Facilitating User Interaction with Complex Systems via Hand Gesture Recognition

Joshua R. New
Knowledge Systems Laboratory
Jacksonville State University
Jacksonville, AL 36265
(256) 782-5103
newj@ksl.jsu.edu

Erion Hasanbelliu
Knowledge Systems Laboratory
Jacksonville State University
Jacksonville, AL 36265
(256) 782-5103
erioni@ksl.jsu.edu

Mario Aguilar
Knowledge Systems Laboratory
Jacksonville State University
Jacksonville, AL 36265
(256) 782-5718
marioa@ksl.jsu.edu

ABSTRACT
Ongoing efforts at our laboratory have been aimed at developing techniques that reduce the complexity of interaction between humans and computers. In particular, we have investigated the use of a gesture recognition system to support natural user interaction while providing rich information content. In this paper, we present a real-time gesture recognition system which can track hand movement, define orientation, and determine the number of fingers being held up in order to allow control of an underlying application. The system was developed in a modular fashion to maximize reutilization and portability. We utilized off-the-shelf image processing libraries and a low-cost web camera. Its main functional components consist of single-frame image processing for noise reduction, hand segmentation, arm removal, hand displacement and pose calculations, and a heuristic approach to finger-counting. In our particular implementation, the hand gesture information is used to control a medical image visualization application developed in our lab to illustrate multi-modal information fusion. Here, a 3D model of a patient's skull, generated from the patient's MRI imagery, is directly manipulated by hand gestures. Within the application, the user can rotate the skull or zoom by reconfiguring or moving their hand in prescribed directions in front of the video camera. For example, moving the hand to the left or right of the camera's field-of-view corresponds to yaw left or yaw right, respectively. By supporting a more natural interface modality and utilizing only common hardware and software components, human-computer interaction has been simplified while also enriched. Sample images, results, videos, and benchmark tests are presented.

Keywords
Gesture recognition, machine vision, computer vision, image processing, human-computer interaction.

1. INTRODUCTION
As today's computer systems grow increasingly complex, the need arises to increase the capability by which humans interact with these computer systems. Computer vision techniques have been applied to this end with significant success. Here, the computer system processes image input from a camera to gain information about the user's intentions. This technique has proven very effective at augmenting standard input devices such as the mouse and keyboard [6]. The work of Kjeldsen [6] serves as a platform for the efforts described in this manuscript. Since this approach requires no additional encumbrance, computer vision allows user-interaction information to be gathered without being intrusive.

The gesture recognition system we have developed is a real-time system that allows the tracking of hand movements as well as counting 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.

Additionally, an interface was developed to test the efficacy of program manipulation using hand gesticulation. This interface allows input to be taken from a camera in real-time and operated on by the gesture recognition system, the results of which are used to manipulate a 3-dimensional skull. This interface has provided us with a platform that supports interaction with more complex environments such as graspable interfaces and augmented reality. This interface provides a truly hands-free method for human-computer interaction in a way that minimizes obstruction of the user's tasks.

2. RECOGNITION METHODS
The gesture recognition system was developed using the Microsoft Visual C++ compiler with the Image Processing Libraries (IPL) and Open Computer Vision library (OpenCV) from Intel. These libraries contain special image-processing functions optimized for the Pentium architecture to ensure efficient computational performance.

The gesture recognition system is capable of tracking hand movement and counting the number of fingers that the user is holding up. The only calibration measure required is the maximum hand size in the x and y directions in number of pixels (shown in Fig. 11). This could be obtained when the system first

starts up by the user placing his/her hand in a calibration box from which the length and width is measured. The system uses this information to process the image through a series of pipelined stages, successively refining the information about the position of the hand by removing noise and segmenting the hand from the background (Fig. 1). Once the position of the hand is known, the system then counts the number of fingers that the user is holding up. Each stage of this pipeline is discussed in the sections below.

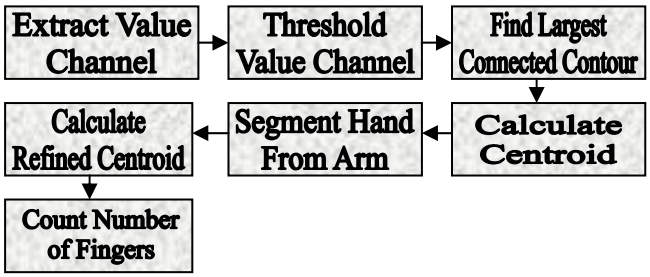


Figure 1. Design and components of system pipeline.

