A Tool for Hand-Sign Recognition

David J. Rios Soria and Satu Elisa Schaeffer

Postgraduate Division in Computation and Mechatronics (DCM),
School of Mechanical and Electrical Engineering (FIME),
Universidad Autónoma de Nuevo León (UANL),
San Nicolás de los Garza, NL, Mexico

Abstract. We present a software tool created for human-computer interaction based on hand gestures. The underlying algorithm utilizes computer vision techniques. The tool is able to recognize in real-time six different hand signals, captured using a web cam. Experiments conducted to evaluate the system performance are reported.

1 Introduction

There are situations where it is necessary to interact with a system without touching it. The reasons for this necessity are numerous, including dirty hands (when repairing a motor, for example), hygiene (to indicate the desired water temperature when washing hands in a public bathroom), and the focus of attention (not wishing to redirect the sight towards the controls when operating delicate equipment or interacting with an augmented-reality scenario). Applications for hand-gesture control include medical systems — provides the user sterility to avoid the spread of infections — entertainment, and human-robot interaction [1].

In this paper, we formulate a hand-sign detection algorithm and implement this for use with a standard web-cam quality image; the proposed system is to be employed in future work in an augmented-reality setting, where the user wears a visor equipped with a camera. We also document user experiments carried out with the implemented tool. The article is structured as follows: after a brief background discussion in Section 2, we present the proposed system in Section 3. Section 4 discusses our prototype implementation and user experiments, and in Section 5 we conclude the work and discuss future directions.

2 Background

Hand-recognition systems are based on processing an incoming digital image, preferably in real-time. The first task is to separate the image of a hand from the background. This can be achieved in several ways and depends on whether the image includes only a hand against a background or the entire person. Options for detecting the hand against a background, which is the typical case for the augmented-reality setting where the user wears a headset with a camera pointing

J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2012, LNCS 7329, pp. 137–146, 2012.

towards his or her field of vision, include either comparing the subsequent frames in a video (to detect movement — sensitive to motion in the background as well as shaking of the hand itself) or using a *skin-color filter* (to classify the pixels of the image into two classes, "hand" or "background", depending on their RGB color values).

The latter approach, which is somewhat sensible to high variations of skin color (the problematic cases being very pale and very dark-skinned users). The advantage is that it can be done on a single frame, that is, a still photograph, but can often be improved by averaging over a few adjacent video frames. The skin-color filtering in such does not yet necessarily produce a picture of the hand only, as some pixels belonging to the background may pass through the filter whereas parts of the hand that are either shadowed of reflect light are excluded. Hence we need to apply several processing steps, first to extract the hand and then to identify the signal that the user is currently making.

2.1 Skin-Color Filtering

Skin color has proven to be a useful and robust cue for face detection, localization, and tracking [2–4]. Image content filtering, content-aware video compression, and color-balancing applications can also benefit from automatic detection of skin in images. The goal of skin-color detection is to construct a decision rule to discriminate between skin and non-skin pixels. This is usually accomplished by introducing a metric, which measures the distance (in a general sense) of the color of a given pixel to defined value for skin tone. The specific distance metric employed is defined by the skin-color modeling method.

Colorimetry, computer graphics, and video signal transmission standards have given birth to many color spaces with different properties. A wide variety of them have been applied to the problem of skin-color modeling. The red-blue-green (RGB) is a color space that originated from cathode-ray tube display applications, where it was convenient to describe each color as a combination of three colored rays: red, green, and blue. Presently, it is one of the most widely used color spaces for processing and storing of digital image data.

2.2 Edge Detection

Edge detection is an essential tool in image processing and computer vision, particularly in the areas of feature detection and feature extraction. An *edge* is defined as the boundary between an object and the background, although it may also indicate the boundary between overlapping objects or, unfortunately, a change of color within an object with no difference of depth. The process of edge detection is generally based on identifying those pixels at which the image brightness has discontinuities. When the edges in an image are accurately identified, the objects in it can be located, allowing the computation of basic properties of each object, such as the area, perimeter, and shape [5].

There are two main methods of used for edge detection; namely the *template matching* and the *differential gradient* methods. In both of these methods, the

goal is to identify locations in which the magnitude of the intensity gradient (that is, the change that occurs in the intensity of pixel color when moving across adjacent pixels) is above a threshold, as to indicate in a reliable fashion the edge of an object. The principal difference between the two methods is the way in which they perform local estimation of the intensity gradient g, although both techniques employ convolution masks.

