A Method for Hand Gesture Recognition

Joshua R. New
Knowledge Systems Laboratory
Mathematical, Computing and Information Sciences Department
Jacksonville State University
newj@ksl.jsu.edu

Abstract - *Ongoing efforts at our laboratory have been aimed at developing techniques to reduce the complexity of interaction between humans and modern computer systems. As an example of these efforts, we have investigated the use of gesture recognition to allow more natural user interaction while still providing rich information content. In this paper, we present a real-time gesture recognition system which can track hand movement, orientation, and recognize the number of fingers being held up. This modularized system utilizes single-image processing techniques for noise reduction, hand segmentation, arm removal, hand movement calculations, orientation calculation, and a heuristic approach to finger-counting. With these capabilities in place, it is straight forward to utilize hand gestures to, for example, control a user interface. By supporting a more natural interface modality and utilizing only common hardware and software components, human-computer interaction can be simplified while also enriched. Sample images, results, and benchmark tests are presented.*

Keywords: Gesture recognition, machine vision, computer vision, image processing.

1 Introduction

As today's computer systems grow increasingly complex, there arises a need to increase the capability in which humans may interact with these computer systems. Computer vision techniques have been applied toward this area in which the computer system processes image input from a camera to gain information about the user's desires. This technique has proven itself as a very effective augmentation to standard input devices such as the mouse and keyboard [1]. Kjeldsen's thesis was used throughout the system design process as an inspirational guide. Since this approach requires no additional encumbrance, computer vision allows information to be gathered without being intrusive.

The gesture recognition system developed is a real-time system which allows the tracking of hand movements as well as the number of fingers being held up by the user. This system is a step toward developing a more sophisticated recognition system to enable such varied uses as menu-driven interaction, augmented reality, or even recognition of American Sign Language.

2 Methods

The system was developed using the Visual C++ compiler with the Image Processing Libraries (IPL) and Open Computer Vision (OpenCV) library from Intel. These libraries contain special image-processing functions which utilize nuances of Pentium chip architectures to ensure efficient computational performance.

2.1 Saturation Channel Extraction

A digital camera was used to capture images of a hand in several positions (top/bottom, left/middle/right). These images were saved as high-resolution JPGs. Since most cameras capture a max of 640x480 images at 30 frames per second (fps), the JPGs were reduced to a resolution of 640x480 and saved as PPMs. The saturation channel was then extracted from these images and stored as PGMs. The saturation channel has been shown in many research papers to be a good choice for gesture recognition systems, especially when the lighting conditions are stable [2, 3]. Figure 1 shows an original 640x480 PPM and its corresponding hue, saturation, and lightness/intensity/value channels.

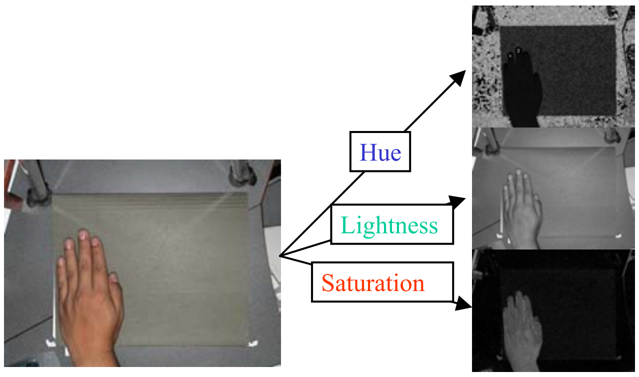


Figure 1. Hue, Saturation, and Lightness channels extracted from a color image

2.2 Threshold Saturation Channel

Once the saturation channel has been extracted, a threshold is applied to create a new image. This choice was made because it has been found that few objects in the real world are as highly saturated as human skin tones. Thus, even in a relatively noisy environment, nature makes a way for human skin to be easily segmented within an image. However, drastic changes in lighting can have a negative impact on segmentation quality.

