Visual Hand Pose Identification for Intelligent User Interfaces

J.R. Parker
Mark Baumback
Laboratory for Computer Vision
Department of Computer Science
University of Calgary

Abstract

We take a common application, the PC game of solitaire, and create an alternative input scheme using hand gestures, in particular those that would be used if playing the game with real cards. The alternative output scheme uses audio, in particular voice descriptions of cards and relative hand positions. The result is playable by anyone using any flat surface with no actual cards and no need to touch or look at the computer. A large class of interfaces can profit from these ideas, and visually impaired users have expanded access to existing systems. We focus on practical aspects of the gesture interface, which is purely vision based, and which has a high reliability.

1. Introduction

Vision remains the key element of games and of most other user interfaces - typically, one must be able to *see* in order to *play.* We propose to have the computer do much of the seeing, and the user's hands will be an input device. Natural interfaces are those that permit the user to communicate with software in a manner natural or traditional for the activity being undertaken. We are connecting a natural interface to the commonly played computer game *Solitaire*, converting it so that it can be played without actually touching the computer.

The player's hands are be used to direct the focus in the scene, requiring that the position of the hands be determined; the user can point to something, which can be described in words using three dimensional computer audio. The player's un-instrumented hands can point to cards and card stacks, which will be described to the player ("Ace of spaces, third stack from the left") by a human voice. Changing the hand posture from open to a fist will 'grab' a card or stack, as would occur when the mouse button is held down in the standard game, and will permit the user to move cards about the tableau. This amounts to a very few logical operations that correspond to simple mouse actions. Since the actual game of solitaire is played with the hands, moving real cards around a table, this game can be thought of as a middle ground between actual and computer solitaire. However, by any standards the hand poses and motions are gestures, and are a natural way to manipulate real playing cards. 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. We feel that our system is both of these things, needing only a basic color surveillance camera (about $300) and a simple video grabber (about $150).

2. Related Work

Much of the work on gesture recognition relates to sign language interpretation by machine. The interface as envisioned needs to determine the position of the hand on the table, relative to the virtual cards and stacks, and the hand pose or posture, that is: open or closed. It needs to keep track of the start and end position of the gesture in those circumstances where a card is being moved.

2. 1 Gestures in HCI

Of course, HCI researchers have been interested in gestures as computer input for some time. The idea that the human hand can be used as a mouse is recent, but not utterly new. However, most actual implementations require some fairly sophisticated devices to make this work properly [2]. 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 details of the problem. Techniques for recognizing pose vary from silhouette matching [9], slope/orientation histograms [4], and 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.

3. Project Overview

The first level of complexity of this project places the tableau on a flat, approximately horizontal surface. The virtual cards are dealt in a standard solitaire organization onto this surface. The user controls the activities by moving their hands - one gesture asks for information about the tableau, another asks to move a card or card stack. The cards are displayed on a standard video display, in addition to the new

audio display, where cards are read off to the player and positions of the hands are described verbally.

Input of hand locations and poses is accomplished using computer vision methods. A simple video camera is positioned above the table looking down, so that the hand is in clear view. A frame grabber captures video of the tableau, which is passed one frame at a time to the vision software for interpretation.

3. 1 Playing Solitaire Without Touching the Computer

The game tableau will be a virtual one, projected horizontally in front of the player onto a real surface. The surface needs to be larger than needed to play a real solitaire game - about 2x2 feet. The hand will move in front of the cards, which will cause them to be identified, and a grabbing gesture will attach the virtual cards to the hand, allowing the player to move cards about. Voices and effects will guide the player across the tableau, identify cards and stacks, and generally assist with navigation through the game. The sound will be three dimensional, allowing voices to appear to come from particular places in space.

A set of protocols are required if the game is to be played without using the mouse or the usual set of window and widget manipulations. What is required is the ability to move cards, to deal from the deck, to deal a new hand, and to back up a step - with no visual display, mistakes will be inevitable. Additionally, we need display mechanisms using audio that supplements the computer video screen display.

Figure 1 shows the interface pattern that will be used by our implementation. The act of moving a hand into (I.E. in front of) one of the indicated areas will result in a response by the game; this will be called a *touch*. There are also two hand poses recognized: *hand open* and *hand closed*. Each pose can be used to ask for a different response.

A hand open pose is a request for information. Whenever a face-up card is touched by an open hand, the card is displayed for the user - display is verbal, so the nature of the card is read out in words. The position of the hand is also described verbally, so that the user can orient themselves quickly when starting a game, and when returning to an interrupted game.