The template matching operates by taking the maximum over a set of component masks (such as the Roberts, Sobel, and Prewitt operators) that represent possible edge configurations. This yields an approximation for g at the pixel in which the templates are centered. The differential gradient method instead computes the pixel magnitudes vectorially with a nonlinear transformation. After computing g for each pixel — with either of these methods — thresholding is carried out to obtain a set of contour points (that is, those that were classified as being part of an edge). The orientation of the edges can be deduced from the direction of the highest gradient (the edge being perpendicular to it at that pixel).

2.3 Existing Methods

There are several techniques for hand sign detection; our work focuses on vision-based hand-sign detection. The literature reviews in the field are numerous [6–10]. In particular, Malima et al. [11] use hand-gesture detection for remotely controlling a robot. Their approach segments the hand, locates the fingers, and finally classifies the gesture into one of five different signs. The algorithm is invariant to translation, rotation in a plane or rotating 180 degrees, and scale of the hand. They obtained 91% precision in their experiments. Images taken under insufficient light (especially using the webcam) led to the incorrect results. In these cases, the failure mainly stems from the erroneous segmentation of some background portions as part of the hand.

3 Proposal for a Hand-Sign Recognition System

In this section, we explain the steps implemented in our proposed system for hand-sign recognition. We use the average over ten frames as input to skin filtering and formulate our system in terms of the RGB color space. Denote by I be the entire input image, and by I_R, I_G and I_B the red, green, and blue channels of the image. We denote the image height in pixels by h and the image width in pixels by w. The pixel in position (i, j) is denoted by $p_{i,j}$ and its three components by $p^R_{i,j}$, $p^G_{i,j}$, and $p^B_{i,j}$. For all components $C \in \{R, G, B\}$, we assume that $p^C_{i,j} \in [0, 255]$, corresponding to eight bits per color channel, yielding 24 bits per pixel. This gives image size of $h \times w \times 24$ bits.

We use a pixel-based skin detection method [12] that classify each pixel as skin or non-skin individually, independently from its neighbors. More complex methods that take decisions based not only on a pixel $p_{i,j}$, but also on its direct neighborhood $\{p_{i-1,j}, p_{i+1,j}, p_{i,j-1}, p_{i,j+1}\}$ (and possibly also the diagonal

Fig. 1. On the left, an original RGB image I. On the right, the resulting binary image B after applying the skin-color filter defined by Equations 1-4.

neighborhood $\{p_{i-1,j-1}, p_{i+1,j-1}, p_{i+1,j+1}, p_{i-1,j+1}\}$) can be formulated, but are computationally heavier. Our aim is to operate the system in real time, for which we seek the simplest and fastest possible method for each step.

A pixel $p_{i,j}$ in I is classified as skin (true or false) if all of the following conditions simultaneously apply:

1. The three components all exceed their corresponding threshold value:

$$p^R_{i,j} > 95 \wedge p^G_{i,j} > 40 \wedge p^B_{i,j} > 20. \tag{1}$$

2. The difference between the maximum value and the minimum value over the three components exceeds a defined threshold:

$$\max\{p^R_{i,j}, p^G_{i,j}, p^B_{i,j}\} - \min\{p^R_{i,j}, p^G_{i,j}, p^B_{i,j}\} > 15. \tag{2}$$

3. The red and green channels differ for more than a predefined threshold:

$$|p^R_{i,j} - p^G_{i,j}| > 15. \tag{3}$$

4. The red channel dominates the green and the blue channels:

$$p^R_{i,j} > p^G_{i,j} \wedge p^R_{i,j} > p^B_{i,j}. \tag{4}$$

We then create a new binary image B of the same dimension $w \times h$ (cf. Figure 1 for an example) where the color of the pixel $b_{i,j}$ is either white (denoted by 1) if the position corresponds to skin or black (denoted by 0) if the position did not pass the skin filter. An obvious advantage of this method is the simplicity of the skin-detection rules that enables the implementation of a very fast classifier [3].

Now that we have a binary image corresponding to the presumed skin pixels, we need to determine which of these form the hand, meaning that we need to identify the border of the hand shape in the image. Having extracted the border (we use the OpenCV [13] library), we have the set of contour pixels and need to determine the connected components of the contour, meaning that we must compute the set of edge points that are connected. In our case, we assume the hand to be in the image foreground (due to the augmented-reality setting), making it likely that the largest connected contour component will correspond to the hand, whereas any smaller components of the contour set, if present, correspond to some objects on the background.

