Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: 0005-1144 (Print) 1848-3380 (Online) Journal homepage: https://www.tandfonline.com/loi/taut20

Hand features extractor using hand contour – a case study

Antonio Guadalupe Cruz Bautista, José-Joel González-Barbosa, Juan Bautista Hurtado-Ramos, Francisco-Javier Ornelas-Rodriguez & Erick-Alejandro González-Barbosa

To cite this article: Antonio Guadalupe Cruz Bautista, José-Joel González-Barbosa, Juan Bautista Hurtado-Ramos, Francisco-Javier Ornelas-Rodriguez & Erick-Alejandro González-Barbosa (2020) Hand features extractor using hand contour – a case study, Automatika, 61:1, 99-108, DOI: 10.1080/00051144.2019.1686568

To link to this article: https://doi.org/10.1080/00051144.2019.1686568

AUTOMATIKA
2020, VOL. 61, NO. 1, 99–108
https://doi.org/10.1080/00051144.2019.1686568

REGULAR PAPER

OPEN ACCESS Check for updates

Hand features extractor using hand contour – a case study

Antonio Guadalupe Cruz Bautista[a], José-Joel González-Barbosa[a], Juan Bautista Hurtado-Ramos[a], Francisco-Javier Ornelas-Rodriguez[a] and Erick-Alejandro González-Barbosa[b]

[a]Instituto Politécnico Nacional, CICATA Querétaro, Queretaro, México; [b]Instituto Tecnológico Superior de Irapuato ITESI-Irapuato, Guanajuato, México

ABSTRACT
Hand gesture recognition is an important topic in natural user interfaces (NUI). Hand features extraction is the first step for hand gesture recognition. This work proposes a novel real time method for hand features recognition. In our framework we use three cameras and the hand region is extracted with the background subtraction method. Features like arm angle and fingers positions are calculated using Y variations in the vertical contour image. Wrist detection is obtained by calculating the bigger distance from a base line and the hand contour, giving the main features for the hand gesture recognition. Experiments on our own data-set of about 1800 images show that our method performs well and is highly efficient.

ARTICLE HISTORY
Received 1 February 2019
Accepted 17 October 2019

KEYWORDS
Hand; gesture; recognition; fingers; NUI

1. Introduction

A natural user interface (NUI) is a method that allows user to interact with information systems with no help of extra devices; in other words, users can only use the nature systems with which they were born. There are three methods for a NUI; voice, touch and gestures. There are some kind of gestures, some of them are related to the face or the body, but this article focuses only on hand gestures.

NUIs are gaining interest and popularity in all areas. The capacity to give commands without any external tools is a better way of communication between humans and computers. Voice commands are used in artificial intelligence, touchscreens mainly in smartphones avoid the use of keys, and gestures also detect human expressions. As Miriam Novack said [1]: Gestures "reveal information that cannot be found in speech", which means that gestures can contain more information than simple words. Hand gesture recognition has been widely used in human–computer interaction, like in virtual reality, robotics, computer gaming and so on.

Based on their methods, hand gesture recognition technology is mainly divided into two categories: data gloves-based method and computer vision-based method. The computer vision-based method can be classified into two main classes [2]: static gestures [3–7] that are done with a hand or both hands with no movement, and dynamic gestures [8,9,2,10,11,12] that are done with a sequence of hand images following a path or a predefined behaviour. Both static and dynamic gestures are extracted from a sequence of hand images (video), but in the first, the gesture has to be the same and to be in the same position, in the second one, gesture has to be a little different in the form or in the position.

Image recognition of colour-based images, depth images and hand shapes was used in [7,11], but they are not to precise with fingers. There are more specific hand-related devices like the Leap Motion Controller [12–14] that detect mainly finger movements; however, the main drawback with this gadget is when hands are overlapping each other. Also, some applications for hand gestures are proposed in medical environments [15–17], in serious games [13] or in pedagogical practices [10]. It is worth mentioning works that use a mix of Kinect, Leap Motion and Oculus Rift [18,19]. Hand segmentation is the first goal in each case. This can be done using helping tools such as depth cameras [20] or stereoscopic vision [21]. The most common method is to use skin colour detection in RGB cameras [21–29]. Hand segmentation can be done in a non-controlled or a controlled environment. In a non-controlled environment, there are some challenges for skin colour detection, such as wide variety of illumination conditions and skin-like colour objects that appear in the image background. Controlled environment uses a specific place where hand detection can be done assuming that background would be static with no illumination changes.

To prevent the effect caused by complex background and light variety, Junxia et al. [30] propose a method that combines depth information and skin colour using a Kinect sensor. They extracted the candidate area of the

CONTACT José-Joel González-Barbosa jgonzalezba@ipn.mx Cerro Blanco 141, Colinas del Cimatario, 76090 Santiago de Querétaro, Qro, México

hand by skin colour segmentation. Moreover, they set a depth threshold to filter out most of the background. Eventually, they regard hand extracted from the depth image as a mask, and projected it on the image after skin colour segmentation, after that they acquired human hand from the colour image.

The main challenge is to detect hand features. Sometimes some features like the centre of the hand [20,26,28] or the wrist [25] are needed, but the main features in the hands are the fingers. This work proposes a method for fingers detection in a controlled environment with black background.

2. Materials and methods

Hand capture sometimes is realized using markers on the hand [31] also can be done with hand gloves [32] or with gloves with certain colours [6], but these approaches are intrusive. Studies have shown that gloves may have negative effects on manual dexterity, comfort and possibly the range of finger and wrist movements [33].

There are also some expensive options like the head-mounted displays (HMD) that can detect hand gestures. Some examples are Google Glass [34] and Microsoft HoloLens [35], the Oculus also can be included with a mounted Leap Motion, but we are interested in a more natural way for the user, without using wearable or touching devices. These approaches are limited in accuracy and precision of current vision-based devices for tracking hand movements, as well as their inherent issue of visual occlusions.

In this article we are using low cost hardware and optimizing all memory resources, processor operations and so on, so we are also focusing on a fast and cheap option. The computer used to test this approach is a Dell Inspiron 5555 in a Linux Ubuntu environment using C++ and OpenCV for CodeBlocks.

