Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

# Finger Recognition for Hand Pose Determination

J. R. Parker
Faculty of Fine Arts
University of Calgary
Calgary, Canada
jparker@ucalgary.ca

Mark Baumback
Amazon.com
Ste 1200, 1200 12th Ave S
Seattle, WA
mark@baumback.com

***Abstract*** - **Hand pose and gesture recognition methods have been based on many properties of hand images, but using finger recognition is unique. Here we describe hand pose from finger recognition, its limitations and its advantages. The implementation is being used for some real applications, including interfaces and computer games, and has a success rate of over 98%**

***Keywords*** **- gesture, hand pose, vision, image processing.**

## I. Introduction

Vision could be a key element of computer games and other user interfaces if only computers could actually *see*. A significant portion of human communication consists of what we call body language, and this includes facial expressions and hand gestures. Clearly, a more *natural* user interface would allow such communication, and providing natural interfaces is one of the goals of computer vision. Natural interfaces are those that permit the user to communicate with software in a manner natural or traditional for the activity being undertaken.

In applications that use hand gesture input, the user's hands are be used to direct the focus in the scene, requiring that the position and pose of the hands be determined; for example, the user can point to something, which can be described in words using three dimensional computer audio. Changing the hand posture could indicate that some activity should take place. A gesture, in fact, can be defined as a sequence of hand postures, or a change from one posture to another. This frequently translates to a very few simple mouse actions or menu selections, in the context of a user interface.

However, by any standards, hand poses and motions are gestures, and are a natural way to manipulate real objects and virtual ones. There are a vast number of applications that could use simple gestures to communicate intention to the computer, if this could be made fast and inexpensive. The system described here is both of these things, needing only a basic color camera and a simple video grabber, or just a USB camera

## II. Previous Work

Researchers in Human-Computer Interaction have been interested in gestures as computer input for some time. The idea that the human hand can be used as a mouse, for example, is recent, but not utterly new [5, 12]. However, many actual implementations require some fairly sophisticated devices to make this work properly [4]. Stark [10] created a system that recognized eight hand poses out of 24 that are claimed to be anatomically possible. These were used, first as a 3D input device to a visualization program, and then in a presentation system that permits a speaker to control a computer display while speaking, using gestures. Success rates were always less than 93%.

Specific work on gesture and hand pose recognition is frequently conducted by vision researchers, and focuses on the technical details of the problem. Techniques for recognizing pose vary from silhouette matching [9] to template/similarity matching and table lookup [1]. While interpretation of pose and gestures for sign language interpretation and on video can be allowed to consume computational resources, an application as a computer interface must operate at a high rate of speed or it becomes frustrating. Most published work does not include performance statistics, making comparisons quite difficult.

What is being proposed here is the use of finger recognition to determine hand pose. This appears to be a new idea - finger*tip* detection has been tried, but the recognition - the detection and labeling of an entire finger - seems not to have been successfully used in hand pose determination.

This work will deal almost exclusively with the problem of finger recognition, and its use in hand pose determination. One of the other problems, that of segmentation, has been dealt with in a previous work [8]. This work segmented color images into skin and non-skin regions. The skin regions, in our applications, represented human hands. The skin segmenter gave a very high rate of success; it used multiple algorithms, and was fast enough to be used in a real-time user interface using hand pose. This segmentation scheme is used in the current work as well. The overall technique achieves a high degree of accuracy over a large number of poses with complex backgrounds, and due to the fact that our implementation runs at usable (interactive) speeds. The key is the finger recognition aspect.

### *Finger Properties*

In order to recognize each finger we must exploit geometric properties of fingers in hand images. Looking from the center of the hand, fingers will be relatively long and thin in comparison to both the forearm and sides of the hand. This property can be exploited in order to find fingers by looking at the distances from the center of the palm to the boundary of the object - the *signature distance*. On a hand with up to five distinct fingers, the signature distance along the side of a finger must steadily increase until a maximum is reached around the fingertip. Each fingertip will be a local maximum in a graph containing distances to each of the boundary pixels, and so fingers can be detected by selecting each local maximum in an angle distance signature [7] originating at the center of the palm. In [11] up to five of these local maxima are selected and used to classify the hand into one of fourteen different hand postures, as well as determining the location of each fingertip.