A closed hand pose is a request for action. Closing the hand over the deck results in a card being dealt. Closing the hand over a card in the stacks results in that card being *grabbed*, or attached to the hand so it can be moved. When the hand is opened the card is released - this is like holding down the mouse button when the cursor is over a card, when playing the standard computer solitaire game. Of course, if the card grabbed is not on the bottom of a stack, then it and all cards below it will be moved together.

The regions labelled ‘A’, ‘B’, ‘C’, and ‘D’ are used for non-play control of the game. It would be very easy to touch one of these regions by mistake, so touching twice is required to activate a control. Touching ‘A’ results in a the previous move being undone, and the game returned to the previous state. Touching ‘B’ results in a new game being dealt. Touching ‘D’ results in the game program being terminated. The ‘C’ region is currently unused.

An alternative to using physical regions for game control would be to use different hand poses for each request. This would be more difficult to implement, and possibly harder for the player to remember and enact, but there is no reason it would not work. We decided to go with the simpler and more robust approach, at least at first.

3. 2 Vision for Input - Practical Issues

The input system consists of two main parts: detection of the hand, and recognition of the hand gesture[6,8]. For the purposes of these goals, the input stream is viewed on a

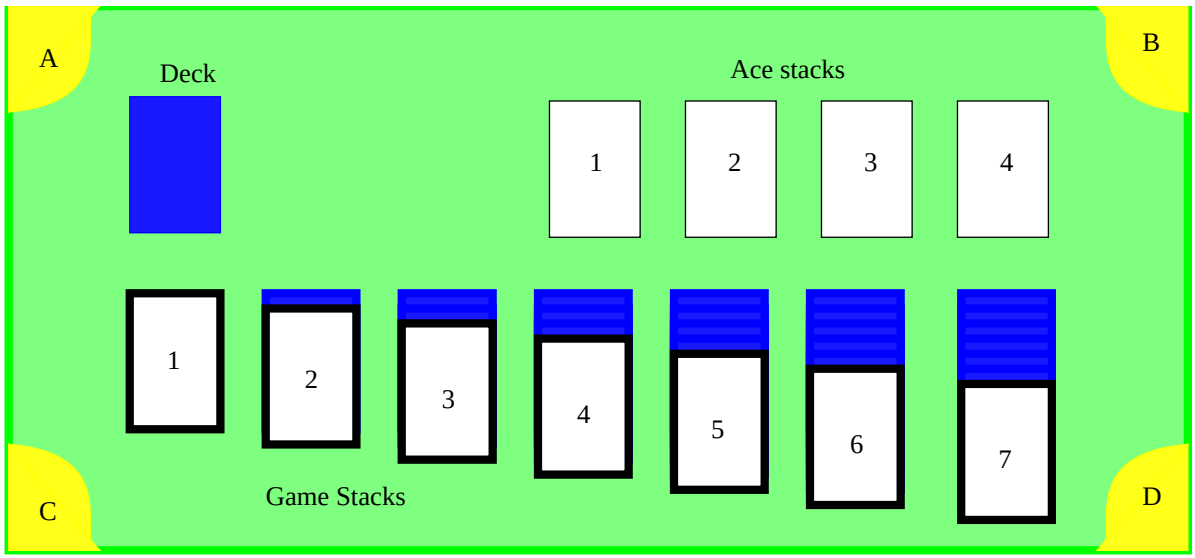

Figure 1:Virtual playing area showing stacks and the special areas for communication.

frame-by-frame basis with limited tracking between frames. The first part, hand detection, can be further broken down into background subtraction and skin detection. The background subtraction allows us to limit the search area, while the skin detector returns a fairly accurate binary representation of hand pixels within that search area.

The gesture recognizer must determine, based on the binary representation of the hand returned from the skin detector, whether the hand is in one of two distinct postures, specifically an open hand (fingers together) and a closed fist. Three separate measures are used to determine the hand posture, each using a nearest-neighbor approach followed by a majority-rules voting procedure between the three metrics.

3. 3 Background subtraction

A simple background subtraction is used to efficiently remove a large portion of the input image that obviously does not contain the hand. A copy of the background is stored when the program first starts, and is converted to the YCrCb color space. For each pixel in the current frame the Euclidean distance between the Cr and Cb values of the current frame and the stored background image is computed, and pixels having a distance greater than a threshold value of 0.05 are assumed to be background pixels and are removed from the image; this threshold was determined through experiment, and works on a range of backgrounds.