We denote the set of contour pixels of the largest connected component by E. We construct a new binary image O by copying B and then setting to zero (black) all those pixels that correspond to the smaller connected components of the contour and their insides, leaving only E and the pixels inside it at one (white). This can be done by a standard bucket-fill algorithm. At this point, we have identified the border of the hand in the image. We now proceed to determining which hand sign is being made in the image. The way in which this is done depends on the the type of hand signs supported by the system — no single design is adequate for all possible hand positions. The signs that we wish to detect are shown in Figure 2; the easiest way to increment the number of signals detected is to allow for the use of both hands and assigning different meanings to distinct finger combinations. However, this would present severe increase to the cognitive load of the user.

Fig. 2. The six hand signs used in our proposed system. These signs correspond to the numbers from zero to five. Note that the separation of the individual fingers is relevant to the detection of these signs under the present algorithm.

As our signs correspond to distinct numbers of fingers elevated, our detection method is based on counting the elevated fingers in the image. It will not be relevant which finger is elevated, only the number of fingers (cf. [14, 11]). This gives us the advantage of the system not being sensitive to which hand is being used, left or right. Additionally we gain not having to control the position of the hand: we can look at the palm or the back and have the person hold his or her arm at any angle with respect to the camera. All we require is that either the palm or the back of the hand faces the camera and that the fingers are separated. This second requirement can be relaxed in future work; we discuss later in this paper how we expect to achieve this.

We identify the peaks of the fingers in the image by computing the *convex hull* of the hand border, and then computing the *defects* of the convex hull. The convex hull is a descriptor of shape, intuitively explained — in two dimensions — as the form taken by a rubber band when placed around the object in question; an example is given in Figure 3). It is used in computer vision to simplify complex shapes, particularly to provide a rapid indication of the extent of an object.

We now copy the binary image O to a new image C. We will then iteratively seek and eliminate *concave* regions. Intuitively, this can be done by examining the values of the pixels in an arbitrary straight segment with both endpoints residing in white pixels. If any of the pixels along the segment are black, they are colored white, together with any black pixels beneath the segment. This repeated "filling" will continue until no more segments with white end points

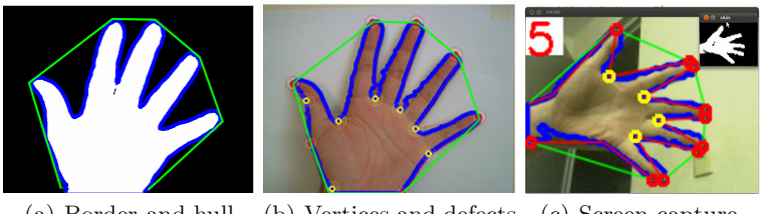
and intermediate black pixels exist. An algorithm for achieving this is given in the textbook of Davies [15]. The resulting white zone in C is now *convex* and the border of that zone — all those white pixels that have at least one black neighbor — form the convex hull of the hand-shape in E. We denote this border by H.

We now proceed to comparing H to E to detect points in which the two differ greatly. First, from H, we compute the *vertices* of the convex hull, that is, the points in which it changes direction. Then, we examine the segments of E between pairs of consecutive vertices of H and find that pixel in each segment that maximizes the distance from H. This maximal distance d_i is called the *depth* of the defect i. The points themselves are called *convexity defects*. Figure 3 shows an example.

From the defect depths, useful characteristics of the hand shape can be derived, such as the depth average μ_d. We use the defects depths, together with the depth average and the total hand length, to count the number of elevated fingers. An above-average depth indicates a gap between fingers (as the finger separations of a person as usually of approximately the samelength), whereas a clearly below-average depth (such as a skin wrinkle or a folded thumb) is not a finger separation. Using the number of defects we can estimate the number of elevated fingers on the hand: an open hand showing five fingers has four convexity defects, whereas a hand showing four fingers has three convexity defects, and so forth.

4 Experiments

We used OpenCV [13] under Python [16] to implement a prototype of the proposed hand-sign detection system. As we wanted the system to be able to run on modest hardware, we performed all our experiments on a netbook with a 1.6 GHz processor and 1 GB of RAM memory, using a web cam with a resolution of 640×480; in future work we will replace the web cam with an augmented



(a) Border and hull (b) Vertices and defects (c) Screen capture

Fig. 3. On the left, the background (in black), the hand-shape region O (in white), the hand border E (in blue), and the convex hull H (in green). On the middle, we add the vertices of the convex hull (in red) and the convexity defects (in yellow). On the right, a screen capture of the implemented prototype for the hand-sign detection tool.

