Hand-gesture recognition using computer-vision techniques

David J. Rios-Soria
Universidad Autónoma de Nuevo León (UANL)
San Nicolás de los Garza, NL, Mexico
david.j.rios@gmail.com

Satu E. Schaeffer
Universidad Autónoma de Nuevo León (UANL)
San Nicolás de los Garza, NL, Mexico
elisa.schaeffer@gmail.com

Sara E. Garza-Villarreal
Universidad Autónoma de Nuevo León (UANL)
San Nicolás de los Garza, NL, Mexico
saraelena@gmail.com

ABSTRACT

We use our hands constantly to interact with things: pick them up, move them, transform their shape, or activate them in some way. In the same unconscious way, we gesticulate in communicating fundamental ideas: 'stop', 'come closer', 'over there', 'no', 'agreed', and so on. Gestures are thus a natural and intuitive form of both interaction and communication. Gestures and gesture recognition are terms increasingly encountered in discussions of human-computer interaction. We present a tool created for human-computer interaction based on hand gestures. The underlying algorithm utilizes only computer-vision techniques. The tool is able to recognize in real time six different hand gestures, captured using a webcam. Experiments conducted to evaluate the system performance are reported.

Keywords: Hand-gesture recognition, computer vision, human computer interaction.

1 Introduction

There are situations in which it is necessary to interact with a system without touching it. The reasons include dirty hands (when repairing a motor, for example), hygiene (to indicate the desired water temperature when washing hands in a public bathroom), and focus of attention (not wishing to redirect the sight towards the controls when operating delicate equipment or interacting with an augmented-reality scenario). The use of voice commands as an alternative to touch-based controls, such as keyboards, buttons, and touch screens, requires a quiet environment and natural language processing; voice commands are, additionally, language-specific and sensitive to dialects and to speech impediments. Another alternative is remote control through *gesture recognition*, also known as remote control "with the wave of a hand". Common applications for this kind of control involve medical systems —provide the user sterility to avoid the spread of infections—, entertainment, and human-robot interaction [WKSE11].

The option explored in this work, *computer vision for gesture recognition*, has advantages over touch-based controls and voice commands. Our proposed hand-gesture detection algorithm works in real time, using basic computer-vision techniques such as filters, border detection, and convex-hull detection; in addition, it only requires a standard webcam, does not need special markers on the hand, can detect the hand regardless of its position (upside down, backwards, leaned to the left or right), and is easily extended for detecting two hands at the same time.

To test this approach, user experiments were carried out and two applications that use our gesture-detection system were developed. In the first application, the detected gestures are used as commands for interaction with a GPS device; in the second one, the detected gestures control the movements of a robot.

This document is organized as follows: Section 2 discusses background for this work and Section 3 reviews related work; Section 4 presents the details of our algorithm for hand-gesture recognition, which is able to recognize six different gestures in real time. Section 5 discusses our prototype implementation and user experiments, and Section 7 offers conclusions and discusses future directions.

2 Background

The use of the hand as an input device is a method that provides natural human-computer interaction. Among the challenges of human-computer interaction is the creation of user-friendly interfaces that use natural communication. Sophisticated applications such as virtual environments or augmented-reality systems should pro-

vide effective human-computer interaction for applications involving complex tasks. In these applications, users should be supplied with sophisticated interfaces allowing them to navigate within the system, select objects, and manipulate them.

The use of computer vision for human-computer interaction is a natural, non-intrusive, non-contact solution. Computer vision can be used for gesture detection and classification, and various approaches have been proposed to support simple applications. To recognize hand gestures using computer vision, it is first needed to detect the hand on an image or video stream. Hand detection and pose estimation involve extracting the position and orientation on the hand, fingertip locations, and finger orientation from the images. Skin-color filtering is a common method for locating the hand because of its fast implementation. Skin-color filters rely on the assumption that the hand is the only skin-colored object. Gesture classification is a research field involving many machine-learning techniques such as neural networks and hidden Markov models [SP09].