In the system developed, a threshold value of 50/255 was found to capture most of the hand with a tolerably small amount of noise. The values at or above the threshold value were changed to 128 and the others were set to 0 via the following equation:

$$@\ \text{PixelValue} = \text{PixelValue} \geq 50\ ?\ 128 : 0 \quad (1)$$

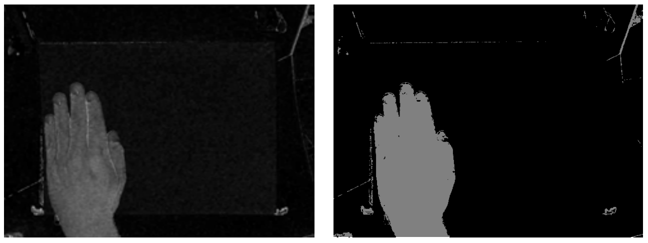


Figure 2. Saturation channel before and after threshold is applied

2.3 Calculate Centroid

The centroid, or center of mass, was calculated for the resultant threshold image. This was done by using the customary 0th and 1st moments of the image. The 0th moment of an image is defined as:

$$M_{00} = \sum\sum I(x,y) \quad (2)$$

The first moments of an image, for x and y respectively, are defined as:

$$M_{10} = \sum\sum x * I(x,y) \quad (3)$$
$$M_{01} = \sum\sum y * I(x,y) \quad (4)$$

The centroid of an image is defined as:

$$(x_c, y_c) \text{ where } x_c = \frac{M_{10}}{M_{00}} \text{ and } y_c = \frac{M_{01}}{M_{00}} \quad (5)$$

where

$$I(x,y) = \begin{cases} 1 & if \quad pixelValue \; = \; targetValue \\ 0 & \quad Otherwise \end{cases}$$

2.4 Reduce Noise

Once the centroid has been computed, a Flood-Fill is applied to the image at that location. This operation acts to change the pixel value of all pixels connected to the centroid and are of the same value, to another value. In this way, all unconnected noise can be relatively suppressed. A threshold could then be applied to eliminate all noise.

In the system developed, the thresholding operation was withheld to a later processing stage in order to save processing time. All 128-valued pixels connected to the centroid were converted to 192.

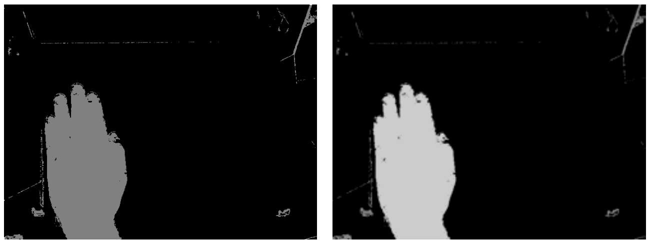


Figure 3. Image before and after Flood-Fill at the centroid

2.5 Remove Arm from Hand

In several gesture recognition systems, there is a problem with the arm playing an unwanted role in the process of gesture recognition. Some researchers address this issue by having the users where long-sleeved shirts in order to keep the arm from being segmented with the hand [4]. In order to circumvent such user restrictions, a heuristic method was used to segment the hand from the arm.

In the current system, a bounding box was computed for the 192-valued pixels by finding the topmost and leftmost pixels of the target value in the image. A box of black pixels is then written using the calibration measure of hand size, effectively drawing a line at the wrist. A Flood-Fill is then applied at the centroid to turn all 192-valued pixels to 254.

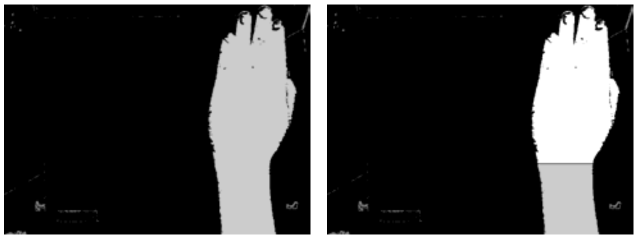


Figure 4. Image before and after bounding box is applied to the hand and Flood-Fill at the centroid to separate the arm from the hand

2.6 Calculate Refined Centroid

Once the hand has been segmented from the arm, a threshold is applied to the image in order to remove all noise as well as the arm. At this point, only pixels representing the hand should be remaining. A refined centroid is then computed which represents the "true" location of the hand.

In the system developed, 254-valued pixels were thresholded to 255. The centroid for 255-valued pixels was then computed in the manner discussed previously.