While it is true that each finger will be a local maximum in the angle distance signature, the converse that all peaks will represent fingers is false. Poor hand segmentation, or the existence of a forearm will also cause spurious peaks within this graph. To avoid the possibility of falsely classifying fingers, based solely on size, the coordinates of each local maximum is selected and later verified by using a fingertip detection routine [6]. Increasing the number of possible fingers practically eliminates the potential for falsely rejecting fingers, while increasing the likely-hood of false acceptances. These false acceptances are later removed by verifying that both their size and shape is correct relative to the hand.

The process of verifying fingers starts with being given a pixel coordinate for each of the candidate fingers; this is used to define a small search area that contains the upper part of the finger. A simple fingertip template is then used to match the object pixels within this search area to verify that they have the properties of a finger.

As described in [6] fingertip shapes have two overall properties. A fingertip is defined by a small circle of object pixels having a diameter of the width of the finger. Along a square outside of the inner circle there is a long chain of non-object pixels and a shorter chain of object pixels. The fingertip template can be defined by the following parameters:

| | |
|---|---|
| $d_1$ | Diameter of the little finger ([5..10] Pixels) |
| $d_2$ | Diameter of the thumb( $1.5*d_1$ ) |
| $d_3$ | Size of the search square( $d_1$+N \| N >= 2 |
| **min_pixels** | Minimum number of filled pixels along the search square (width of the smallest finger : $d_1$) |

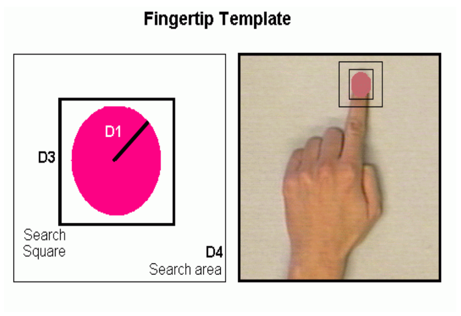

Figure 1: Basic Fingertip template.

**max_pixels** Maximum number of filled pixels along the search square(width of twice the thumb: $2*d_2$)

The template consists of a small circle with a diameter of $d_1$ or $d_2$, surrounded by a small search square with size $d_3$. (Figure 1) To have a successful match of the template at an object pixel *(x,y)* the object must pass the following constraints:

1. have a sufficiently sized circle of object pixels in a neighborhood of the position *(x,y)* ;
2. have a number of object pixels along the described search square between min_pixels and max_pixels around *(x,y)* ;
3. have one connected chain of object pixels along the search square;
4. be a minimum distance away from other fingertips.

The template, to be effective, must verify the correct shape and proportions of the fingertips, while being invariant to position, scale and orientation of the hand. Constraint 1 ensures that the potential fingertip has the correct size and proportions, while the correct shape of the fingertip is verified by constraints 2 and 3. In it's current state the method is rotationally and positionally invariant, efficient and quite effective at removing non-finger objects. However, the template is not scale invariant as it requires prior knowledge of the working environment in order to specify the size of a fingertip. This also implies a specific camera geometry relative to the target hand; when the size of the fingers are unknown or can change, the parameters of the template must be able to adjust in size relative to the hand.

### *Improved Fingertip Templates*

The template can become scale-invariant by estimating the two parameters d1 and d2, which together define the size of the template, from the size of the hand. The size of a fingertip (*d1*) is estimated from the size of the hand as:

***$d_1$*** Diameter of a fingertip (palm_circle_radius/8)

The palm circle radius requires some explanation; to find the palm, a distance transform is computed over the binary image resulting from the skin tone segmentation [8]. For each pixel P(x,y) the Euclidean distance ED(x,y) is estimated between it and the nearest non-skin pixel. The distance transform used is a fast version, linear in the number of pixels in the image. The palm center is the largest value in the distance transform, with a possible exception of the upper part of the forearm, and will define the radius of a circle encompassing the palm.

The value $d_1$ provides a rough estimate to the size of a fingertip, but is susceptible to segmentation errors. To compensate for this variance the second template size ($d_2$) is removed, the first two constraints are relaxed, and the third constraint is removed entirely.