However, hand-pose estimation is still a challenge in computer vision. Several open problems remain to be solved in order to obtain robustness, accuracy, and high processing speed. The need of an inexpensive but high-speed system is rather evident.Development of these systems involves a challenge in the research of effective input/output techniques, interaction styles, and evaluation methods [EBN+07].

3 Related work

There are several areas where the detection of hand gestures can be used, such as device interaction, virtual-object interaction, sign-language recognition, and robot control. Wachs et al. [WKSE11] present some examples of applications such as medical assistance systems, crisis management, and human-robot interaction. In the following subsection we present some examples of gesture-based interaction systems.

3.1 Device interaction

There are works related to electronic device interaction; for example, Stergiopoulou et al. [SP09] use self-growing and self-organized neural networks for hand gesture recognition. Another example is *Finger counting* [CB03] a simple human-computer interface. Using a webcam, it interprets specific hand gestures as input to a computer system in real time.

The *UbiHand* [AM06b] is an input device that uses a miniature wrist-worn camera to track finger position, providing a natural and compact interface. A hand model is used to generate a 3D representation of the hand, and a gesture recognition system interprets finger movements as commands. The system is a combination of a pointer position and non-chorded keystroke input device to track finger position [AM06a].

An interactive screen developed by The Alternative Agency[1] in UK is located in a department store window (Figure 1). The *Orange screen* allows interaction just by moving the hands in front of the window without the need to touch it.

Figure 1: The world's first touchless interactive shop window

Lenman et al. [LBT02] use gesture detection to interact with electronic-home devices such as televisons and DVD players.

MacLean et al. [MHP+01] use hand-gesture recognition for real-time teleconferencing applications. The gestures are used for controling horizontal and vertical movement as well as zooming functions. Schlömer et al. [SPHB08] use hand-gesture recognition for interaction with navigation applications such viewing photographs on a television, whereas Roomi et al. [RPJ10] propose a hand-gesture detection system for interaction with slideshow presentations in PowerPoint. The gesture-detection system presented in Argyros et al. [AL06] allows to control remotely the computer mouse.

Sixthsense [MMC09] is a system that converts any surface into an interactive surface. In order to interact with the system, hand gesture recognition is used. In the *Sixthsense* system, color markers are used in the fingers to detect the gestures.

3.2 Virtual object interaction

Gesture detection can be used for interaction with virtual objects; there are several works that show applications for this scenario.

Hirobe et al. [HNW+09] have created an interface for mobile devices using image tracking. The system tracks the finger image and allows to type on an in-air keyboard and draw 3D pictures.

HandVu [Kol10] is a hand-gesture vision-based recognition system that allows interaction with virtual objects (Figure 2) *HandVu* detects the hand in a standard posture, then tracks it and recognizes key postures, all in real-time and without the need for camera or user calibration. Although easy to understand, the used gestures are not natural.

[1] http://www.thealternative.co.uk/

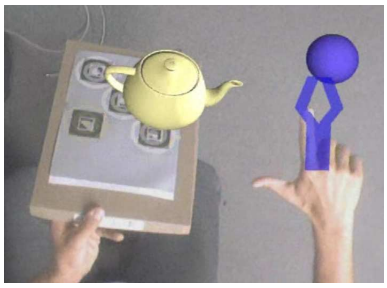


Figure 2: The gestures used in the *Handvu system [Kol10] are not natural gestures.*

Wobbrock et al. [WMW09] propose a series of gestures in order to make easier the use of interactive surfaces. Wachs et al. [WSE+06] use real-time hand gestures for object and window manipulation in a medical data visualization environment.

3.3 Sign language recognition

Zahedi et al. [ZM11] create a system for sign language recognition based on computer vision. Wang et al. [WP09] present a work where hand gesture detection is used in three aplications: animated character interaction, virtual object manipulation, and sign language recognition.

3.4 Robot-control

Malima et al. [ÇMÖ06] use hand-gesture detection for remote robot-control. They have noted that images taken under insufficient light (especially using the webcam) have led to incorrect results. In these cases the failure mainly stems from the erroneous segmentation of some background portions as the hand region.