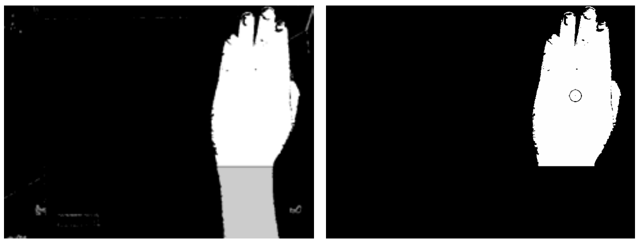


Figure 5. Image before and after threshold is applied to remove noise; the circle was added only to enhance the location of the refined centroid

2.7 Calculate Orientation

The orientation, or image major axis, of the hand is also calculated using the second moments of the image, for x and y respectively:

$$M_{20} = \sum\sum x^2 * I(x,y) \quad (6)$$

$$M_{02} = \sum\sum y^2 * I(x,y) \quad (7)$$

where the orientation of the image is given by:

$$\Theta = \frac{\arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c{}^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c{}^2\right)} \right)}{2} \quad (8)$$

This orientation measure was implemented in order to take into account rotations of the hand in the image. That is, the current system only works when the hand is straight up or down. However, the orientation measure was never verified to behave correctly or applied toward rotation-independent gesture recognition.

2.8 Count Fingers

In order to count the number of fingers being presented, a heuristic approach was used which sweeps out a circle centered at the refined centroid and with a radius computed based on the calibration measures. A good radius size for this circle was found to be:

$$radius = .17 * \left(HandsizeX + HandsizeY\right) \quad (9)$$

In this system, a finger is defined to be 10+ white pixels separated by 3+ black pixels, which function as salt/pepper tolerance, minus 1 for the hand itself. This proved to be very effective for counting fingers in most cases, but thumb counting proved problematic since a widely extended thumb would be counted as part of the wrist.

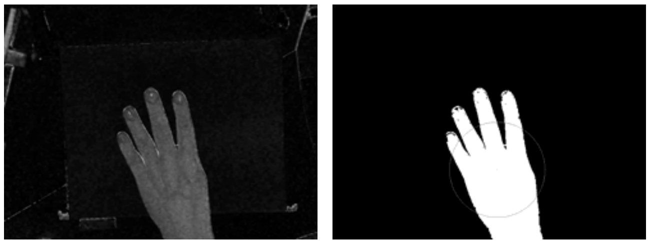
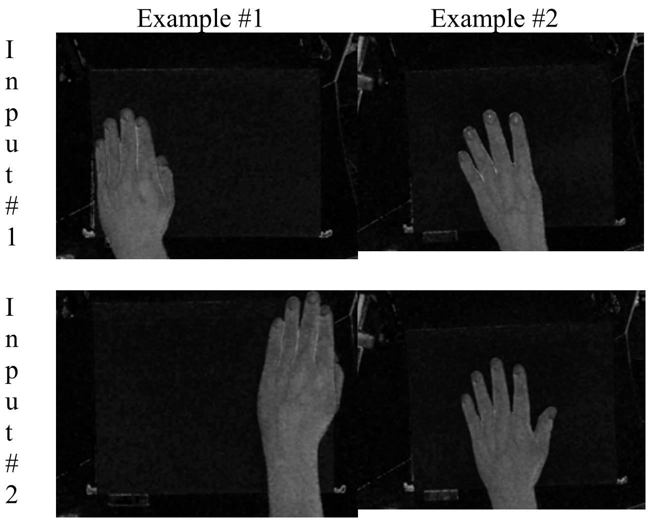


Figure 6. Original saturation channel and result after complete processing ; shows the arc swept out to count the fingers

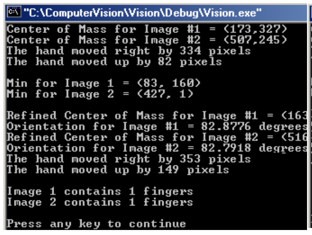
3 Results

The gesture recognition system developed was designed to run in real-time on 640x480 images, which translates to 30 frames per second (fps). The system was tested on various images on two computers. The average results are shown in Table 1.

The current system takes 16.5 ms to completely process one frame on a 1.33 Ghz system (without reading or writing the ASCII pgm files which is slower than reading or writing Binary pgm files). This translates to an effective processing of 60 fps, allowing future extensions to be made to the system while still being capable of running in real-time.



