>
</tr>
<tr>
<td>LS</td>
<td>Left</td>
<td>Spread</td>
</tr>
<tr>
<td>LR</td>
<td>Left</td>
<td>Right</td>
</tr>
<tr>
<td>RF</td>
<td>Right</td>
<td>Flat</td>
</tr>
<tr>
<td>RS</td>
<td>Right</td>
<td>Spread</td>
</tr>
<tr>
<td>RL</td>
<td>Right</td>
<td>Left</td>
</tr>
</tbody>
</table>
Given this method, a detection of spread hand that is slightly tilted to the right will give a positive result in SR detector and will be qualified as Spread. Figure 5 summary of the training flow.

**Figure 4: 3 Test images demonstrates cross posture similarity**

**Figure 5: Training Scheme**

III. **Decoding Algorithm**

When detecting using Viola and Jones, we check if there is a face in certain window. To do so groups of classifiers are sent one at a time. If a classifier in the group did not satisfy the trained threshold the window location is then rejected. If all the classifiers in the group have satisfied the threshold the next group is sent. When all groups meet the threshold the window is detected as a face. When mapping Viola and Jones decode with a small number of classifiers more than one posture was detected, while using a large set of classifiers none of the postures were detected. To resolve it, I broke the search into rounds. For each round a group of classifiers ran against each of the twelve decoders. For each decoder I found the window that satisfied the largest number of classifiers.

Decoders that satisfied only a small number of classifiers were discarded. After a few rounds, if decoders associated with more than one posture were left, I selected the posture associated with the decoder with the largest accumulated classifiers match.

Figure 6 shows another key insight. Since the sliding window (39x39) contains only part of the hand, different decoders could have best matched different parts of the hand. What I have noticed is that large location deviation between the three decoders associated with the same posture is a good predictor that this posture could be excluded. Excluding posture based on location simplifies the detection by two folds. First, we can exclude decoder early on, improving compute time, and second we can lock the search window after a few rounds which further reduce the process time.
After some experiments I found out that the following parameters work best.
Classifier per Round = 100
Max number of Rounds = 10
Lock Window = after 3 rounds
STD to reject a posture based on location = 15
Outlier excluded = 30% less than max

More of the lower level details of selection between the 12 decoders are embedded in GestureDecoder.m file.

IV. RESULTS

Two sets of tests have ran (1) 20 pictures under the same lighting condition as the trained images, Figure 7. (2) 12 pictures with a new set of lighting condition, which the decoder was not trained for. All ran under the parameters specified in the previous section. In two tables below Figures 7 and 8, I color coded in green pictures that detected correctly and in red pictures that did not. Overall the detection rate is 80% for set one and 50% for set 2. Further optimization that was not captured in this article increased the detection rate to 90% for set 1. From the results it can be concluded that right and spread postures should be better trained.

V. FUTURE WORK

(1) Algorithm speed
a. A better way to reduce the number of classifiers needs to be developed ~100 classifiers that are used now would not satisfy real time requirement
b. Search algorithm is written in naïve way and would need to be adjusted using methods that were mentioned in ee368 class i.e diamond search could work fairly well here

(2) Algorithm robustness
a. Training would be needed to extend from 16 images to >1000 images.
b. “Inverse response”

It seems reasonable to expect that the response ratio of the inverse decoders i.e. FS <-> SF to the same window would be very low at the right location. If this is exploited it will increase the robustness of the system.
Figure 7: Test Set1 Images, and the detected feature