2.1 Extract Saturation Channel

The system uses an off-the-shelf web camera, capable of capturing 640x480 images at 30 frames per second (fps), to capture the desk area where the user's hand is located. Individual frames are captured and sent directly to memory for initial processing rather than storing and loading image files from disk. This supports reduced latency and real-time performance.

The current implementation of the gesture recognition system supports the use of live camera input or stored image files. The saturation channel (HSL space) of the captured RGB image is then extracted [7] since it has been shown [8] and [10] to be a good choice for gesture recognition systems when lighting conditions are stable. Fig. 2 shows an original 640x480 image and its corresponding hue, saturation, and lightness channels.

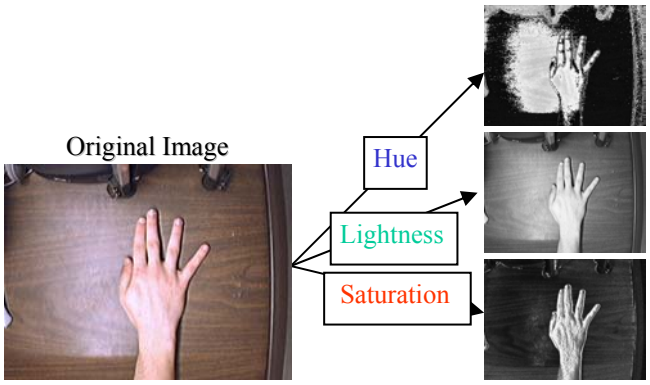


Figure 2. Hue, Saturation, and Lightness channels extracted from a color image.

2.2 Threshold Saturation Channel

Once the saturation channel has been extracted, a threshold is applied to create a new image. Few objects in the real world are as highly saturated as human skin tones so nature makes a way for human skin tones to be easily segmented within an image. Additionally, it is more robust than the value channel to lighting and camera changes.

In the system developed, a threshold value of 100 was found to capture most of the hand with a tolerably small amount of noise in our static, indoor lighting environment. The pixels with a value at or above the threshold value were changed to 128 and the others were set to 0 via the following equation:

$$@\ \text{PixelValue} = \text{PixelValue} \geq 100\ ?\ 128 : 0 \quad (1)$$

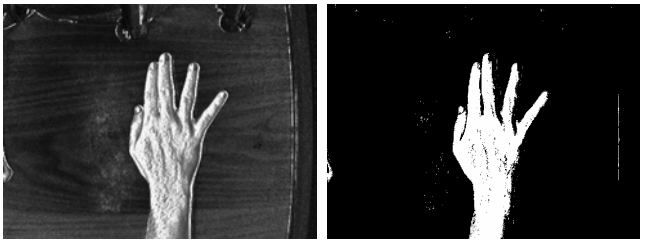


Figure 3. Saturation channel before and after threshold is applied.

2.3 Find Largest Connected Contour

This stage finds all contours in the resultant threshold image. Once all contours are found, only the largest one is retained and all pixels on or inside that contour are set to 192. This process effectively selects the contour of the hand and fills all holes/lakes within the hand for pixels that had not made the threshold. This also removes all noise not connected directly to the edge of the hand and thus eliminates most salt and pepper noise in the image.

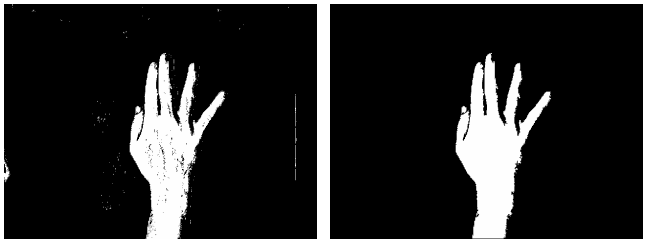


Figure 4. Before and after largest connected contour.

2.4 Calculate Centroid

The centroid, or center of mass, was calculated for the resultant largest contour image. This was done by using the customary 0^{th} and 1^{st} moments of the image. The 0^{th} moment amounts to a simple count of the number of pixels of a particular value within an image and is defined as:

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

The first moments of an image, for x and y respectively, are the sum of the x and y values for pixels of a particular value and are defined as:

$$M_{10} = \sum_x \sum_y x * I(x,y) \quad (3)$$

and

$$M_{01} = \sum_x \sum_y y * I(x,y) \quad (4)$$

where,

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

Finally, the centroid of the segmented hand is computed as:

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

(This computation is isolated to the centroid of the hand due to our prior segmentation stage.)