In Figure 1 is depicted the block diagram of the process, from image acquisition to fingers and wrist detection. In the first step, we start by simultaneously acquiring the different views of the object of interest (one or two hands). Images then pass through different, rather standardized processes. As usual, these processes allow faster hand detection. Because we are using different points of view of the same object we can calculate the 3D position of the arm, wrist and fingers of each hand (left and right) for each view. If no points are obtained, the returned values are NULL.

There are some other considerations for the calculation of the 3D points. For example, if the process of the top image detects a wrist and the process of the right image does not detect the wrist, there is not enough information to calculate the 3D point and the data are discarded, so no hand is detected.

Considering that the work area is a desk, there are two cameras, one on the rigt side of the desk and

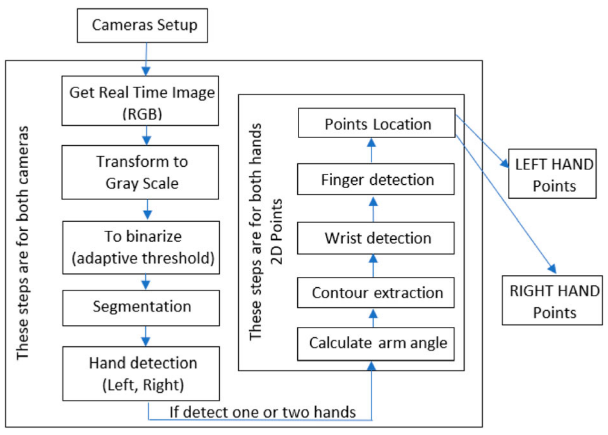
Cameras Setup
These steps are for both cameras
Get Real Time Image (RGB)
Transform to Gray Scale
To binarize (adaptive threshold)
Segmentation
Hand detection (Left, Right)
If detect one or two hands
These steps are for both hands 2D Points
Points Location
Finger detection
Wrist detection
Contour extraction
Calculate arm angle
LEFT HAND Points
RIGHT HAND Points

Figure 1. All steps from image acquisition to fingers and wrist detection. All cameras are connected to the same computer, they capture the RGB image sequence in real time (also can be a video). The process needs a feed of simultaneously captured images (top and right), and each image is transformed to grey scale. Next step is to binarize the images with a threshold, with the result, the segmentation process begins to work. The number of objects detected is the number of hands (0, 1, 2). If detects one or two hands, we can calculate the arm angle, if no hand is detected, the process take another pair of images and the process continue over and over again. Next the Sobel filter is applied to extract the contours. The obtained points of the contour process give us the possibility to detect the wrist. After the wrist detection, a finger detection algorithm is applied to obtain finger points in 2D. The combination of the data give us the 3D points of each hand.

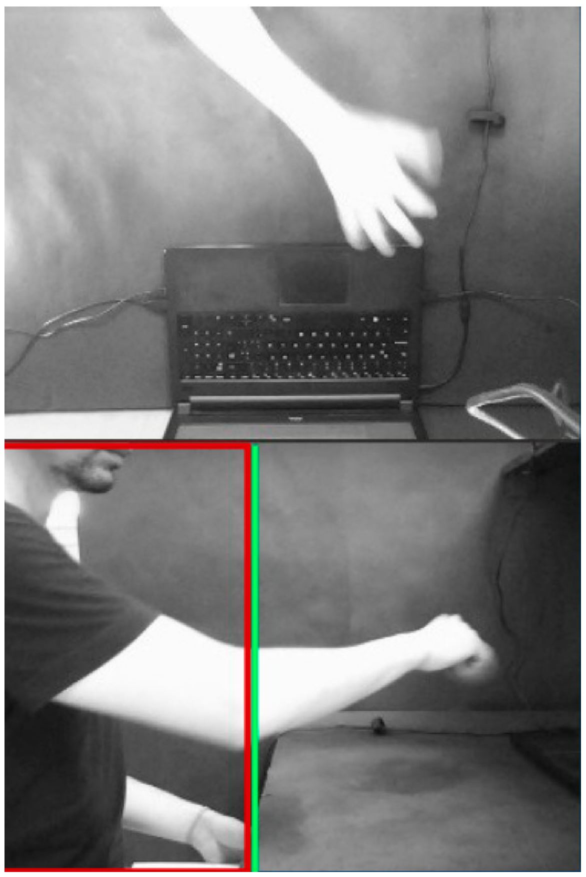

Figure 2. Upper and right cameras with black background. Bottom image shows a rectangle that enclose non-significant data, all the right part are the significant data.

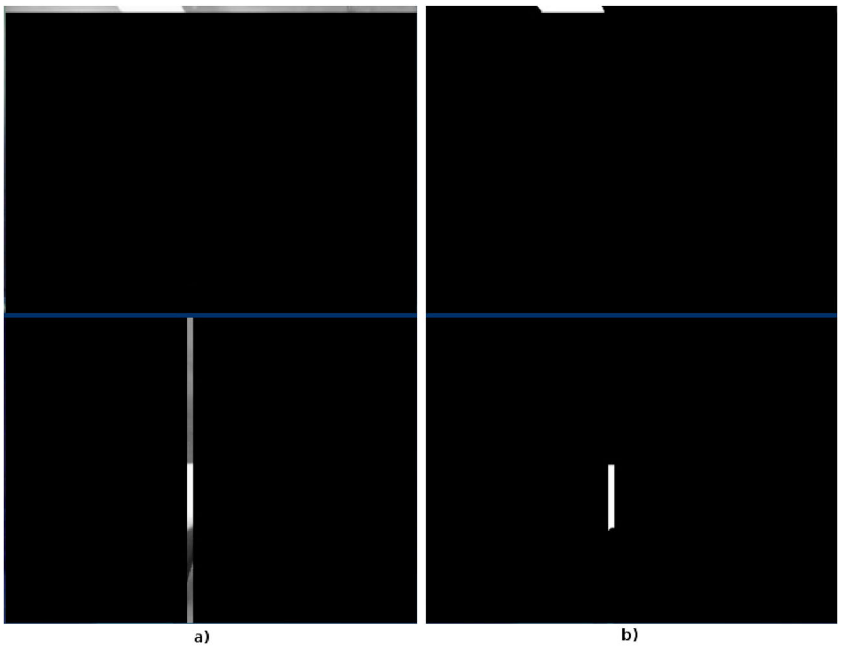

Figure 3. Calculus for adaptive threshold in a region of five pixels. (a) Region of five pixels to find the threshold in the image. The brightest pixels are the pixels of the hand. (b) The result for the process after find the threshold.