4 Theory

Our proposed algorithm performs hand-gesture recognition by utilizing computer-vision techniques and is able to recognize six different gestures in real-time. The processing steps included in the algorithm are explained in detail in the following subsections.

4.1 Hand recognition

Hand-recognition systems are based on the processing of 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 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 color values). In this work, we employ the latter approach, which is somewhat sensible to high variations of skin color (the problematic cases being very pale and very dark-skinned users). This 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; in our work we use the average over ten 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 or reflect light are excluded. Hence we need to apply several processing steps; first to extract the hand and then to identify the gesture that the user is currently making.

4.2 Skin-color filtering

Skin color has proven to be a useful and robust cue for face detection, localization, and tracking [Mah08, VSA03, KMB07]. 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 of the color of a given pixel to a defined value representing skin tone. The specific distance metric employed depends on the skin-color modeling method. An obvious advantage of such methods is the simplicity of the skin-detection rules that enables the implementation of a very fast classifier [VSA03].

Colorimetry, computer graphics, and video-signal transmission standards have given birth to many *color spaces* with different properties. A wide variety of them has 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. This remains one of the most widely-used color spaces for processing and storing of digital image data.

YC_BC_R is a family of color spaces used as a part of the color-image pimage pipeline in video and digital photography systems. Y is the *luma component*, sometimes called luminance, that represents the brightness in an image. C_B and C_R are the blue-difference and red-difference chroma components; chroma is the signal used in video systems to convey the color information of the picture

In contrast to RGB, the YC_BC_R color space is luma-independent, resulting in a better performance. YC_BC_R is not an absolute color space; rather, it is a way of encoding RGB information. The actual color displayed depends on the actual RGB primaries used to display the signal.
The hand-gesture detection algorithm uses skin-color detection. The skin-color filter used in our work can also be used for face detection, localization, and tracking of persons in videos.

Denote by I be the entire input image, and by I_Y, I_{C_B} and I_{C_R} the luma, blue, and red components of the image, rspectibly 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 $p^Y_{i,j}$, $p^{C_B}_{i,j}$, and $p^{C_R}_{i,j}$. For all components $C \in \{Y, C_B, C_R\}$, 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 [KPS03] that classifies each pixel as skin or non-skin individually. 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 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 —heuristically, based on existing literature— as skin if all of the following conditions simultaneously apply:

1. The luma component exceeds its corresponding threshold value:
$$p^Y_{i,j} > 80. \tag{1}$$

2. The blue and red components are within their corresponding ranges:
$$\begin{array}{rcccl} 85 & < & p^{C_B}_{i,j} & < & 135, \\ 135 & < & p^{C_R}_{i,j} & < & 180. \end{array} \tag{2}$$

We write $S(p_{i,j}) = \top$ if the pixel $p_{i,j}$ passes the filter, and $S(p_{i,j}) = \bot$ if it does not fulfill the above conditions.

We then create a new binary image B of the same dimension $w \times h$ (cf. Figure 3 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:

$$b_{i,j} = \begin{cases} 1, & \text{if } S(p_{i,j}) = \top, \\ 0, & \text{if } S(p_{i,j}) = \bot. \end{cases} \tag{3}$$

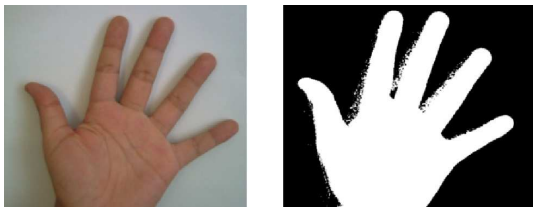


Figure 3: On the left, an original image I. On the right, the resulting binary image B after applying the skin-color filter defined by Equations 1-2.

4.3 Edge detection

Using the 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 edge of the hand shape in the image. *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.

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 [Par96].

There are two main methods 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).