Constraint 1, which ensures that the fingertip fills enough of the template, is lowered to compensate for $d_1$ being estimated too large. Constraint 2, which ensures that the fingertip is connected to a finger, is lowered in case the size of the search square is estimated to be too small. Constraint 3, which ensures that the fingertip is connected to only one finger is too restrictive when the size of the search square is slightly too small and is removed.

The fingertip template is no longer attempted at every object 3 pixel within the image. Instead a new constraint $d_4$ is introduced, which defines a small search area within which to match the template:

***$d_4$*** Size of the square search area ( $N*d_1|d_4>d_3$)

This search area is centered around each pixel location defined by the local maxima of the angle distance signature calculated above. Only object pixels within this square are matched against the template instead of the entire image.

The final stage of the template matching algorithm requires finding an *(x,y)* pixel coordinate that best represents the location of the fingertip. The fourth constraint, which defines the minimum distance between fingertips, selects the first pixel location that passes the template. However, if we define the best fingertip location to be one that is correctly centered on the fingertip and that is minimally affected by segmentation variances then the first pixel isn't the best choice. Instead, constraint 4 must be altered so that it can not reject possible fingertip locations.

By relaxing constraints 1 (sufficient number of object pixels) and 2 (correct ratio of object/non-object pixels) and removing constraints 3 (one chain of object pixels along search square) and 4 (minimum distance between fingertips) it is not surprising that a large number of fingertip locations pass successfully through the fingertip template. The last constraint, which prevents multiple fingertip locations for one finger, is modified slightly into averaging the pixel locations that pass the template around a given fingertip. This minimizes the effect of slight segmentation errors and provides a more stable fingertip location.

### *Standard hand position: knuckle detection*

The location of the extended fingers relative to the hand, being the most distinctive property between the different hand postures, should be the basis of any pose recognition algorithm. However, orientation of the fingers allows the location of different fingers to occupy the same space, and thus causes an overlap between the different hand postures. To ensure the classification algorithm is invariant to rotation, it not only has to deal with the orientation of the hand, but

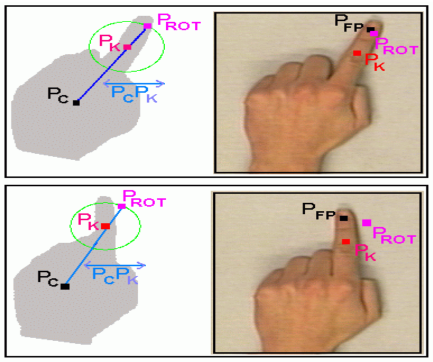

Figure 2: Fingertips rotated about the estimated knuckle poition. Top: Standard position. Bottom: Rotated.

also the orientation of each finger. Since the fingertip is the most pronounced part of the finger, which can be found reliably while being robust to segmentation variances, it is used as the main reference point for feature extraction. It is however, rotated to a standard position to remove any overlap between the classes. Figure 2 shows the fingertip locations rotated about the knuckle.

To rotate the finger into a standard position, the knuckle, which is center of rotation, must first be found. To estimate the knuckle, the angle of the finger is calculated to define a direction vector that indicates the direction from the fingertip to its base. The knuckle point $P_K$ is then estimated by moving along the finger by a length estimated from the size of the hand.

To determine the angle of the finger the top part of the finger is separated from the image. A bounding box of width 3*max(4, *fingerwidth*) is centered around the known fingertip location, where *fingerwidth* is estimated to be the value at the fingertip location in a distance transform of the hand. The object pixels within this bounding box are grouped into connected regions, removing any extra regions by selecting the largest one. The last region is roughly the top quarter of a finger and has the following properties:

1. The finger will intersect with the bounding box.
2. The fingertip will not intersect with the boundary of the bounding box

A line that best estimates the angle of the finger is defined as one that:

- *Starts at a pixel on the boundary of the object*
- *Ends at a pixel on the boundary of the bounding box*
- *Passes through the centroid of the object*
- *Maximizes the sum of distances counted for each pixel along the line.*