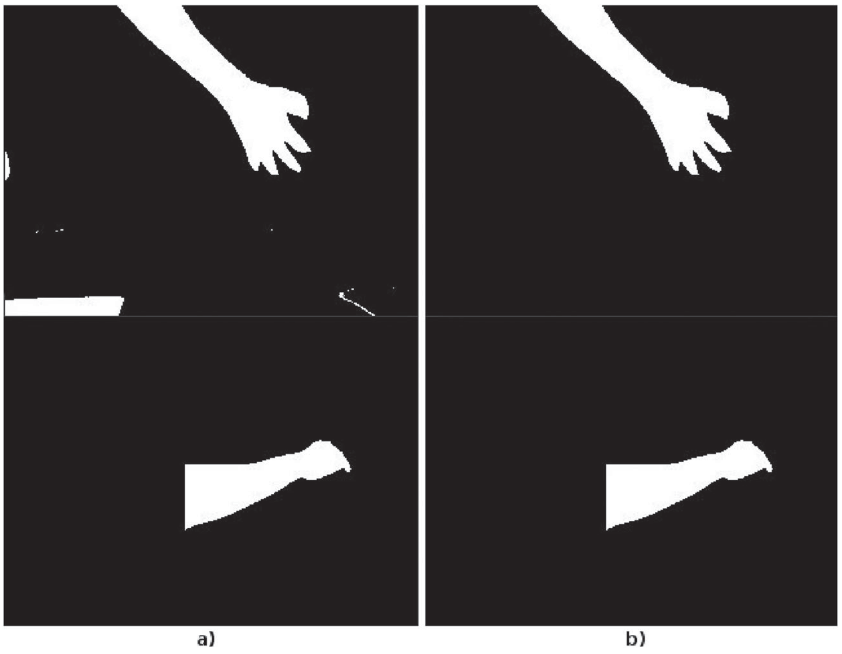

Figure 4. Segmented image. (a) Images can have noise or objects different of hands. Undesired pixels can be removed taking all continuous white pixels starting off the ones obtained in the five pixels region. (b) Images after the process are cleanest segmented images.

another is placed on the upper side. The images are captured with 320 × 240 resolution with a black background. The black background is used to remove shadows of the hand over the surfaces and to take the brightest pixels of the image as the hand itself. Figure 2 shows images from the upper and right cameras with non-significant data.

Figure 2 shows that in the upper view, hands are always coming out from the upper part of the image, whereas in the right view the hands are always coming out from the green line. We can calculate an adaptive threshold in a region of five pixels where the hand is coming out in each view. The purpose of having this region is to simplify computer resources; instead of finding the threshold in all the image pixels, it is better to work in a small region (5 pixels length). The resulting threshold is applied to the whole image. Figure 3 shows this region and the result after the process.

After this step, images can contain noise or undesired objects. This can be solved taking all continuous

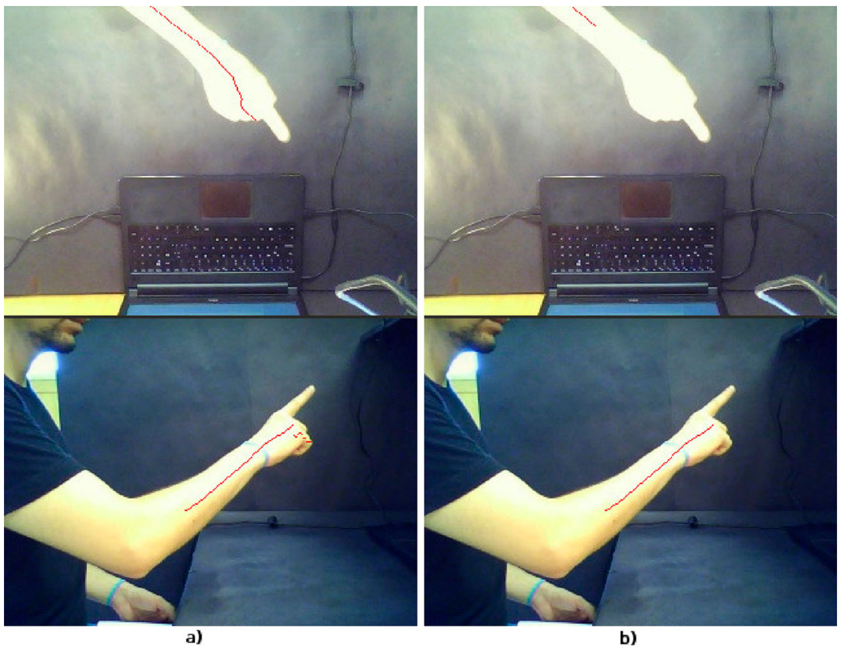

Figure 5. Arm pixels. (a) Images show the results for finding the central points in binarized segmented images (points inside the arm and the hand). (b) Images show the result after applying a filter.

white pixels starting from the pixels obtained in the previous step. Figure 4(b) shows how undesired pixels are removed.

The next step is to detect arm pixels. We know that each image has rows (Y coordinates) and cols (X coordinates). In the upper view, we divide the binarized segmented image row by row (every row has many X coordinates). In each row we take the first and last X coordinates of the white pixels on the object and the half is an arm pixel. In other words, think as if you have an image of $1\times N$ pixels, where 1 is Y dimension (a row) and N is X dimension (N cols), so you only have a region with white pixels, this can be seen in Figure 3 section b, top image but in that image there is a region of $5\times N$ pixels (5 rows).

Equation (1) shows the calculation for the red X coordinate in each row of the image (Figure 5 top images).

$$x_r = \frac{x_2 - x_1}{2} + x_1 \tag{1}$$

where:

x_r = X coordinate for the arm point

x_1 = first white point in the row of the image from left to right

x_2 = last white point in the row.

The same process is applied in the right view, but instead of rows we take columns and instead of X coordinate we use the Y coordinate (Figure 5 bottom images).