Output



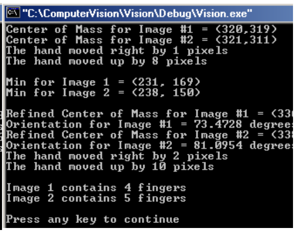
```
"C:\ComputerVision\Vision\Debug\Vision.exe"
Center of Mass for Image #1 = (173,327)
Center of Mass for Image #2 = (507,245)
The hand moved right by 334 pixels
The hand moved up by 82 pixels

Min for Image 1 = (83, 160)
Min for Image 2 = (427, 1)

Refined Center of Mass for Image #1 = (163
Orientation for Image #1 = 82.8776 degrees
Refined Center of Mass for Image #2 = (516
Orientation for Image #2 = 82.7918 degrees
The hand moved right by 353 pixels
The hand moved up by 149 pixels

Image 1 contains 1 fingers
Image 2 contains 1 fingers

Press any key to continue
```



```
"C:\ComputerVision\Vision\Debug\Vision.exe"
Center of Mass for Image #1 = (320,319)
Center of Mass for Image #2 = (321,311)
The hand moved right by 1 pixels
The hand moved up by 8 pixels

Min for Image 1 = (231, 169)
Min for Image 2 = (238, 150)

Refined Center of Mass for Image #1 = (33
Orientation for Image #1 = 73.4728 degree
Refined Center of Mass for Image #2 = (33
Orientation for Image #2 = 81.0954 degree
The hand moved right by 2 pixels
The hand moved up by 10 pixels

Image 1 contains 4 fingers
Image 2 contains 5 fingers

Press any key to continue
```

Process Steps	**Time (ms)**	
	Athlon MP 1500 (1.33 Ghz)	Pentium III (850 Mhz)
2.1) Saturation Extraction	NA	NA
Reading Image	208	340
2.2) Threshold	.5	6.5
2.3) Centroid	3.5	18.5
2.4) Flood Fill	1.5	27
2.5a) Bounding Box Top-Left	3.5	5.5
2.5b) Arm Removal	2	34.5
2.6) Refined Centroid	4	19
2.8) Finger Counting	.5	1
Write Image	233	324
Time w/o R&W	16.5	112
Time w/o Write	224.5	452
Total Time	457.5	776.5

Table 1. This table shows the gesture system runtimes step-by-step for two computers

4 Future Work

Many heuristic approaches were taken in the course of this work to ensure real-time performance. Since there is still much time which could be utilized, many other capabilities and other sophisticated approaches could be used to enhance the current system.

4.1 Input Automation

The current system's automated processing begins by reading saturation channel images saved by the human operator. A better approach would be to read memory straight from the capturing camera and automatically extract the saturation channel. The saturation channel is simply a linear combination of the image's RGB values [5].

4.2 Optimization

While the current system is highly optimized through the use of OpenCV's processor-dependent functions, other approaches could be used to accomplish the same function as those currently used. For example, a *largest connected component* algorithm could be used instead of finding the center of mass and then flood-filling. This would also allow for two hands to be used in the image while also being more noise tolerant.

4.3 Orientation for Hand Registration

The current system is capable of computing the orientation of the hand. However, this measure has not been validated to be robust nor been used by other stages in the system such as application of the bounding box for arm removal. The orientation of the hand is also an important variable for hand registration, which is required by a learning system.

4.4 New Finger Counting Approach

The finger counting approach utilized is a simple heuristic and is not robust, especially for thumb counting. A better heuristic would be to count along a semicircle centered at the wrist in line with the orientation of the image major axis. However, the best approach would be to add the capability of a machine learning system for gesture classification.

4.5 Learning System

A learning system is the next logical addition to the gesture recognition system. Current approaches vary from recognition based on edge-detection to interior pixel values [4, 6]. A learning system would allow much more sophisticated gesture recognition capabilities.

References

[1] Kjeldsen, "Visual Interpretation of Hand Gestures as a Practical Interface Modality", Ch. 6, Columbia University, 1997.

[2] M. Störring, H.J. Andersen, E. Granum, "Skin Colour Detection Under Changing Lighting Conditions", *7th Symposium on Intelligent Robotic Systems*, Coimbra, Portugal, 1999, 187-195.

[3] M. Soriano, B. Martinkauppi, S. Huovinen, M. Laaksonen, "Skin Detection in Video Under Changing Illumination Conditions", *IEEE International Conference on Pattern Recognition*, Vol.1, Barcelona, Spain, 2000, 839-842.

[4] Kjeldsen, "Visual Interpretation of Hand Gestures as a Practical Interface Modality", Ch. 4, pg. 6, Ch. 3, pgs. 29-36, Columbia University, 1997.

[5] L. Shapiro, G. Stockman, Computer Vision, pg. 196, Prentice Hall, 2001

[6] K. Yow, R. Cipolla, "Feature-Based Human Face Detection", Cambridge University, 1996.