At this point, we have the set of contour pixels and need to determine the connected components of the contour, meaning that we must compute the connected

sets of edge points. To create a connected set we select one contour pixel as a seed and recursively add to the set pixels that are also contour pixels and are adjacent to at least one pixel in the set, until there are no more adjacent contour pixels. If there are contour pixels left, then we select another contour pixel as a seed to create a new connected set; we repeat iteratively until all the contour pixels are in a connected component.
In our case, we assume the hand to be in the image foreground, 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.

4.4 Convex hull and convexity defects

At this point, we have identified the edge of the hand in the image. We now proceed to determining which hand gesture is being made in the image. The way in which this is done depends on the the type of hand gestures supported by the system —no single design is adequate for all possible hand positions–. The gestures that we wish to detect are shown in Figure 4.

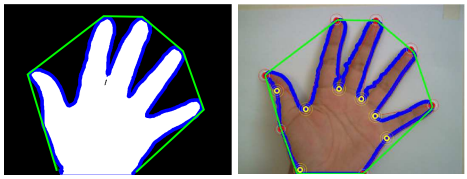
Figure 4: The gestures used in our proposed system that correspond to the numbers from zero to five. Note that the separation of the individual fingers is relevant to the detection of these gestures.

As our gestures 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. [CB03, ÇMÖ06]). 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 nearly 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 edge. The convex hull is a descriptor of shape, is the smallest convex set that contains the edge; 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 5). 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 text book of Davies [Dav04].

The resulting white zone in C is now *convex* and the edge 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 edge by H.



(a) Edge and hull. (b) Vertices and defects.

Figure 5: On the left, the background (in black), the hand-shape region O (in white), the hand edge E (in blue), and the convex hull H (in green). On the right, we add the vertices of the convex hull (in red) and the convexity defects (in yellow).

We now proceed to comparing H to E to detect the *defects*, 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 5 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 defect 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, whereas a clearly below-average depth 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.

5 Material and methods

We used OpenCV[2] under Python[3] to implement a prototype of the proposed hand-gesture 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 webcam with a resolution of 640×480 pixels. The prototype operates in real time and indicates on the screen the detected gesture; Figure 6 shows a screen capture.

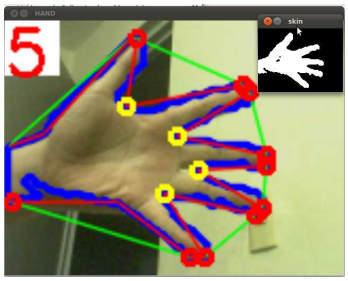


Figure 6: A screen capture of the implemented prototype for the hand-gesture detection tool.

5.1 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 webcam and measured whether the detection was successful. An observer recorded whether the output produced by the algorithm corresponded to the actual gesture being made. The lighting, camera position, and image background were controlled, as illustrated in Figure 7. We hope to relax these requirements in future work, as the proposed method is designed to accommodate a less restricted use setting.

The user was shown a gesture sequence —on a computer screen (see Figure 8 for an example)—. Each gesture sequence contains a randomly permuted sequence of hand gestures to perform. The sequence was available on the screen while the user performed the gestures one at a time. We instructed the users to a take three-second pause between gestures. Each sequence was performed once with the right hand and then again with the left hand. When the user finished to perform the last gesture in the sequence, a new random sequence was show. Each user carried out five different sequences.

[2] http://opencv.willowgarage.com/
[3] http://www.python.org/

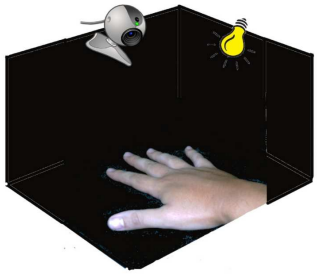


Figure 7: The experimental setting: our arrangement for controlled background, fixed camera position, and constant illumination.

Figure 8: An example of a random gesture sequence assigned to a user.

6 Results of user experiments

We evaluated the prototype with ten users; each performed five sequences of gestures with both hands (each sequence was composed of six gestures from zero to five, in random order). Therefore, each user performed 60 gestures, giving us a total of 600 gesture-detection attempts. Table 6 shows the percent of gestures correctly detected, grouped by the gesture made and the hand used.