Finally, the location of the knuckle is estimated by moving from the fingertip in the direction of the vector calculated above. The pixel that is 35% of the distance between the fingertip and the centroid of the hand is selected as the location of knuckle. This distance is roughly half way down the finger and is obviously not the correct location about which to rotate the finger. However, the rotated finger point doesn't necessarily have to be anatomically correct, it only has to be co sistent and unique for each finger. The further the distance is away from the fingertip, the more of an effect a slight error in the angle will have, and thus the point half way down the finger is chosen as a trade-off between utility and correctness. A standard finger position is defined by a line between the centroid of the hand $P_C$, the estimated knuckle of the finger $P_K$ and the fingertip $P_{FT}$. The fingertip point $P_{FP}$ is rotated around it's knuckle position $P_K$ until $P_{FP}$ is on the line segment. The hand is defined to be in a standard position if all of the 4 extended fingers are in a standard position.

### *Hand posture classification*

The 32 templates, one for each hand posture, use two different features to classify the hand. The first feature, the physical part, reduces the search space by determining how physically possible it is that the hand matches the template. It labels each of the extended fingers and determines the orientation of the hand.

The second feature, the shape part, determines how well the hand looks like the template, and is used to classify the hand into one of the remaining templates.

So, each template consists of two parts:

**The physical part:**

*NF* - The number of extended fingers

Labels for each extended finger [Thumb, Index, Middle, Ring, Pinky]

*FA* - The angles of each extended finger

$D_{Finger}$ - The fitness value of the finger angles

$D_{Expected}$ - Fitness value of finger angles applied to the each specific template.

**The shape part**:

[$SC_T$,$SC_I$,$SC_M$,$SC_R$,$SC_P$] - The set of shape descriptors for each finger.

$D_{Shape}$ - The fitness of the individual shape descriptors

$D_{Average\ Shape}$ - The fitness of the averaged shape descriptors.

### *The Physical Part*

The set of hand postures is dependent on the number of extended fingers and can be divided into six different groups. The six groups, which are separated by the number of extended fingers NF (0,1,2,3,4,5), each contain 5!/((5-NF)NF!) templates for a total of 32. One of these groups is selected by counting the number of fingers detected in the fingertip detection stage. This set of templates is used to compare against the hand, where each template contains the same number of extended fingers but has a distinct set of fingers.

The selected set of hand postures can be further refined by removing postures that are impossible based off of the current hand position. Even though the hand is a highly deformable object, it is limited by a well defined set of rules. Enforcing that the hand stays flat facing the camera, the location of the fingertips relative to each other is restricted.

The location of each fingertip is measured by its angle on a plane centered at the centroid of the hand. Due to the properties of a hand there exists a minimum and maximum threshold that define the range of angles between each pair of fingers. The minimum threshold is limited by how spread out two adjacent fingers can be such that the fingertips are not touching. This value is dependent on how well the hand is segmented from the background, but is consistent between different people. The maximum threshold is defined by how far two adjacent fingers can be spread away from each other, and is much harder to define then the minimum threshold. It is dependent upon the user of the system as opposed to the quality of the image.

These thresholds, while not actually defined, are used to eliminate hand postures that are not possible. Each template contains the angles of their extended fingertips calculated on a known hand posture. These angles, after being quantized into the range of [0..90], are aligned with the fingertips of an unknown hand. The difference between the angles of these two sets, called $D_{Finger}$, is thresholded to eliminate unlikely postures.

Each of the templates in the list generated above contains the set of finger angles that match their specific combination of extended fingers. To compare the template to the hand, the finger angles of the template and the rotated finger angles (*RFA*) of the hand are sorted into ascending order. The distance between these two sets is defined as the sum of the squared distances:

$$D_{finger} = \sum_{k=1}^{NF} (FA_k - RFA_k)^2$$

The finger angles, and thus $D_{Finger}$, are invariant to position and scale but are dependent on the orientation of hand. As the hand rotates, the set RFA will shift either up or down within the range of [0..90] depending on the direction of rotation. If the hand rotates enough, so that the angles wrap around, the order of RFA will also change. To determine the orientation of the hand, and thus match each finger with one in the template, the set of finger angles is translated and shifted until the lowest $D_{Finger}$ value is found. The finger angle set is translated by aligning the lowest FA angle with the lowest angle in RFA. The set FA is then circularly shifted, each time aligning itself with the RFA set.