2.1. Linear regression with vectors

We remove points whose distance is greater than n pixels, with this the red points are the closest to a line. With

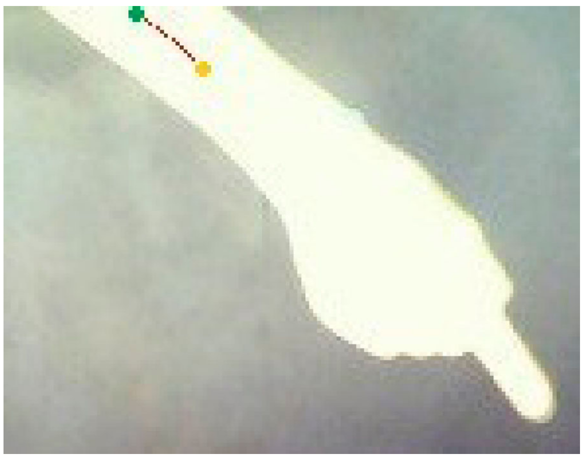

Figure 6. Initial and final points. Initial = Upper Point (y_{initial}, x_{initial}). Final = Lower Point (y_{final}, x_{final}).

the arm points we can use a linear regression to obtain the closest line to all these points and with that line we can calculate the arm angle. To solve this kind of problems, the first thing that comes to our minds is equation of the line, but it only works when we have two points! So, what if we have too many points like in our case? First, let us see Equation (2), which is the equation of the line in the matrix form.

$$\begin{bmatrix} x_j & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = y_j \tag{2}$$

where x_j and y_j, represent the ordinate pair of each point obtained before (red ones). If the first matrix is represented by the letter E, and the third matrix with letter F, we have Equation (3).

$$E \begin{bmatrix} m \\ b \end{bmatrix} = F \tag{3}$$

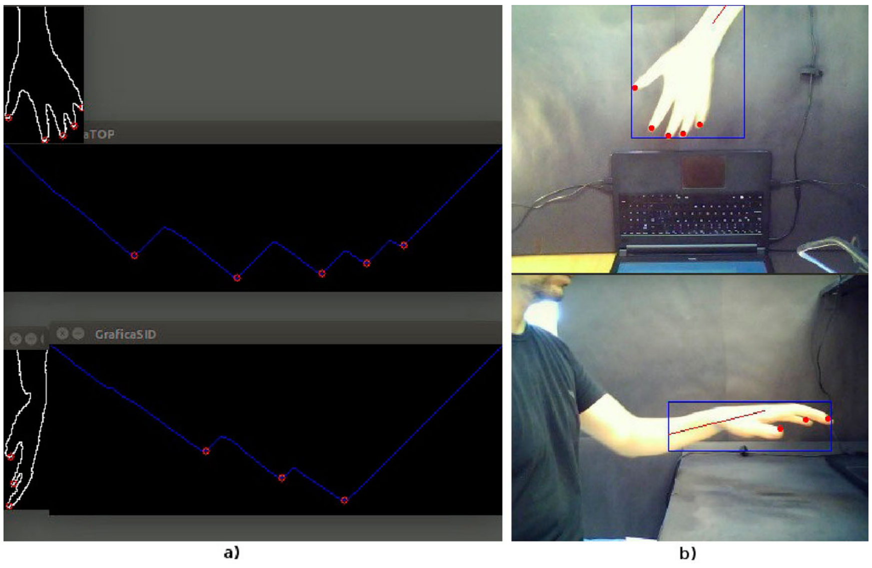

Figure 7. Fingers detection. (a) Images show the Sobel filtered image with Y variations graph. Lower points in the graph show the minimums which are the fingers. (b) Images show the detected fingers with the respective rotation.

The final equation used for linear regression is expressed in Equation (4).

$$\begin{bmatrix} m \\ b \end{bmatrix} = G^{-1}(E^t F) \tag{4}$$

where:

G $= E^t E$.

$E =$ Matrix from $x_j$points.

$F =$ Matrix from $y_j$points.

The left part of Figure 5 shows arm pixels with and without the filter.

With the beginning and ending points of the line we can calculate the arm angle with the formula in Equation (5), see Figure 6.

$$\theta = a\mathrm{Tan}\left(\frac{y_{final} - y_{initial}}{x_{final} - x_{initial}}\right) \tag{5}$$

2.2. *"Y variations" method*

For finding the fingers we need to apply the Sobel filter to the segmented images and rotate θ, so the hands point downwards in both the views. After that the contour is followed and Y variations are printed in a graph, the lower points in the graph are the fingers. "Y variations" mean that, following in order the contour (white pixels one by one), each white pixel has X and Y coordinates, and when there is a different Y coordinate, there is a Y variation. Figure 7 shows the graph in each view and fingers detected are in red colour. They are better explained in Algorithm 1.

For finding the wrist we propose a method in which the Sobel image is divided at the middle to left side and right side. Working with the left side, we need to find a line of reference for finding the left side of the wrist. We need two points to form this line. The first point is the first white pixel in the first row of the image. The second point is the white pixel located more to the left. With the line formed we take five-pixel samples in the line and in the contour and calculate their distances. The bigger distance is the wrist left part. The same process is applied in the right part of the image like a mirror. Two points of the wrist are obtained, and the half of these two points is the wrist. Figure 8 show images for this process.

With the wrist point and finger points, we can calculate the angle of each finger using Equation (5).

Algorithm 1 Calculate "Y" variations (finger points). "I" represents the Sobel filtered image. "V" is a vector that will contain all white pixels, and F are the finger points.

```
  pos ← 0
  for all pixel ∈ I do
    if pixel is white then
      V(pos) ← pixelpoint
      pos ← pos + 1
  end if
end for
V ← sort(V)
F ← getMiminumLocals(V)
return F
```

2.3. *The "convex" method*

This method uses two functions included in OpenCV: convex hull and convexity defects. Figure 9 shows convexity defect results.