2.5 Segment Hand from Arm

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 wear long-sleeved shirts in order to keep the arm from being segmented with the hand [5]. In order to circumvent such user restrictions, a heuristic method was used to segment the hand from the arm while not encumbering the user.

In the current system, a bounding box is computed for the 192-valued pixels by finding the top-most and left-most pixels of the target value in the image. A box of black pixels is then defined using the measure of hand size as determined during calibration, effectively drawing a line at the wrist. Then, pixels inside the box are set to 254 to differentiate them from the arm.

Figure 5. 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 the arm. At this point, only pixels representing the hand should remain. A refined centroid is then computed which represents the "true" location of the hand.

In the system we implemented, pixels were thresholded at 254 and their values set to 255. The centroid for the remaining pixels was then computed in the manner discussed in section 2.4.

Figure 6. Image before and after threshold is applied to remove the arm; the circle was added only to enhance the location of the refined centroid.

2.7 Count Number of 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. In our implementation, an effective radius size for this circle was found to be:

$$radius = .19 * (HandsizeX + HandsizeY) \qquad (9)$$

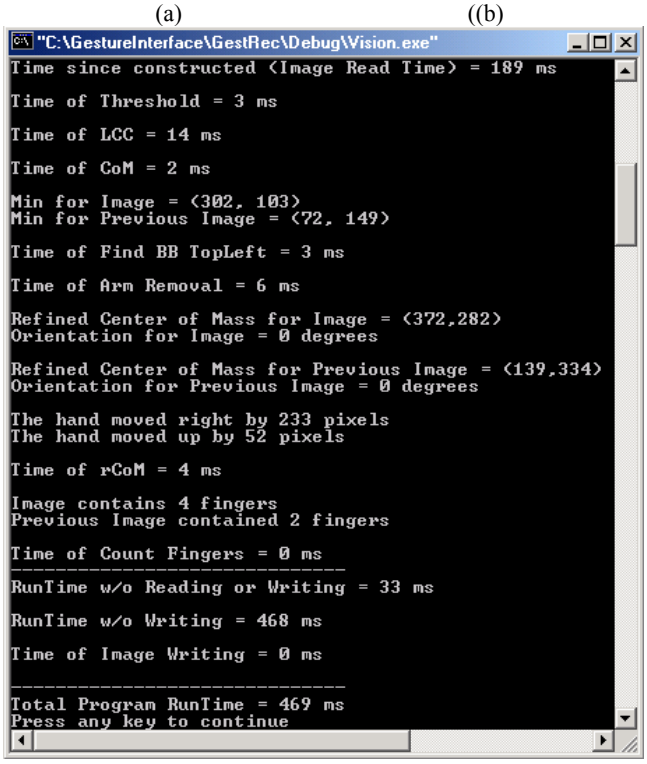
In this system, given hand size calibration, a finger is defined to be 15+ white pixels separated by 18+ black pixels, which serve as salt/pepper tolerance, minus 1 for the hand itself. This means that for the fingers to be counted as separate, they must be sufficiently spread. This proved to be very 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. This problem is typically circumvented by including a small portion of the wrist in the calibration measure.

Figure 7. Original saturation channel and result after complete processing; shows the arc swept out to count the fingers.

3. RESULTS

Two sample extracted saturation channels were written to PGM files and used as input to the gesture recognition system. These images are shown in Fig. 8. The output produced by the gesture recognition system after processing these images is also shown. The output screen shows not only the information extracted from the image, but also the execution time required by each stage of the system.

(a) ((b)



(c)

Figure 8. Gesture recognition system input and output a) First input image; b) Second input image; and c) Output screen showing quantitative processing results and time required per stage.

One of the main goals in the design of the gesture recognition system was that it runs in real-time on 640x480 images. The system was tested on various images with an AMD Athlon XP 1900 (1.6 Ghz) machine. The average results are shown in Table 1 below.