The orientation of the FA set that produced the minimum $D_{Finger}$ value is chosen as the orientation of the hand and thus defines a 1-1 match between the fingers of the hand and fingers of the template. $D_{Finger}$ provides information about how physically possible it is that the current hand matches the templates. However, it does not provide a high level of distinction between the templates with similar finger arrangements. Thus, this idea is further developed by specifying an expected finger difference $D_{Expected}$. This value is calculated by assuming that there is an expected difference of 24 degrees between each pair of adjacent fingers, and 44 degrees between the finger and thumb.

The orientation of the hand has already been estimated, and thus there exists a one-to-one correspondence between the actual fingers and those in each template. $D_{Expected}$ is calculated as the sum of the squared differences between the expected difference between each finger angle and the actual difference in their angles.

### The Shape Template

The finger distance feature provides a good distinction between hand postures that are unique within the set of the 32 postures. An example would be one that has 2 extended fingers, namely the thumb and the pinky finger. Although there are 10 different hand postures that could match the hand, the total distance between these two angles (thumb and pinky) is so great that a majority of the templates are easily removed by thresholding $D_{Finger}$. On the other hand, the $D_{Finger}$ distance tends to fail on hand postures that do not have unique finger positions. An example would be one with two adjacent extended fingers, namely the middle and the index finger. This posture, in terms of the finger-angle distance, is not unique because every other pair of adjacent fingers (except for possibly the thumb/index pair) have similar distances. $D_{Finger}$ degrades even more to the degenerative case of only one extended finger. In this situation, due to the rotation adjustments, each of the five templates all have the same $D_{Finger}$ value of zero, which provides no distinction between the templates.

Even though the finger-angle distance does not work in all cases, it provides a way to refine and sort the set of possible templates. For the small subset of hand postures that are unique with respect to their finger arrangements only one template will remain after thresholding and no further classification is needed. The others need to be further verified by using the set of shape descriptors from the shape part of the template.

### 1) Shape Features

The second feature, $D_{Shape}$, analyzes the shape of each finger to classify the hand into one of the remaining postures. Instead of looking at the physical relationships between the fingers, it analysis the shape of the hand by comparing sets of shape contexts.

A descriptive representation of an object can be formed by the set of vectors between each boundary point $P_i$ and the remaining N-1 points along the boundary. The complete list of sets, where each set represents the list of vectors between it and every other boundary pixel, is a complete and rich descriptor for a given object. As described in [2,3], this list can be condensed into a coarse histogram from a set of sampled boundary pixels. A histogram between a reference point $P_i$ and the remaining N-1 boundary pixels is defined as

$$H_i(k) = \#(q \neq P_i : ((q - P_i) \in \text{bin}(k)))$$

This histogram, which is the shape context of $P_i$, represents the shape of the object from the point of view of the point $P_i$. If each boundary pixel is selected, there will be N different shape histograms all representing the object from slightly different points of view. However, the complete set of boundary pixels isn't necessary to accurately represent the entire shape and thus a subset can be selected. This set of shape contexts $SC_A$ is used to represent an object A.

### 1) The Shape Histogram and Invariance

The size and quantity of the bins within the histogram, and the coordinate system used for the vectors between the pairs of points determine how the histogram will represent an object. The histogram is divided into five even bins representing the angle component of the vector and 12 bins for the distance component. Log-polar vectors are used in [2,3] based off the assumption that points closer to the reference point should have a greater effect on the shape context of $P_i$.

For the shape histograms to be successful at determining the similarity between planar objects they must be invariant to position, scale and orientation. The position of the object is automatically compensated for by selecting the set of shape descriptors from points along the boundary of the object. The scale of the object is affected by how the distance bins of the histogram are mapped onto the object. In [2] the radial distances are normalized by the mean distance between the $n^2$ point pairs in the shape. The shape histograms, by definition, are dependent upon the orientation of the object. By using the log-polar coordinate system the angles are calculated based off of a normal oriented plane. The method suggested in [2,3] is to orient the plane around the tangent at the reference point of the object for each shape context.