The use of the convex hull and convexity defects allow us to find an approximation of the wrist point, the two longer lines of the result of convex hull and the convexity defects of these lines give us two points that can be the wrist. The point with the bigger distance to the respective line is the one with bigger probabilities of being the wrist. The angle of the arm can be defined

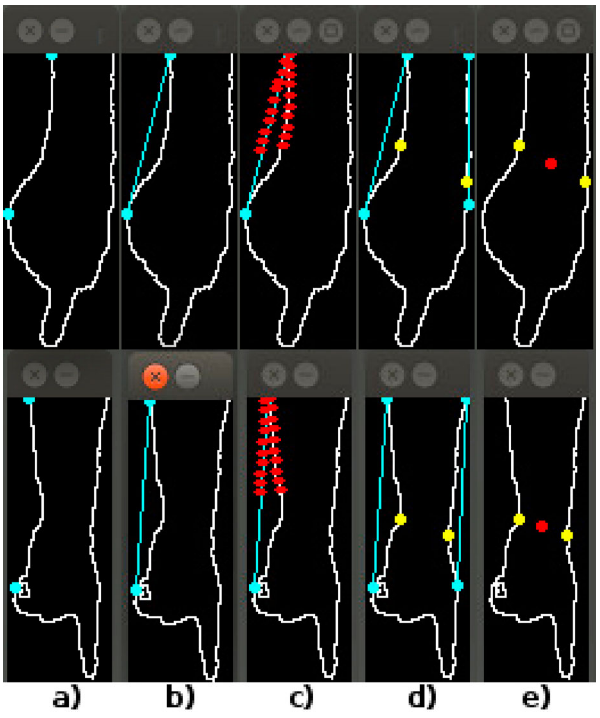

Figure 8. Wrist detection. Working in the left side of image: (a) Bigger points are the first white point and the point more to the left. (b) Both points form a line. (c) Five-pixel samples are taken to calculate distances from contour to line (distances of five pixels). (d) Bigger distance in contour is taken. Same process is applied in right side of images and another point is obtained. (e) The half in these two points is the wrist showed in the image.

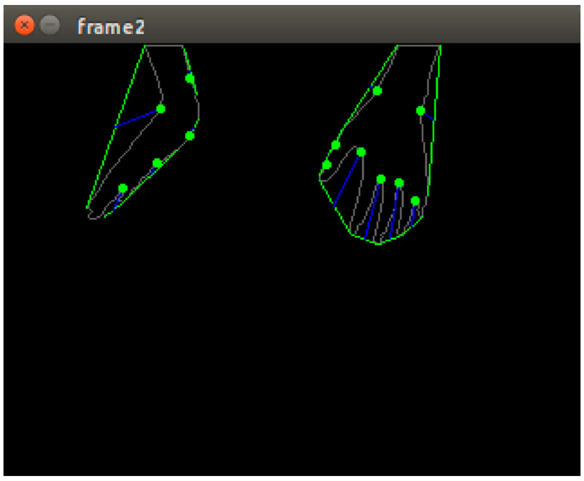

Figure 9. The image shows the points obtained by the convexity defects function (after ignoring duplicated or non-significant points).

as the angle obtained by the line formed by the middle point of the possible wrist points and the x or y line middle point (x for top camera and y for side camera). See Figures 10–12.

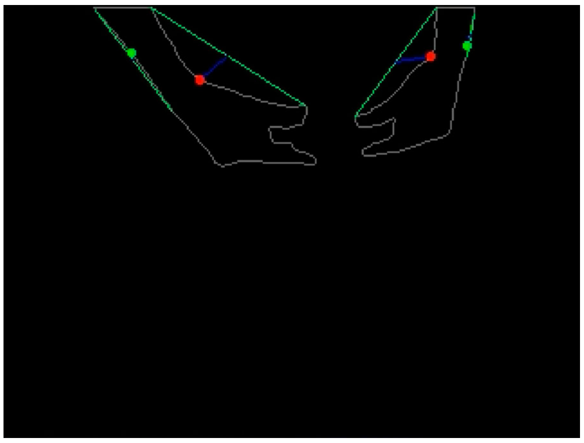

Figure 10. The image shows two points that indicate the possible wrist on both sides. The point with bigger distance has priority.

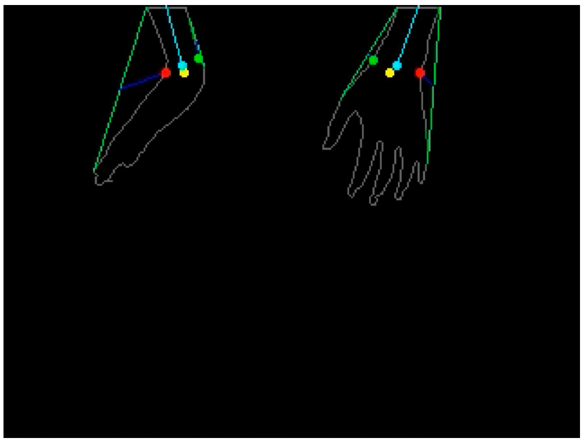

Figure 11. The image shows the points of the possible wrist (points over the contour). The point that is the middle of the possible wrist points has a line, and the last point is only used as reference from the row for the priority one.

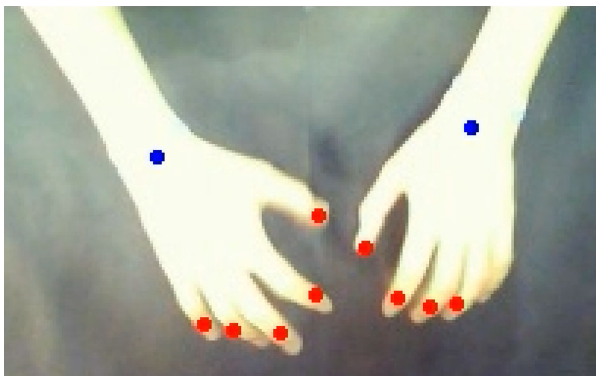

Figure 12. This is the final image for the fingers and wrist detection. You can observe a point over the wrist and the other points are fingers detected.

3. Results and discussion

We work with the algorithm from three videos taken. Each video has about 600 images each. Results are good enough to detect the corresponding interest points, so this is a cheap and fast way to detect them. Figure 13 show many images of hands with finger and wrist detection.