3. 4 Skin detection

To locate the hand, three separate skin tone detection algorithms are used over two color spaces. YCrCb and HSV are chosen because they have relatively compact skin tone ranges and they have good separation of luminance and chrominance [11]. Skin tone detectors are lenient to scale, orientation, and occlusions, but have a major disadvantage in that they could fail under poor illumination and brightness conditions [5]. To help minimize the problem, the incoming RGB image is translated into YCrCb and HSV [23] color spaces, both of which allow for invariance to luminance by considering it apart from the chrominance information. Separating the luminance information from an RGB image is not entirely accurate, and extreme lighting conditions will still affect the accuracy of the skin detection algorithms [8]. The hue component in our HSV space was calculated so that it is both brightness and gamma invariant [4].

After background subtraction is complete a majority of the frame has already been removed and a rectangular bounding box can effectively define the range of skin tones. A large number of skin tone samples were taken and produced two independent skin tone ranges:

HSV:

$$((0.83 \le H \le 1) \vee (0 \le H \le 0.11)) \wedge (0.2 \le S \le 0.9)$$

YCrCb: $(0.16 \le C_b \le 0.67) \wedge (0.51 \le C_r \le 0.98)$

To further aid in distinguishing between skin-toned and non skin-toned pixels which are under poor lighting conditions, a distance measure is computed between the color of a given pixel in YCrCb color space and the mean skin color, defined as:

$$\overline{C_b} = 0.41$$

$$\overline{C_r} = 0.75$$

For any pixel P(x,y), the Euclidean distance between it and the mean defined in the two equations above is calculated as:

$$\sqrt{(Cb_p - \overline{C_b})^2 + (Cr_p - \overline{C_r})^2}$$

A successful heuristic is: the lowest 10% of these distances are also defined to be skin pixels. Thus for any given image, a pixel P(x,y) is defined as skin toned if:

P is in the lowest 10% of distances as defined above

OR

P is in the valid Hue/Saturation range

AND

P is in the valid Cr/Cb range

The skin detector above returns a binary image where each pixel is either skin-tone or not. Background, especially wood and other such objects, usually has sparse skin toned pixels where real skin regions are denser. Thus all pixels without enough skin-toned neighbors are removed. The binary image is then morphologically opened [17] to separate different objects, followed by a fill routine to close areas that are completely surrounded by skin pixels. Regions are then connected, and ones that aren't considered large enough are removed from further consideration.

The best way to approach hand gesture recognition is to not have the forearm present. This, however, would not be user friendly and would also require a special marker to identify the wrist. Removing this special requirement allows a player to be wearing either a long- or short-sleeved shirt, and so the actual absence or presence of the forearm cannot be assumed. If the forearm is present, it will extend past the image boundaries and result in drastic changes to the shape and size of the hand. Hence, for gesture recognition of the hand, the forearm must be removed prior to recognition.

In order to remove the forearm from the hand the location and size of the palm is found, and its intersec-

tion with the forearm becomes the wrist location. A line approximately perpendicular to the forearm defines which pixels are to be removed.

To find the palm a distance transform is computed over the binary image acquired from the skin tone selector [3]. 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 modified version of the two pass fast transform [17], but with the addition of diagonal distances. This provides more accurate results, while still being in linear complexity relative to 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. To find the wrist it is necessary to find the direction from the center of the palm to the center of the wrist and intersect the palm circle with that point [3]. If the hand is not upside down, we can assume that the wrist will be vertically below the palm. Figure 2 shows how this is done using an image from the working system.

Since the angle from the center of the hand to the wrist is unknown, and can vary anywhere between 0 and 180 degrees below the palm, all points are checked. The palm circle is extended 1.5, 1.7, and 2.0 times to find potential locations for the wrist.

The number of skin-toned pixels along lines tangent to the circle are calculated in 2 degree arcs along the lower half of each circle; the maximum is assumed as the line intersecting the wrist. The direction between the center of the palm and the center of the wrist is stored for each circle and the two closest angles (defined by absolute difference) are averaged, and the resulting angle is assumed to point towards the center of the wrist. The three different circle sizes are used to avoid situations where the wrist is covered by a watch, bracelet, or some other occlusion.

The line determined above is then followed from the center of the palm to 1.1 times the radius of the palm circle, and a tangent to the circle at that point is calculated. All pixels beyond the tangent are assumed to represent the forearm and are removed.