### 1) Histogram Matching

Once the set SC of shape contexts is created it is necessary to evaluate how similar an object A is to another object B. More specifically, a correspondence problem exists that requires finding a unique match between each histogram $H_i$ in $SC_A$ and the corresponding histograms in $SC_B$. The cost of matching two histograms can measured using the $\chi^2$ test statistics:

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{K}\frac{[H_i(k) - H_j(k)]^2}{[H_i(k) - H_j(k)]}$$

To determine how similar two objects are the total cost between each pair of histograms is calculated as:

$$C_{AB} = \sum C_{ij}\Big| i \in SC_k,\ \ j \in SC_B$$

The minimum total cost $C_{AB}$, which has a 1-1 correspondence between the sets $SC_A$ and $SC_B$, is chosen to represent how well the two objects A and B matched.

### 1) Shape context of hands

There will inevitably be variations to the hand that will alter its appearance. The orientation of the hand and the forearm are both uncontrollable variables that must be considered. A hand, unlike many other objects, can rotate around two different focal points. Movements of the hand which slightly alter its orientation can be considered a translation and a rotation. Since the shape descriptor is positionally invariant, this becomes a simple rotation around the center of the palm. However, the hand can also rotate around its base, or more specifically then wrist. The forearm, which will remain stationary as the hand rotates around it, causes the sides of the hand around it to grow and shrink and thus altering its appearance. This changes the shape of the hand along with its position and orientation. All of these factors including some uncontrollable factors like noise and segmentation variances will effect a shape context of the object, and must therefore be considered.

The shape descriptor, when applied to hands, is used to determine the posture of the hand. One method would be to calculate a set of shape histograms *$SC_i$* for each of the possible hand postures, which could then be used as templates in a nearest neighbor classification approach. Another approach however, would be to classify each of the individual fingers of the hand, which could then be used to determine its posture.

### 1) Finger Classification Approach

The shape histogram matching algorithm applied to finger classification doesn't solve the correct problem. Instead of trying to determine what the object is, the problem becomes determining which finger is which. The question that needs to be answered becomes, "What does a specific finger look like?" Of course, almost all fingers look pretty much alike, and thus in order to distinguish specific fingers the correct question becomes "What does the hand look like from a specific finger?". Distinguishing between fingers of a hand is different than distinguishing between separate objects. Since each of the five fingers are part of the same object, several sets of shape contexts must be built for only one object. Each of these sets must contain enough information to distinguish it from every other set on the same object. The set of shape contexts SC cannot be the set of boundary pixels, as described above, because they will be the same for each finger. Instead, the set of shape contexts are unique for each fingertip, which is what provides the distinct representation of the hand for each fingertip.

The set of histograms built for a specific finger must represent enough information to distinguish it on a hand. The training of the shape histograms for a specific finger is done with only the specific finger showing. However, the same set of histograms must classify a finger with the existence of multiple fingers tips to recognize a large set of different hand postures. These extra fingers, when calculating the set of histograms, become outliers and will adversely effect the values within the histogram. Shape contexts, by definition, are not invariant to rotations.

The method suggested above relies on the accuracy of calculating a line tangent to the reference point of the shape context. This vector is not only easily susceptible to segmentation and noise variances, but is impossible to determine if the reference point is not on the boundary of the object. Since it is important that shape contexts for fingers be from unique and stable points of view, the rotated finger angle must be used for a descriptor. This point is not guaranteed to be on the object, and thus impossible to calculate a tangent vector for and thus a shape context.

A shape histogram has five different angle bins located around a reference point. The orientation of these bins causes the histogram to be susceptible to simple rotations because they are angles in the polar coordinate system. Instead of using the normal polar coordinate system to define the vector, as described above, another suggestion is: where θ is the interior angle between the reference point $P_i$, the boundary pixel $BP_j$, and the centroid of the object C; and ψ is the Euclidean distance between the point $P_i$, and the boundary pixel $BP_j$.

$$V = (\pm\theta, \psi)$$

The angle theta is defined as the interior angle θ in the triangle defined by the points ($P_i$, C, $BP_j$) (Figure 3). The range of theta is between [-π .. π], with a negative angle defined by which side the point $B_j$ is relative to the line $P_iC$. The addition of the negative angle creates a larger distinction between the number of boundary pixels on each side of the finger angle, which is the main characteristic between different fingers that the shape histogram is trying to exploit.