Hand used	**Gesture detected**						
	0	1	2	3	4	5	**Total**
Right hand	100%	72%	96%	94%	98%	100%	**93.33%**
Left hand	100%	76%	94%	96%	98%	94%	**93.00%**
Total	100%	74%	95%	95%	98%	97%	**93.17%**

Table 1: Percentage of correctly identified gestures.

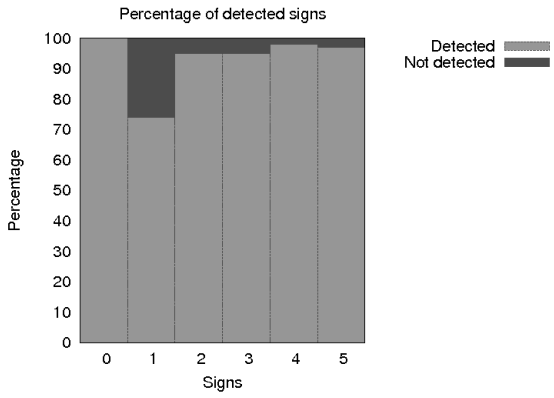


Figure 9: Correctly detected gestures.

In total, 93.1% of the gestures were correctly detected, improving the results for a previous work [RSS12]; the gestures for numbers three, four, and five have the highest accuracy and present low variation between hands. The gestures for number one, however, has the lowest detection percentage. Also, gestures for zero, one, and two show variability according to the hand used. The gesture-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 gesture detection was the particular form in which each user performs the gesture: sometimes, for example, the fingers were very close to each other. Some examples are shown in Figure 10. We discuss a possible work-around to this problem as part of future work in the next section.

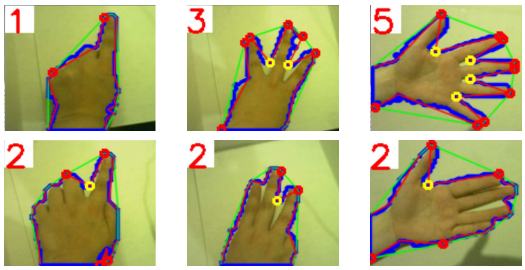


Figure 10: Some examples of correct and incorrect detection from the user experiments. Above, a correctly detected gesture, and below, an incorrect detection of that same gesture. The gestures requested were, from left to right, one, three, and five.

7 Conclusions and future work

We have presented a method for detecting hand gestures based on computer-vision techniques, together with an implementation that works in real time on a ordinary webcam. The method combines skin-color filtering, edge detection, convex-hull computation, and a rule-based reasoning with the depths of the convexity defects. We had reported as well user experiments on the detection accuracy of the developed prototype, detecting correctly nine in ten hand gestures made on either hand, in a controlled environment.

As future work, we plan to add in the gesture 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. The finger width can be 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.

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. We expect these additions 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—.

8 References

REFERENCES

[AL06] Antonis Argyros and Manolis Lourakis. Vision-based interpretation of hand gestures for remote control of a computer mouse. In Thomas Huang, Nicu Sebe, Michael Lew, Vladimir Pavlovic, Mathias Kölsch, Aphrodite Galata, and Branislav Kisacanin, editors, *Computer Vision in Human-Computer Interaction*, volume 3979 of *Lecture Notes in Computer Science*, pages 40–51. Springer, Berlin / Heidelberg, Germany, 2006.

[AM06a] Farooq Ahmad and Petr Musilek. A keystroke and pointer control input interface for wearable computers. In *IEEE International Conference on Pervasive Computing and Communications*, pages 2–11, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[AM06b] Farooq Ahmad and Petr Musilek. Ubihand: a wearable input device for 3D interaction. In *ACM Internacional Conference and Exhibition on Computer Graphics and Interactive Techniques*, page 159, New York, NY, USA, 2006. ACM.