3. 5 Hand gesture recognition

After passing through the background subtraction and skin-tone detection routines, it is assumed that only the hand and possibly one or more unconnected forearm regions will be present in the resulting binary image. It is also assumed that the hand is vertically above the forearm, and thus the uppermost region is selected as the hand.

Gesture recognition is composed of the three signatures described below. Each signature has five pre-calculated centroids, referred to as reference vectors, for both the open and closed positions. To classify the gesture of the current hand, a feature vector is computed for each of the signatures. A mean squared error measure is calculated between these feature vectors F and each of their associated reference vectors Ri:

$$MSE = \sqrt{\frac{\sum_{r \varepsilon F} (F_r^{\,2} + R_{i,r}^{\,2})}{|F|}}$$

A nearest neighbor approach is used to determine the closest match. Each feature vector votes either open or closed and the class with the most votes wins. If the distances between the feature vector and the reference vectors are too large, the object is not considered a hand and is ignored. If no hand has been successfully found the previous location and state are assumed.

Assuming a hand gesture is recognized, the location of the hand "hot spot" on the screen is calculated differently depending on whether it is open or closed:

(x,y) = (Center of mass + Center of palm circle + Center of bounding box)/3

For open hands, the (x,y) location of the hand is adjusted by moving it 10% of the distance between the center of the wrist and the center of the palm. Similarly, for closed hands the (x,y) location is moved 80% of the same distance. This adjustment helps to compensate for the change in dimensions between an open hand and a closed hand, which affects the center of mass and center of bounding box calculations. The 10% and 80% values were chosen because they place the hot spot in a fairly consistent location relative to the wrist.

However, due to noise, the hot spot can jump by small amounts every frame, even when the hand is physically stationary. A simple tracking method is used to help avoid this jumpiness which makes it easier to hold your hand over a small card. If the Euclidean distance between the current location of the hot spot and the last 4 hot spot locations is greater than 10 pixels, it is assumed to be in the new location. If the distance is less then 10 pixels, then the current location is averaged with the last 4 locations and the result is considered the new location of the hand. This enables the hand to move large distances quite easily, but still allows stability of the hand while hovering over small objects.

3. 6 Signatures

The **wrist ray** feature vector F is computed by projecting rays in a 360-degree arc from the calculated center of the wrist [10]. Each ray contains the Euclidean

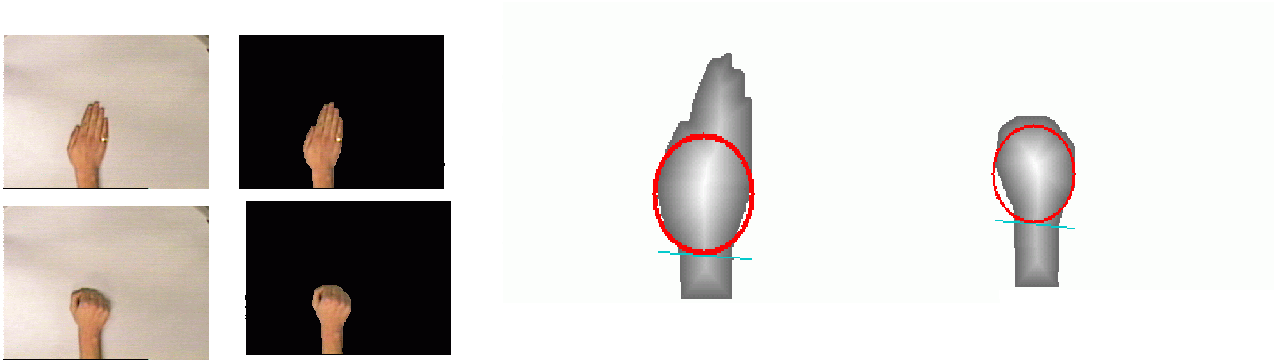

Figure 2:Steps in the processing of a hand image. (Far left) Input images of open and closed hands, as captured by the system. (Near left) The same two images after background removal and skin segmentation. (Near right) Open hand showing distance transform and wrist/forearm detection. (Far right) Closed hand showing distance transform and wrist/forearm detection.

distance from the center of the wrist to the last object pixel, in evenly spaced 4-degree arcs, giving a total of 90 rays:

$$F(\theta) = \frac{\sqrt{(row_{wrist} + row_{last(4\theta)})^2 + (col_{wrist} + col_{last(4\theta)})^2}}{width_{hand}}$$

for 0<θ<=90 degrees. Having 90 rays provides a good set of data, while not being too expensive to calculate. Fewer rays would cause too many false positives, while too many rays will be overly restrictive and cause too many false negatives. Each distance is divided by the diameter of the palm circle, which has the effect of representing the width/height ratio of a human hand. It is useful to know that the hand has approximately the right proportions, which should be relatively consistent across all human hands.