When a user shows a closed hand (no visible fingers), the minimum is considered as a finger. This is because

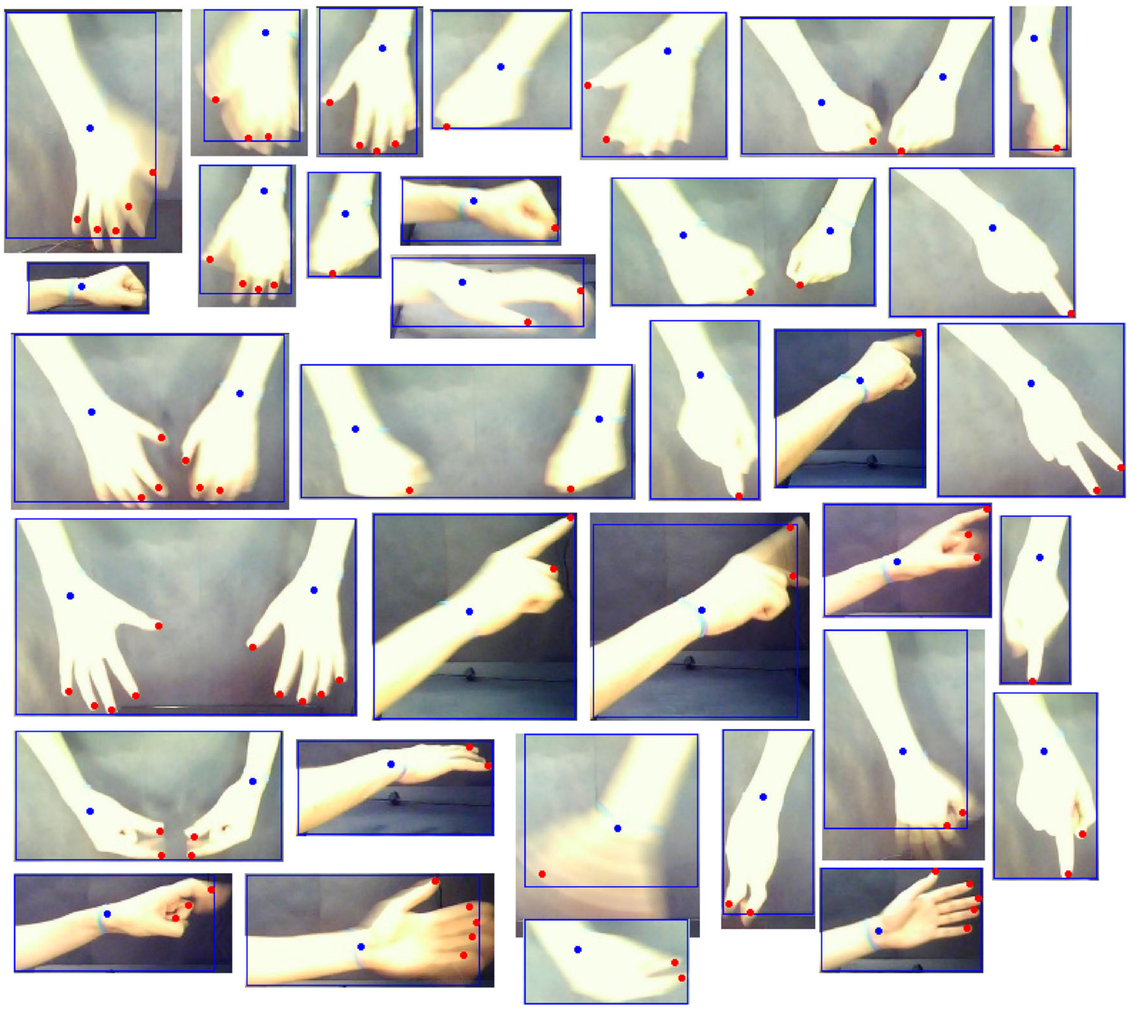

Figure 13. More results of finger and wrist detection. In this collage, we can observe the proposed fingers detection method with hands in different positions.

even if it is not a finger, there is a minimum in the graph (Figure 14). This problem does not always happen but it is a point to improve for better results.

Sometimes they are misplaced wrist points when the hand is not fully visible for the cameras, in other words, when cameras cannot view fingers or fist, or when the wrist is not visible for the camera. Also there is the problem of hand occlusion mainly in the right view. Figure 15 shows images about this kind of problems.

But after all these data, there is the question of which one of the two methods (described in Section II.b and II.c) is faster? Table 1 shows the comparison of these two methods.

4. Conclusions

As we can observe from Table 1, the elapsed time of the proposed method is a little bit faster than the convex method. There are some inconveniences in both the methods, for example, the convex method detects too many points, for this reason it can detect fingers easily but it also has too many bad detected fingers. In both cases, the well-detected percentage is low because

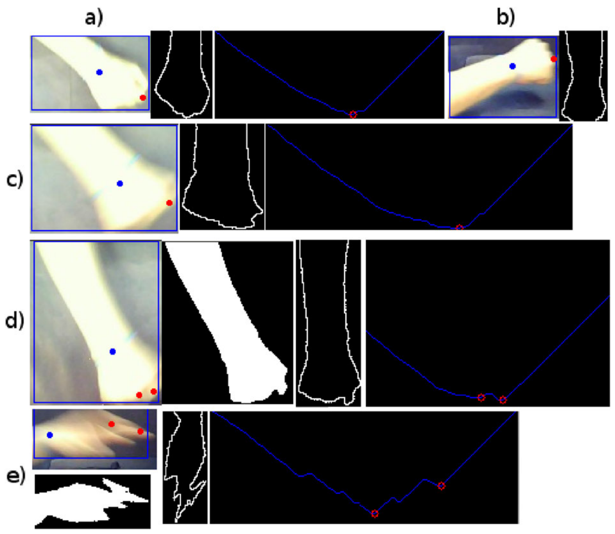

Figure 14. True negatives. In most of the images shown (a, b, c, d), there is a closed hand with no visible finger but the algorithm detects it like a finger (true negative). This is because using the "Y variations method", sometimes a minimum is detected and the algorithm take it like a finger. (e) shows another true negative, but this is because the hand is moving too fast for the camera, and the camera captures the previous position (motion blur).

