HandMouse: a hand tracking and gestures recognition application based on RGB camera

Simone Caldarella, Gaia Trebucchi, and Federico Pedeni

Abstract—Controlling the pointer of a PC using your own hands and a simple camera, instead of the mouse, has been a great challenge for years, due to various problems, including distinguishing the movement of the hand from that of other elements in the background and being able to separate the face from the hands. RGB-depth cameras helped solve these problems, along with the use of various deep learning techniques. However, in this project, developed for the "Signal, Image, Video" course at the University of Trento, we tried to tackle the aforementioned problems without using deep learning and rgb depth camera techniques, relying solely on low and mid level processing techniques. In this paper we will show our method and the results we obtained, combined with some considerations concerning the limitations of this approach.

HandMouse: Gesture Recognition (above) and Tracking (below) phaes.

1 Introduction

The possibility of using a simpler interface to control a computer has been studied in a particularly extensive way in recent years. The paradigm shift was certainly pushed by the development of touch devices, which made it possible to develop more intuitive interfaces than the classic mouse, such as control via gestures or via voice commands. For classic PCs not equipped with a touch screen, the only solution to try to simplify interfacing and that does not require a redesign is to use the camera and your hands to control the pointer on the screen.

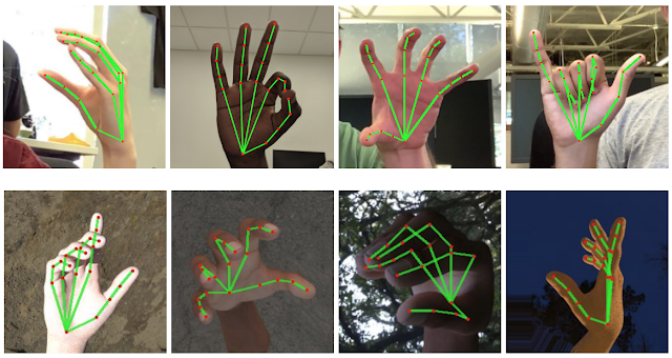
To do this, various difficulties must be faced that can make this type of use uncomfortable or unusable. Among these we find problems related to variations in brightness and color tones, the movement of other elements in the image in addition to the hands, the noise present in the tracking algorithms and the complexity of automatically recognizing a hand inside the image in a real-time context and having a low quality camera typical of laptops available.

In the last 15 years, a lot of research has been done on this and the literature is very large. However, it was soon understood that using purely image processing methods the task was too complicated to be solved by developing robust algorithms. The research therefore shifted first to the use of rgb-depth cameras to be able to separate the foreground (hands) from the background (including the face) and later, thanks to the exponential development of deep learning on an approach based on neural networks.

Currently the state of the art is represented by various standard-bearers, among which the Mediapipe library developed by google [1] stands out, which among the various

- *Email: simone.caldarella@studenti.unitn.it (Simone Caldarella)*
- *Email: gaia.trebucchi@studenti.unitn.it (Gaia Trebucchi)*
- *Email: federico.pedeni@studenti.unitn.it (Federico Pedeni)*

tasks (pose detection, hand detection, objectron, etc ...) optimally solves hand tracking and gestures detection, without requiring excessive computational capacity (executable real-time at at least 15 fps on any laptop).



Our goal in this project was not to overcome the state of the art, but to develop a system, based on a low level approach and strictly linked to the world of image processing, in order to first learn a set of skills required for tackle a problem of this complexity and secondarily understand the intrinsic limits in this type of approaches.

In this paper we will explain the preliminary steps in order to develop this application and the salient points of the final application, ending then by evaluating the limits and problems encountered, overcome and not.

2 Our work

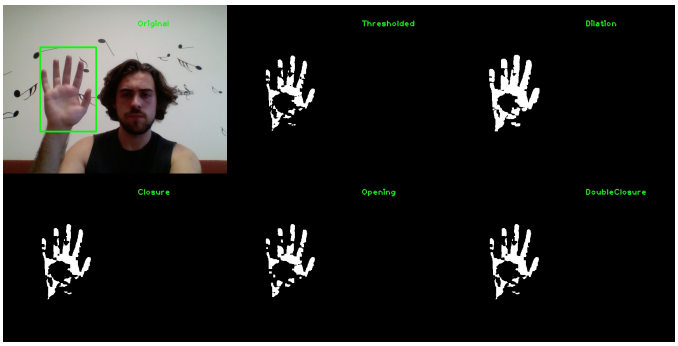
In this section we will show the initial research we made to develop the final app as well as the two modules that compose: hand tracking and gestures recognition. The application was developed using Python3, OpenCv and Numpy.

2.1 Initial research

We started by reading a survey about Hand Gesture recognition methods [2], through which we first understood the better the task and later we were able to evaluate which techniques could be the most suitable for us. We also read other papers more focused on the approaches we preferred. Among these, the most interesting ideas we captured are from "Real-time hand gesture recognition using finger segmentation" [3] and "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware" [4]. From this point we decided to concentrate more on skin-color based detection, for the tracking part and the convex-hull computation for the gesture detection part.