[CB03] Stephen C. Crampton and Margrit Betke. 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*, pages 1357–1361, Crete, Greece, June 2003. HCI International.

[ÇMÖ06] Müdjat Çetin, Asanterabi Kighoma Malima, and Erol Özgür. A fast algorithm for vision-based hand gesture recognition for robot control. In *Proceedings of the IEEE Conference on Signal Processing and Communications Applications*, pages 1–4, NJ, USA, 2006. IEEE.

[Dav04] E. Roy Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[EBN+07] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly.

Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108:52–73, 2007.

[HNW+09] Yuki Hirobe, Takehiro Niikura, Yoshihiro Watanabe, Takashi Komuro, and Masatoshi Ishikawa. Vision-based input interface for mobile devices with high-speed fingertip tracking. In *22nd ACM Symposium on User Interface Software and Technology*, pages 7–8, New York, NY, USA, 2009. ACM.

[KMB07] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40(3):1106–1122, 2007.

[Kol10] Kolsch. Handvu. `www.movesinstitute.org/\textasciitildekolsch/HandVu/HandVu.html`, 2010.

[KPS03] J. Kovac, P. Peer, and F. Solina. Human skin colour clustering for face detection. In *Internacional conference on Computer as a Tool*, volume 2, pages 144–147, NJ, USA, 2003. IEEE.

[LBT02] S. Lenman, L. Bretzner, and B. Thuresson. Computer vision based hand gesture interfaces for human-computer interaction. Technical report, CID, Centre for User Oriented IT Design. Royal Institute of Technology Sweden, Stockhom, Sweden, June 2002.

[Mah08] Tarek M. Mahmoud. A new fast skin color detection technique. *World Academy of Science, Engineering and Technology*, 43:501–505, 2008.

[MHP+01] J. MacLean, R. Herpers, C. Pantofaru, L. Wood, K. Derpanis, D. Topalovic, and J. Tsotsos. Fast hand gesture recognition for real-time teleconferencing applications. In *Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 133–140, Washington, DC, USA, 2001. IEEE Computer Society.

[MMC09] Pranav Mistry, Pattie Maes, and Liyan Chang. WUW - wear ur world: a wearable gestural interface. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4111–4116, New York, NY, USA, 2009. ACM.

[Par96] J. R. Parker. *Algorithms for Image Processing and Computer Vision*. John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 1996.

[RPJ10] S.M.M. Roomi, R.J. Priya, and H. Jayalakshmi. Hand gesture recognition for human-computer interaction. *Journal of Computer Science*, 6(9):1002–1007, 2010.

[RSS12] David J. Rios Soria and Satu E. Schaeffer. A tool for hand-sign recognition. In *4th Mexican Conference on Pattern Recognition*, volume 7329 of *Lecture Notes in Computer Science*, pages 137–146. Springer, Berlin / Heidelberg, 2012.

[SP09] E. Stergiopoulou and N. Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *Engineering Applications of Artificial Intelligence*, 22(8):1141–1158, 2009.

[SPHB08] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a Wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14, New York, NY, USA, 2008. ACM.

[VSA03] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *Proceedings of international conference on computer graphics and vision*, pages 85–92, Moscow, Russia, 2003. Moscow State University.

[WKSE11] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications ACM*, 54:60–71, feb 2011.

[WMW09] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1083–1092, New York, NY, USA, 2009. ACM.

[WP09] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28:63:1–63:8, jul 2009.

[WSE+06] Juan Wachs, Helman Stern, Yael Edan, Michael Gillam, Craig Feied, Mark Smith, and Jon Handler. A real-time hand gesture interface for medical visualization applications. In Ashutosh Tiwari, Rajkumar Roy, Joshua Knowles, Erel Avineri, and Keshav Dahal, editors, *Applications of Soft Computing*, volume 36 of *Advances in Soft Computing*, pages 153–162. Springer, Berlin / Heidelberg, 2006.

[ZM11] Morteza Zahedi and Ali Reza Manashty. Robust sign language recognition system using ToF depth cameras. *Information Technology Journal*, 1(3):50–56, 2011.