The current system takes 41 ms to completely process one frame from the camera on a 1.6 GHz system. This translates to an effective recognition processing rate of 24 fps. However, additional optimizations could be made to decrease processing time and thus allow the addition of more robust processing techniques. Nonetheless, for the purposes of gesture-based human-computer interaction, we find that 15+ fps are quite sufficient to provide smooth and rapid control of a human-computer interface system. These results are described in the following section.

Table 1. Gesture System Benchmarks

Process Steps	Time (ms) Athlon XP 1900 (1.6 Ghz)
1) Extract Sat Channel	9
2) Threshold	3
3) Find Connected Contour and Fill	14
4) Centroid	2
5) Segment Hand From Arm	9
6) Refined Centroid	4
7) Count Number of Fingers	0
Total Time	41

4. INTERFACE TESTING

The interface was developed using a C++ compiler with the platform independent, GUI-building Qt toolkit. This toolkit allows targeting of multiple operating systems while incurring no performance hit. However, the current system has only been tested on Windows XP and 2000 Professional.

The interface created includes two panels: one with the 3D model of a skull to provide the user with a complex object for direct manipulation; the other with the camera feed so that the user can obtain feedback on their interaction with the computer without ever having to take their eyes off the screen. The interface also provides dials so that the user can easily quantify the roll/pitch/yaw angle or zoom factor that has been applied to the 3D model.

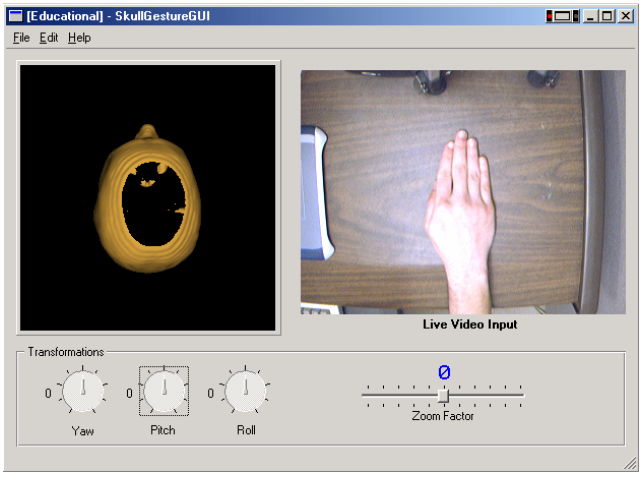


Figure 9. The GUI provided for interaction with the system

The setup of the camera and gesturing by the user can be seen in Fig. 10. This configuration was considered ideal since the user sits naturally at the computer and needs only shift his/her hand away from the mouse to be in the camera's active gesticulation zone. This allows for quick and easy interaction while allowing the user to maintain concentration on the task at hand even when switching between interface modalities.

Figure 10. Current system configuration

The interface system allows input to be taken from a camera and operated on by the gesture recognition system in real-time, the results of which are used to manipulate the 3-dimensional skull in the opposing window. The gestures supported for defining the rotation of the skull are illustrated in Fig. 11. Other manipulations not listed there include rolling left or right by holding up two or three fingers and zooming in or zooming out by holding up four or five fingers, respectively. Care was taken so that the mappings of gestures to actions were as intuitive as possible; however, the effects of some gestures must still be learned by the user either through experimentation or training. Rotations of five degrees and zooming speeds of ±5% of the current zoom factor per frame were found to be acceptable speeds for providing smooth control of the interface.

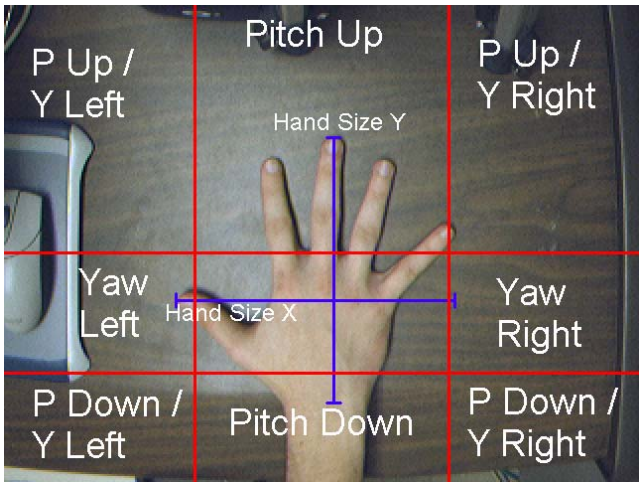


Figure 11. Mapping of hand placement to system control

It is obvious that the use of this system is facilitated by the ease with which users can naturally interact via gesturing. However, the power of the system will increase tremendously as its recognition capabilities are enhanced, more complex interactive environments are introduced, and more user training of gestures are provided. However, the mapping between gestures and actions should be defined as intuitively as possible to ease the burden of learning to use the new interface. Due to the fact that this simple system can prove to be natural and fun for most users, it is expected that the learning of more sophisticated gesture-action mappings will be learned independently by the user as he/she gains experience with the system.