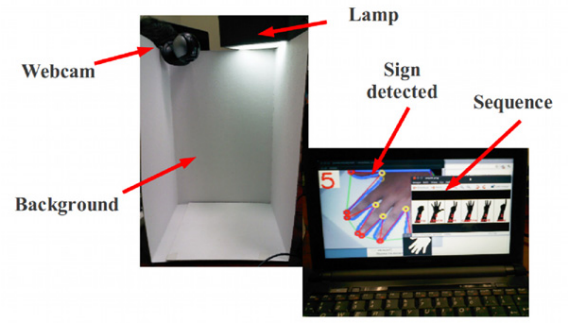


Fig. 4. The experimental setting: our arrangement for controlled background, fixed camera position, and constant illumination. The camera position was chosen to be similar to that of the augmented-reality headset that will be used in future work.

reality headset, using the visor's camera for capturing the image and the on-eye screens for presenting the user with the signs to perform.

The present prototype operates in real time and indicates on screen the detected sign; Figure 3(c) shows a screen capture of the resulting detection, not shown to the user but recorded for the analysis of the experiment — the user only sees the assigned sequence (cf. Figure 5) on a separate computer.

4.1 Experimental Setup

We carried out experiments with users to evaluate the functionality of the proposed gesture detection algorithm. We requested the users to carry out a series of gestures in front of the web cam for the system and measured whether the detection was successful; an observer recorded whether the output produced by the algorithm corresponds the actual gesture being made. The lighting, camera position, and image background were controlled, as illustrated in Figure 4; we hope to relax these unrealistic requirements in future work, incorporating the system into a wearable augmented-reality device.

The user was shown — on a computer screen (see Figure 5 for an example) — a randomly permuted sequence of hand signs to perform, one at a time. We instructed the users to take three-second pause between signs. Each sequence was performed once with the right hand and then again with the left hand. Each user carried out five different sequences, once on each hand.

4.2 Results of User Experiments

We evaluated the prototype with seven users; each performed five sequences of signs with both hands (each sequence is composed of six gestures from zero to five). Therefore, each user performed 60 signs, giving us a total of 420 sign-detection attempts. Table 1 shows the percent of signs correctly detected, separated by sign made and the hand used by the user. The users were not given

Table 1. The percentage of correctly identified signs in the user experiment

Hand used	**Sign detected**						
	0	1	2	3	4	5	**Total**
Right hand	52%	56%	84%	76%	92%	92%	**75.3%**
Left hand	84%	32%	48%	88%	84%	80%	**69.3%**
Total	68%	44%	66%	82%	88%	86%	**75.3%**

any explanation of how the algorithm functions, but could observe the detection while performing the signs. In future experiments, we plan to hide the detection in itself from the user, only displaying the number corresponding to the detected sign.

In total, 75.3% of the signs were correctly detected; the signs for numbers three, four, and five have the highest accuracy and present low variation between hands. The sign for number one, however, has the lowest detection percentage. Also, signs for zero, one, and two show variability according to the hand used; we believe this to be a result of a change in the angle in which the user holds the primary hand compared to the secondary hand.

The sign-detection algorithm works correctly a majority of the time, under the conditions used in our experiments. User observation helped us notice that the primary cause for incorrect sign detection was the particular form in which each user performs the sign: sometimes, for example, the fingers were very closed to each other. Some examples are shown in Figure 6. We discuss a possible work-around to this problem as part of future work in the next section.

Fig. 5. An example of a sequence screen that was presented — as is — for a user on a computer screen; all sequences consisted in a random permutation of the six signs implemented in the system

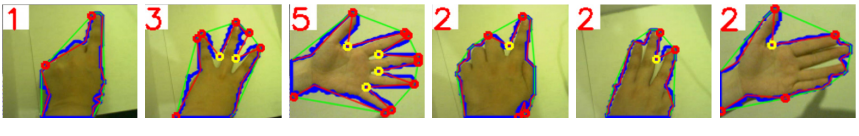


Fig. 6. Some examples of correct and incorrect detection from the user experiments. First three from the left correspond each to a correctly detected sign (one, three, and five), whereas the last three on the right are incorrect detections of one, three, and five.

5 Conclusions and Future Work

We present a method for detecting hand gestures based on computer-vision techniques, together with an implementation that works on a ordinary web cam. The method combines skin-color filtering, border detection, convex-hull computation, and a rule-based reasoning with the depths of the convexity defects. We report user experiments on the detection accuracy of the developed initial prototype, detecting correctly three in four hand signs made on either hand, in a controlled environment. In future experiments, the web cam and the computer screen indicating the signs to perform will be replaced with an augmented-reality headset.