The scale of the object is compensated for by the distribution of the distance bins within the shape histogram. The method suggested above normalizes the distance component of a log-polar vector to distribute the bins over the object. With respect to hands however, the outlier pixels belonging to the unrotated finger, as well as the other possible fingertips are located close to the reference point. Because of this proximity the outliers are emphasized instead of suppressed and thus makes the log-polar vectors unsuitable. Instead, each of the five distance bins contain an even percentage of the total number of boundary pixels. This will distribute the information about the hand evenly across the entire histogram and minimize the total impact of the outliers.

As discussed above, the set of reference points for building the shape contexts must be unique for each finger in order to distinguish between them. Instead of sampling points along the boundary, the set of reference points used to build the shape histograms are selected from the location of the finger.

The larger the set of histograms built for each finger, the more information there will be to distinguish between different fingers. However, since each finger needs its own set of histograms, the larger the difference between the shape context sets of each finger the easier it will be to distinguish them. This becomes a trade off between selecting more histograms to represent the shape of a hand for a finger but losing variability between them. The reference points used for each finger follow this pattern:

1. The rotated finger point
2. The point opposite the rotated finger point on a line passing through the centroid of the object
3. The 2 boundary points on the line perpendicular to the rotated finger through the centroid of the object

This creates a set of perpendicular lines, (See: Figure 4) centered around the centroid of the object, defined by the angle of the rotated finger location. This set of cross-hairs can be further subdivided by selecting two more perpendicular lines that subdivide each of the 4 quadrants, which can be repeated by further subdividing each quadrant.

Each of the five fingers will follow this pattern having the same number of different histograms. Due to the slightly different angles of the fingers (relative to the centroid of the

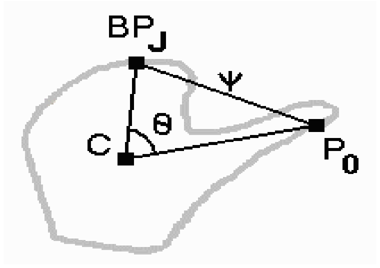

**Figure 3:** The two parameters of the vector used in building the histogram for one specific reference point and one specific boundary point.

hand) the set of reference points between the fingers will be simple rotations around the centroid. The more points selected for reference points the smaller the rotation angle will be between each finger set. Due to many different variations, including the estimated finger angle and the variations between segmented regions, these reference points will vary. The maximum amount of variation defines the minimum distance (or angle) between two sets of reference points for different fingers. Each template contains the sets of shape contexts belonging to the extended fingertips of the template. Each of the NF sets of shape contexts contain N different histograms depending on the number of reference points selected.

Once the shape contexts are built for each rotated finger location ($SC_1 \ldots SC_{NF}$) they can be compared to the set of shape contexts in the remaining templates. The order of the histograms within each set is known since they were specifically calculated and thus the correspondence problem does not exist. Instead, the similarity between the template and the hand is calculated by comparing the shape contexts of the template and the shape context of the each fingertip.

The first value $D_{Shape}$ is calculated by summing the total costs of each shape context of the template and the shape context of it's matching fingertip: This provides a number as to how well the shape contexts of the fingers matched those of the hand. However, since the values are summed together it may not best represent how well the two objects matched. As an example, one of the cost values may be large due to an incorrect rotated finger location but the cost values of the other fingers might still be correct. The second shape value attempts to compensate for these problems by averaging the two sets of shape contexts. $D_{Average.}$ Shape is calculated by comparing these two averaged shape histograms in a similar fashion as above.

After matching the physical part of the template to the hand, the orientation of the hand was estimated and thus each finger of the hand was matched to one in template. This configuration, for each of the templates, is used to determine which shape contexts in the template to match with those calculated for the fingers. These templates are used to calculate both $D_{Shape}$ and $D_{Average\ Shape}$, which combined with $D_{Finger}$ and $D_{Expected}$ provide all four features used to classify the hand posture.

If the number of fingers (NF) belonging to the hand is either 0 or 5, the selected set of templates will contain only 1 possibility. In the case of 0 fingers none of the features calculated above can be used to classify the hand and thus it is automatically selected as the current posture. However, a shape analysis approach as suggested in [8] could be used. If the number of fingers is 5, $D_{Finger}$ can be thresholded to ensure that it is indeed a hand.