5. FUTURE WORK

Many heuristic approaches were utilized in the course of this work to ensure real-time performance. Since our current implementation may execute at a faster frame rate than that required for a specific task, other capabilities and more robust approaches could be used to enhance the current system.

5.1 Optimization

While the current system is highly optimized through the use of OpenCV's processor-dependent functions and several similar functions were written that computed only the information required, there may exist other approaches that accomplish the same function as those currently used with lower computational costs. Also, for existing functions, there is still some improvement that could be made such as the time required to retrieve and edit image data.

5.2 Computation of Hand Pose

The current system is capable of computing the orientation of the hand. However, this measure has not been validated to be robust nor has it been used in other stages in the system, such as in application of the bounding box for arm removal. This isn't an important issue since the user's hand is almost always oriented vertically due to the way the system is setup. However, the orientation of the hand is also an important variable for hand registration, which is required by a gesture recognition learning system.

5.3 New Finger Counting Approach

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

5.4 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 more robust gesture recognition capabilities and thus increase the number and reliability of mappings from gestures to system actions.

Considerations involved in the design of a learning system would include invariance to hand pose, selection of the

training set to support robust performance across multiple users and/or lighting conditions, and the type of training system utilized (i.e. off-line vs. on-line learning, supervised, etc.) The learning system could not only open up a number of new applications but also support the calibration of the system by adapting the system parameters as a function of current conditions.

5.5 Interface Extensions

Previously, similar recognition systems have been used successfully to aid user's with reduced mobility or impaired motor skills, such as those who suffer from cerebral palsy [9]. While the current system has not been evaluated for those use in that context, relatively easy extensions could be made to address such applications.

While the interface implemented here was effective in demonstrating gesture-based control of a 3D object, it barely tapped the potential for interaction via such systems. This interaction technique could be used for more complex applications where the user may draw and manipulate objects in a 3D modeling program by using intuitive hand gestures. Furthermore, other contexts and applications of the system are being considered including interaction with graphical user interfaces utilizing a "graspable interface" [3] similar to that used in Spielberg's "Minority Report" film [2]. Other examples include interaction with multiple objects in a virtual environment [4] and its use in augmented reality [1] applications in order to enable transparent interaction.

6. ACKNOWLEDGMENTS

This work was made possibly by a Graduate Research Assistantships awarded to the first author by Jacksonville State University. The second and third authors were funded under a NASA Ames Research Center Grant contract number NCC2-1330. Opinions, interpretations and conclusions are those of the authors and not necessarily endorsed by Jacksonville State University nor NASA.

7. REFERENCES

[1] Bajura, M., Fuchs, H., and Ohbuchi, R., "Merging virtual objects with the real world: Seeing ultrasound imagery within the patient", *ACM SIGGRAPH Computer Graphics*, Vol. 26, 203-210, July 1992.

[2] Clarke, D., "MIT grad directs Spielberg in the science of moviemaking", *MIT Tech Talk*. July 17, 2002.

[3] Ernst, H., Schafer, K., and Bruns, W., "Creating Virtual Worlds with a Graspable User Interface". *15th Twente Workshop on Language Technology*, University of Twente, NL, 1999.

[4] Furness, T., Exploring Virtual Worlds. ACMMEMBERNET, Vol. 34. No. 7, July 1991.

[5] Kjeldsen, F., "Visual Interpretation of Hand Gestures as a Practical Interface Modality", Ch. 3, 29-36, Ch. 4, 6-7, Columbia University, 1997.

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

[7] Shapiro, L., and Stockman, G., Computer Vision, 196-197, Prentice Hall, 2001.

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

[9] Starner et al., "The Gesture Pendant: The Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring", *International Symposium on Wearable Computing*, Georgia Tech, GA, 2000.

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