Also, as future work, we plan to add in the sign detection phase an estimate of the width of each finger. This allows us to determine whether a single finger is elevated at that position or whether multiple fingers are elevated but held together, eliminating the need for explicit finger separation while performing the sign. The finger width can be automatically calibrated for each person by measuring the width of the hand base itself and assuming that anything that has the width between one sixth and one fourth of the base width is a single finger. The number of fingers in a wider block can be estimated as the width of the block (computable from the points used for finger counting at present) divided by one fifth of the base width, rounded down to the preceding integer value. We expect this to improve the accuracy of the detection system, as well as ease the cognitive burden of the end user as it will no longer be necessary to keep the fingers separate, something that one easily forgets.

Another aspect that needs to be addressed in future work is the sensibility of the system to lighting conditions, as this affects the skin-color filtering, particularly with reflections and shadows. As the system will ultimately be used on an augmented-reality system, the hand will always be on the foreground, and as the user is looking at the hand, likely in an acceptable angle, but the lighting conditions and the background may vary greatly. In future versions, a light filter will be added to reduce the bright spots and the shadows on the hand itself. We are presently investigating methods to detect the wrist and/or arm on the image border to ease the separation of the hand from the background.

Also the effect of the angle in which the hand is held is object of further study; we expect the system to function for wide range of angles, once the heights of the finger separations and the finger widths are adequately auto-adaptive. To achieve this, we will record video of users performing the signs while turning their hands and analyze these videos off-line. This also allows for incrementing the number of users and signs to the realm of statistical testing as well as comparing the proposed algorithm with existing work of Malima et al. [11], using the same video sequence as input to both our method and alternatives proposed in the literature.

A further possibility for expanding the set of signs detected is to permit the user to combine a series of hand signs into a single message, as we do when representing numbers: joining digits into a sequence. Here the challenge is to distinguish the pauses between signs that form a sequence and the pauses between sequences, as different users will attempt to apply a different rhythm when interacting with such a system.

References

1. Wachs, J.P., Kölsch, M., Stern, H., Edan, Y.: Vision-based hand-gesture applications. Commun. ACM 54, 60–71 (2011)
2. Mahmoud, T.M.: A new fast skin color detection technique. World Academy of Science, Engineering and Technology 43, 501–505 (2008)
3. Vezhnevets, V., Sazonov, V., Andreeva, A.: A survey on pixel-based skin color detection techniques. In: Proceedings of GraphiCon, pp. 85–92 (2003)
4. Kakumanu, P., Makrogiannis, S., Bourbakis, N.: A survey of skin-color modeling and detection methods. Pattern Recognition 40, 1106–1122 (2007)
5. Parker, J.R.: Algorithms for Image Processing and Computer Vision, 1st edn. John Wiley & Sons, Inc., New York (1996)
6. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X.: Vision-based hand pose estimation: A review. Computer Vision and Image Understanding 108, 52–73 (2007)
7. Garg, P., Aggarwal, N., Sofat, S.: Vision based hand gesture recognition. Engineering and Technology 49, 972–977 (2009)
8. Zabulis, X., Baltzakis, H., Argyros, A.: Vision-based hand gesture recognition for human-computer interaction. In: Stephanidis, C. (ed.) The Universal Access Handbook, pp. 1–56. Lawrence Erlbaum Associates, Inc. (2009)
9. Hassanpour, R., Wong, S., Shahbahrami, A.: VisionBased Hand Gesture Recognition for Human Computer Interaction: A Review, pp. 125–134. IADIS (2008)
10. Sánchez-Nielsen, E., Laguna, L., Canalís, L.A., Hernández-Tejera, M.: Hand gesture recognition for human-machine interaction. Journal of WCGS 12, 395–402 (2004)
11. Çetin, M., Malima, A.K., Özgür, E.: A fast algorithm for vision-based hand gesture recognition for robot control. In: Proceedings of the IEEE Conference on Signal Processing and Communications Applications. IEEE (2006)
12. Kovac, J., Peer, P., Solina, F.: Human skin colour clustering for face detection. In: EUROCON 2003: Computer as a Tool. IEEE (2003)
13. OpenCV (visited in December 2011), `http://opencv.willowgarage.com/`
14. Crampton, S.C., Betke, M.: Counting fingers in real time: A webcam-based human-computer interface game applications. In: Proceedings of the Conference on Universal Access in Human-Computer Interaction, pp. 1357–1361 (2003)
15. Davies, E.R.: Machine Vision: Theory, Algorithms, Practicalities. Morgan Kaufmann Publishers Inc., San Francisco (2004)
16. Python (visited in December 2011), `http://www.python.org/`