For the remaining set of hand postures $D_{Finger}$ is used to threshold the set of templates by removing those that can not physically match the fingers of the hand. The remaining features $D_{Shape}$, $D_{Average\ Shape}$, and $D_{Expected}$ are then combined using a *Borda* count voting system to classify the hand image into one of the remaining postures.

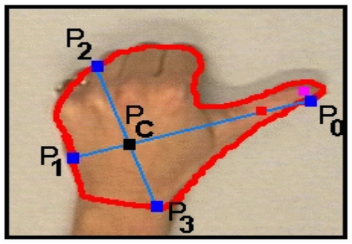

**Figure 4:** The four histogram reference points.

## III. Results and Conclusions

The system was trained to identify 32 hand postures, and the formal results presented here were acquired by creating 35 samples of each posture, for a total of 1120 postures, and performing a recognition on each one. Each number in the matrix is a percentage. Training was done using only a single template for each finger. The pose recognizer has also been used to create a gesture input for a solitaire game, with results having a very high reliability in practice. The overall success rate is 95.9%. There are a few poor performers - if three poses are removed, we improve the success rate to 98.3%. 17 poses are recognized perfectly in the trial. This is acceptable for most user interface applications that would use hand postures. The confusion matrix diagonal in Figure 5 is the result of this experiment.

## References

[1] Athsitsos and S Sclaroff, "3D Hand Pose Estimation by Finding Appearance Based Matches in a Large database of training Views", *Proc. IEEE Workshop on Cues in Communication*, Kauai, Hawaii, December 9, 2001.

[2] S. Belongie, J. Malik, and J. Puzicha. "Matching shapes", In *Proc 8th Int'l. Conf. Computer Vision*, July 2001.

[3] S. Belongie, J. Malik, and J. Puzicha. "Shape Matching and Object Recognition Using Shape Contexts", Technical Report UCB//CSD00-1128, UC Berkeley, Jan. 2001.

[4] Lars Bretzner and Tony Lindeberg, "Use Your Hand as a 3-D Mouse, or, Relative Orientation from Extended Sequences of Sparse Point and Line Correspondences Using the Affine Trifocal Tensor", *Proc. 5th European Conference on Computer Vision*, 1998.

[5] G. Baratoff and D. Searles, "Gesture Recognition", http://www.hitl.washington.edu/scivw/EVE/1.D.2.b.GestureRecognition.html.

[6] C. von Hardenberg and F. Berard, "Bare-Hand Human Computer Interaction", *Proc. ACM Workshop on Perceptive User Interfaces*, Orlando, Florida, Nov. 15-16, 2001.

[7] Parker, J.R., "Histogram Methods For Scientific Curve Classification", *SPIE Vision Geometry VI*, San Diego, July 28-29, 1997.

[8] Parker, J.R., Baumback, M., "Visual Hand Pose Identification for Intelligent User Interfaces", *Vision Interface 2003*, Halifax, Nova Scotia, Canada June 11-13, 2003.

[9] N. Shimada and Y. Shirai, "3D Hand Pose Estimation and Shape Model Refinement from a Monocular Image Sequence", *International Conference on Virtual Systems and Multimedia* '96, Gifu, Japan, Sept. 18-20, 1996.

[10] M. Stark and M. Kohler, "Video based gesture recognition for human computer interaction", in W. D.-Fellner (ed.), *Modeling - Virtual Worlds - Distributed Graphics*, November 1995.

[11] M. Takaoka, R. Abiko and R. Ishii, "A Hand Shape Recognition and its Application to a GUI support system for a sight handicapped person", *IECON 2000.*

[12] J. Triesch and C. von der Malsburg, "Robust classification of hand postures against complex backgrounds", *Proc. Second Int. Conference Automatic Face And Gesture Recognition*, Killington, VT. Oct 1996.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 100 | 100 | 97 | 60 | 100 | 97 | 100 | 83 | 91 | 97 |
| 10 | 91 | 100 | 100 | 100 | 100 | 100 | 100 | 91 | 100 | 94 |
| 20 | 97 | 100 | 97 | 100 | 80 | 97 | 100 | 97 | 97 | 100 |
| 30 | 100 | 100 | | | | | | | | |

**Figure 5:** Diagonal of the confusion matrix.