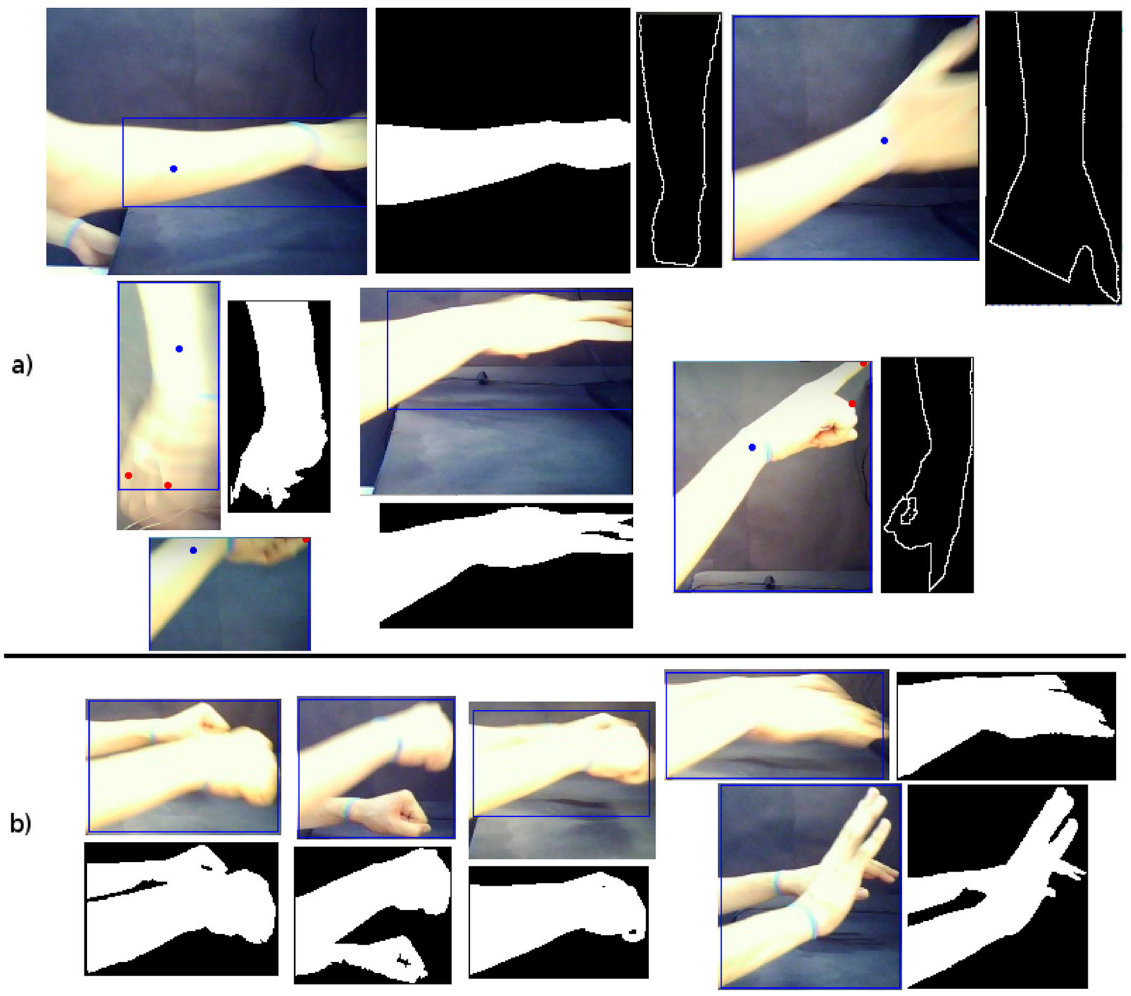

Figure 15. Hands not fully visible and hand occlusion. (a) Images where the hand gets out of the camera's view, lacks of information to work with. (b) The occlusion problem occurs when the user uses both hands, so, in the segmentation process is difficult to separate left from right hand.

Table 1. Comparison of two methods.

	WD	ND	BD	ET (ms)
"Y variation" method	68.086%	25.532%	12.766%	31.472
The "convex" method	78.723%	18.085%	126.60%	32.468

Note: Well Detected (WD), Not Detected (ND), Bad Detected (BD), Elapsed Time (ET).

images had too much motion blur in each image, this can be solved doing the gesture slowly or to use cameras with a bigger frame rate.

The convex method needs a filter for bad detected points, with this the result data will be better than the other, but this will increase the elapsed time. So the proposed method is a low resource method to implement finger detection with good results. It can be used in scenarios where you do not need too much precision on the finger detection but you want speed.

4.1. Future work

The next step after this process is to develop the hand gesture detection. Also, it is necessary to obtain good results with low computational resources. We are working on a pixel to pixel comparison but there are problems when hand gestures are so similar.

We are thinking of a method that uses the hand contour image to obtain interest points and to compare it with base images to detect hand gestures.

To solve the "not fully visible" hand problem, it is necessary that the camera's position will be farther to the user. We are working with different distances of the cameras to have a considerable working area that can detect the hand positions easily.

When the hand gesture detection process is completed, we can work on a Client-Server application and use a Raspberry-Pi to detect the points and hand gestures, send the data to a computer and execute some commands on it.

Some other improvements can be done to the code and the method for fingers and wrist detection. Maybe it will be necessary to find the central point of the hand's palm or to improve the calculation of the points.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

José-Joel González-Barbosa http://orcid.org/0000-0002-6720-8282
Juan Bautista Hurtado-Ramos http://orcid.org/0000-0003-2663-2463