To account for the rotation of the hand the feature vector is rotated around a 360- degree arc. The nearest correlation, Ci, for each of the reference vectors is selected as the minimum mean squared error: A nearest-neighbor approach is used on the resulting Ci values to classify the hand gesture as either open or closed.

This **center ray** feature vector is similar to the wrist rays but the origin point of the rays is at the center of the palm circle, instead of the wrist [10]. This has the same effect as representing the width/height ratio, which produces good separation between open and closed hands. While the same approach is used for the two signatures, it is often the case where one will correctly characterize the hand even when the other fails.

The **hand circle** feature vector is a ratio of the number of skin pixels to the total number of pixels within concentric circles[10]. The innermost circle is the palm circle, and always has a value of one. The remaining circles have radii of:

$$mW, \quad m \in [1.2, 1.4, 1.6, 1.9, 2.2, 2.5]$$

where W is the radius of the palm circle.

Circles are rotation-invariant and these values do not require any correlation to match the reference vectors. Thus a single mean squared error measure is computed directly between the feature vector and each reference vector instead of minimizing a set of correlations as was done for the wrist and center rays measures. Again, a nearest-neighbor approach is taken to classify the gesture.

Because each of the three signatures may fail under various conditions, and often do not fail simultaneously, a majority-rules approach is taken. The classification assigned to the gesture by each of the three methods is considered to be a “vote”, and the class (either open or closed) with the most votes is considered correct [7]. To assist in minimizing small errors due to noise, which may cause an incorrect vote by more than one method, a bias is used to give precedence to the previous gesture.

4. System Evaluation and Conclusion

Although the vision input system is quite reliable, it could possibly fail in a number of places. First, the skin tone could be mis-classified, resulting in errors in later processing and culminating in an inaccurate hand location or pose. Figure 3 shows what happens in this case. It should be mentioned that the failure rate is so low that a reliable failure rate could not be determined.

Forearm removal is considered successful when it separates the hand from the forearm consistently across different hands and rotations. It doesn’t necessarily require that the forearm be removed exactly at the wrist location, but only that it is relatively consistent. Conversely, failure is defined as when considerably more or less of the forearm is removed than expected.

When the forearm removal fails, it is usually caused by an incorrect location and width of the palm circle. If the binary image returned from the skin selector is ‘wrong’ it will cause the highest distance in the distance transform to no longer be the center of the palm. Since this distance defines both the location and width of the palm circle, and therefore the location of the tangent vector that is used to

cut the forearm, an error will cause the whole forearm removal process to fail, as seen in Figure 3.

With a failure in the forearm removal, both the location of the center of the hand and the location of the wrist will not be accurate. Since these locations are used in all 3 recognition signatures, the hand will usually not recognize properly. In particular, there were three methods used to identify the hand area after forearm removal; the rates of success are:

Method	**Closed Hand**	**Open Hand**
Wrist	0.990	0.984
Center	1.000	0.923
Circle	0.995	0.668

After the majority vote of the three methods has been completed, the overall success rates are:

All three	1.000	0.987

This shows that using the vote does result in an increase in success, and that the overall error rate is low - less than 2.5%. The overall system operates at a sufficiently high speed to be practical, about 15 frames per second.

4. 1 Evaluation by Student Volunteer Subjects

An evaluation by volunteer subjects has been performed, but the results are not quantifiable. The overall response is that the players are fascinated by the ability to play the game without computer contact, and like the fact that the hand can 'grab' a card in a natural motion. Use of 3D sound was considered a plus, and the error messages were considered to be very good, even by experts. Playing the game was generally an enjoyable experience, and most of the players see obvious uses of this technology to other interfaces.

Finally, there were one or two comments that the interface was 'jerky', that the system could not quite process enough frames per second. The computer is a 1.4 GHz PC, and performance will almost certainly be adequate if a 2.0 GHz machine were used. This system did appear to be faster than the others appearing in the literature.

5. REFERENCES

[1] V. 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] 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, also Vol. 1406 of Lecture Notes in Computer Science, (Freiburg, Germany), pp. 141-157, Springer Verlag, Berlin, June 1998.