References

[1] Novack MA, Goldin-Meadow S. Learning from gesture: how our hands change our minds. Educ Psychol Rev. 2015;27(3):405–412.
[2] Çağrı Kılıboz N, Güdükbay U. A hand gesture recognition technique for human–computer interaction. J Vis Commun Image Represent. 2015;28:97–104.
[3] Bhuyan M, MacDorman KF, Kar MK, et al. Hand pose recognition from monocular images by geometrical and texture analysis. J Visual Languages Comput. 2015;28:39–55.
[4] Cao J, Yu S, Liu H, et al. Hand posture recognition based on heterogeneous features fusion of multiple kernels learning. Multimed Tools Appl. Oct 2016;75:11909–11928.
[5] Badi H. Recent methods in vision-based hand gesture recognition. Int J Data Sci Analytics. 2016;1(2):77–87.
[6] Lamberti L, Camastra F. Handy: a real-time three color glove-based gesture recognizer with learning vector quantization. Expert Syst Appl. 2012;39(12): 10489–10494.
[7] Yeo H-S, Lee B-G, Lim H. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. Multimed Tools Appl. Apr 2015;74:2687–2715.
[8] Shen X, Hua G, Williams L, et al. Dynamic hand gesture recognition: an exemplar-based approach from motion divergence fields. Image Vis Comput. 2012;30(3): 227–235. Best of Automatic Face and Gesture Recognition 2011.
[9] Elakkiya R, Selvamani K, Kanimozhi S, et al. Intelligent system for human computer interface using hand gesture recognition. Procedia Engineering: International Conference on Modelling, Optimization and Computing (ICMOC). 2012;38:3180–3191.
[10] Liang H, Chang J, Kazmi IK, et al. Hand gesture-based interactive puppetry system to assist storytelling for children. Vis Comput. Apr 2017;33:517–531.
[11] Zhang C, Tian Y. Histogram of 3d facets: a depth descriptor for human action and hand gesture recognition. Comput Vis Image Underst. 2015;139:29–39.
[12] ZaiTi I-A, Pentiuc S-G, Vatavu R-D. On free-hand TV control: experimental results on user-elicited gestures with leap motion. Pers Ubiquitous Comput. Aug 2015;19:821–838.
[13] Sárkány A, Tősér Z, Verö AL, et al. "Maintain and improve mental health by smart virtual reality serious games," in *Pervasive Computing Paradigms for Mental Health: 5th International Conference, MindCare 2015, Milan, Italy, September 24-25, 2015, Revised Selected Papers* (S. Serino, A. Matic, D. Giakoumis, G. Lopez, and P. Cipresso, eds.), p. 220–229, Cham: Springer International Publishing, 2016.
[14] Falcao C, Lemos AC, Soares M. Evaluation of natural user interface: a usability study based on the leap motion device. Procedia Manuf. 2015;3:5490–5495. 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015.
[15] Jacob MG, Wachs JP. Contextbased hand gesture recognition for the operating room. Pattern Recognit Lett. 2014;36:196–203.
[16] Anacleto J, Fels S, Silvestre R. Transforming a paper based process to a natural user interfaces process in a chronic care hospital. Procedia Comput Sci. 2012;14:173–180. Proceedings of the 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2012).
[17] Colombo G, Facoetti G, Rizzi C, et al. Mixed reality to design lower limb prosthesis. Comput Aided Des Appl. 2016;13(6):799–807.
[18] Vitali A, Rizzi C. Acquisition of customer's tailor measurements for 3d clothing design using virtual reality devices. Virtual Phys Prototyp. 2018;13(3): 131–145.
[19] Vitali A, Rizzi C. A virtual environment to emulate tailor's work. Comput Aided Des Appl. 2017;14(5): 671–679.
[20] Liang W, Guixi L, Hongyan D. Dynamic and combined gestures recognition based on multi-feature fusion in a complex environment. The J China Univ Posts Telecommun. 2015;22(2):81–88.
[21] Laskar MA, Das AJ, Talukdar AK, et al. Stereo vision-based hand gesture recognition under 3d environment. Procedia Comput Sci. 2015;58:194–201. Second International Symposium on Computer Vision and the Internet (VisionNet'15).
[22] Priyal SP, Bora PK. A robust static hand gesture recognition system using geometry based normalizations and krawtchouk moments. Pattern Recognit. 2013;46(8):2202–2219.
[23] Gupta A, Sehrawat VK, Khosla M. Fpga based real time human hand gesture recognition system. Procedia Technol. 2012;6:98–107. 2nd International Conference on Communication, Computing & Security [ICCCS-2012].
[24] Huang D-Y, Hu W-C, Chang S-H. Gabor filter-based hand-pose angle estimation for hand gesture recognition under varying illumination. Expert Syst Appl. 2011;38(5):6031–6042.
[25] Premaratne P, Ajaz S, Premaratne M. Hand gesture tracking and recognition system using lucas–kanade algorithms for control of consumer electronics. Neurocomputing. 2013;116:242–249. Advanced Theory and Methodology in Intelligent Computing.
[26] Sharma RP, Verma GK. Human computer interaction using hand gesture. Procedia Comput Sci. 2015;54:721–727. Eleventh International Conference on Communication Networks, ICCN 2015, August 21–23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21–23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21–23, 2015, Bangalore, India.
[27] Lee Y-W. Implementation of an interactive interview system using hand gesture recognition. Neurocomputing. 2013;116:272–279. Advanced Theory and Methodology in Intelligent Computing.
[28] Ozturk O, Aksac A, Ozyer T, et al. Boosting real-time recognition of hand posture and gesture for virtual mouse operations with segmentation. Appl Intell. Dec 2015;43:786–801.
[29] Zhang Y, Wang J, Ye L, et al. A virtual music control system based on dynamic hand gesture recognition. In: Z Pan, AD Cheok, W Müller, M Zhang, editor.

Transactions on Edutainment XIII. Berlin, Heidelberg: Springer Berlin Heidelberg; 2017. p. 74–85.

[30] Junxia B, Jianqin Y, Jun W, et al. Hand detection based on depth information and color information of the kinect In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, p. 4205–4210, May 2015.

[31] Kang J, Zhong K, Qin S, et al. Instant 3d design concept generation and visualization by real-time hand gesture recognition. Comput Ind. 2013;64(7):785–797.

[32] Iliukhin V, Mitkovskii K, Bizyanova D, et al. The development of motion capture system based on kinect sensor and bluetooth-gloves. Procedia Eng. 2017;176:506–513. Proceedings of the 3rd International Conference on Dynamics and Vibroacoustics of Machines (DVM2016) June 29–July 01, 2016 Samara, Russia.

[33] Dianat I, Haslegrave CM, Stedmon AW. Methodology for evaluating gloves in relation to the effects on hand performance capabilities: a literature review. Ergonomics. 2012;55(11):1429–1451. PMID: 22897425.

[34] Nakhla J, Kobets A, la Garza Ramos RD, et al. Use of google glass to enhance surgical education of neurosurgery residents: "proof-of-concept" study. World Neurosurg. 2017;98:711–714.

[35] Syed A, Zakaria A, Lozanoff S. Dark room to augmented reality: application of hololens technology for oral radiological diagnosis. Oral Surg Oral Med Oral Pathol Oral Radiol. 2017;124(1):e33.