[3] Bernd Deimel and Sven Schröter: Improving Hand-Gesture Recognition via Video Based Methods for the Separation of the Forearm from the Human Hand, GW'99 - the 3rd gesture workshop - 17.-19. March 1999, Gif-sur-Yvette, France.

[4] Graham Finlayson and Gerald Schaefer, Hue That is Invariant to brightness and gamma, In Proceedings of the 12th British Machine Vision Conference, sep, 2001.

[5] B. Martinkauppi, M. Soriano, and M. Laaksonen, Behavior of Skin Color Under Varying Illumination Seen by Different Cameras in Different Color Spaces. Proc. SPIE Vol. 4301 Machine

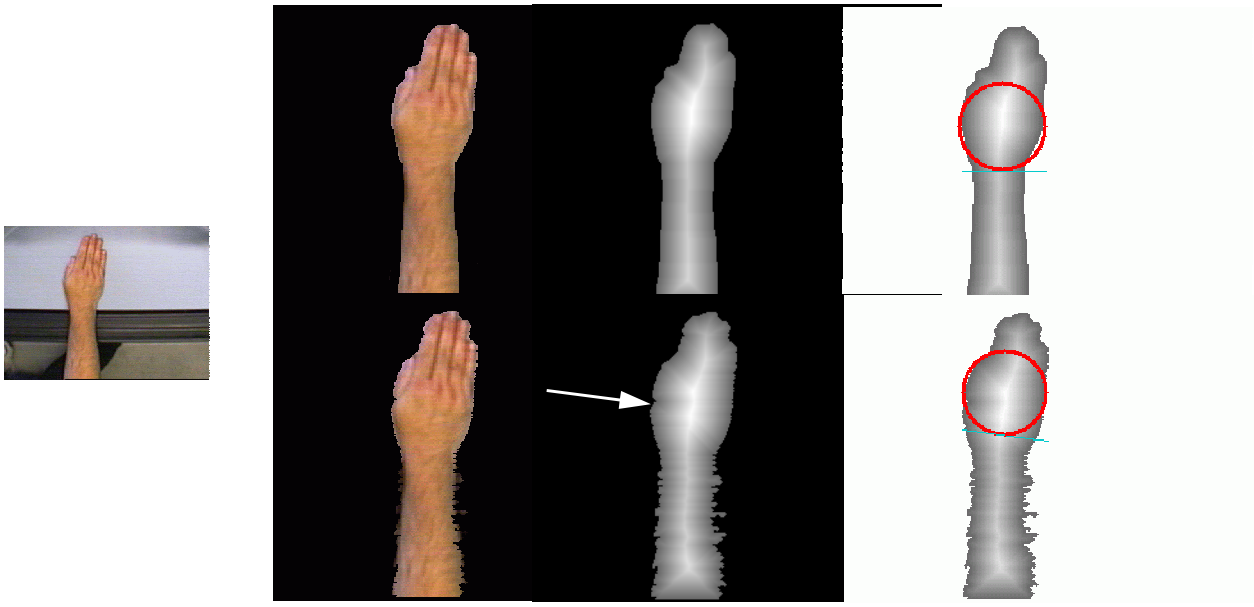

Figure 3:A failure in the skin segmentation can result is bad hand location. (Left) The original hand image, in context. (Top row) Successful skin segmentation, distance transform, and hand isolation. (Bottom row) The skin segmentation fails, resulting in a very coarse boundary - this cascades into a poor localization, and forearm

Vision in Industrial Inspection IX, January 19-26, San Jose, California, 102-113. (2001)

[6] Nobuhiko Janibata and Nobutaka Shimada, Extraction of Hand Features for Recognition of Sign Language Words, The 15th International Conference on Vision Interface May 27-29, 2002, Calgary, Canada.

[7] J R. Parker, Rank and Response Combination from Confusion Matrix Data, Information Fusion Vol. 2, 2001. Pp. 113-120.

[8] R. Sanchez-Reillo, C. Sanchez-Avila and A. Gonzales-Marcos, Biometric Identification Through Hand Geometry Measurements, Transactions on Pattern Analysis and Machine Intelligence, October 2000

[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, M. Kohler and P. G. Zyklop, Video Based Gesture Recognition for Human Computer Interaction, In W. D. Fellner (editor), Modelling - Virtual Worlds - Distributed Graphics, November 1995.

[11] J. Terrillon and S. Akamatsu, Comparative Performance of Different Chrominance Spaces for Color Segmentation and Detection of Human Faces in Complex Scene Images, Vision Interface '99, Trois-Rivieres, Canada, pp.1821, 1999.