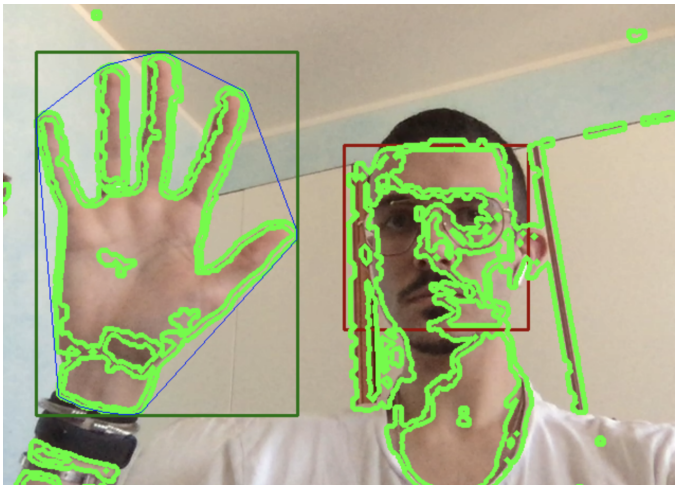
2.2 Development and tests

We decided to begin from the first module, related to tracking, and specifically from the segmentation of the hand, the lowest level task, trying to evaluate all the best possible parameters. For example, we have evaluated any differences between HSV and Y-Cb-Cr (2 of the most used color spaces in image processing application as it separates the colors from their intensity, unlike RGB) and we have chosen the best values for the thresholds, through independently developed applications, of the mask (used to separate the foreground from the background). We then tested several techniques in parallel to eliminate the noise and "compact" the hand blob as much as possible. We tried raw threshold and then dilation, opening, closure and double closure (as in figure).



After doing this, to try to eliminate the problem of the background (mainly the face) that moves, albeit to a lesser extent, we applied a background subtraction algorithm. Finally, for the tracking part, we have tried different method, but the most stable one, without involving deep learning, was the use of "good features to track" as key points. Making the pointer movement stable was equally difficult. We decided to use the average of the difference between the previous points and the new position of the key points, scaled by an acceleration parameter.

For the module concerning gesture recognition we have done similar segmentation processes, but in the pipeline we have also applied the Canny edge detector [5]. This step was crucial in order to then be able to calculate the convex hull of larger area and obtain an adequate tool to be able to analyze the gestures. Finally we applied a convexity defects detection algorithm to count the raised fingers and we use this information to identify different gestures (clickUp, clickDown, etc ...).



2.3 The application

In this section we will talk about some of the implemented elements in details, to better understand how we realised the 2 modules for the final version and how we join them together.

2.3.1 First module

A the end the tracking module is composed by a pipeline, repeated in the main loop, in which features are extracted from the camera and then are used for the tracking. At first the image is processed by a set of transformations, such as change of the colorspace, gaussian background subtraction, median blur and thresholding. After this, from the image Good features to Track [6] key points are extracted and parsed to obtain another set of moving corners and then this corners are saved for the next loop.

In a set of subsequent loops (the number is determined by parameter) the previous corners are tracked using the Lucas-Kanade optical flow [7], with the pyramidal implementation, and both previous and new corners are used for computing the estimated motion of the hand for every tracking point. In the last point the average motion is computed and then this value is scaled by an acceleration factor and used to move the simulated pointer.

2.3.2 Second module

In the second module, on the other hand, we start with already masked frames to which dilation is applied. Subsequently, through Canny the edges are extracted and they are subjected to dilation again to improve the stability of the processing in the subsequent phases. At this point the external contours are extracted and among these the one with the greatest area is chosen. The Convex hull algorithm is then applied to this boundary and the minimum enclosing circle is calculated. As a last step, the convexity defects are extracted between the Convex hull and the wider contour, found earlier, and controls are applied to these in order to be used for gesture recognition. Finally the number of fingers raised, and the change in this number are mapped to a specific action, and that action is written in the simulator window.

2.3.3 Join everything together

To join the two modules together we have opted for a switch approach based on a window of frames. Specifically, we have decided to develop the program in such a way that as long as a movement is detected the tracking module remains active, when the hand is still for a certain number of frames, the gestures recognition module is called up. Moreover, to return to the tracking mode it is request to raise all the 5 fingers at the same time, or alternatively press the key "g".

3 Usage

The program can be launched with 'python track.py' from the correct environment. It will detect hand motion and gestures, then show both of these onto 2 separate windows.

By pressing the 'C' key, one can activate the "threshold-setting mode": it allows to clearly see the color-thresholded image, in order to have a more suitable threshold for the current lighting and color conditions. Since the program's performance highly relies on threshold quality, we recommend to set this up at the first usage.

A custom threshold can be set thanks to the sliders appearing together with the thresholded image, and it can be saved for future use by pressing the 'S' button. Any saved custom threshold will be used by default on next uses.

To deactivate the color-threshold choice mode, press again the 'C' button.

To activate the 'gesture mode', the user should stay still for approximately 1-2 seconds. To deactivate the 'gesture mode', one can either use the 5-fingers (open hand) gesture or press the 'G' key. Recognized gestures are:

- 2 fingers –> Left Click Down;
- 3 fingers –> Right Click Down;
- 4 fingers –> Click Up;
- 5 fingers –> Close Gesture Mode.

Once initialized, this program will open a window with 3 frames:

- the first showing the GoodFeaturesToTrack (moving) points over the masked image;
- the second showing the result of the convex hull to detect gesture, only when gesture mode is active;
- the third showing predicted mouse motion.

To quit the program, press Q.

4 Conclusion and results

In conclusion, in the development of this project we encountered many difficulties, due to the level of processing to which we followed. Despite this, some of these have been overcome by taking into consideration several aspects in sequence, such as the exclusion of the face, which is performed by combining the concept of background subtraction with that of the calculation of motion as a difference with the previous points. As said introductively, the work carried out was not intended to compete with the state of the art in this field, but to develop an application that would allow us to acquire knowledge in this type of task, as well as learn how to implement resolutions already studied by other researchers. The application run using a simulator just for the final stability, which although satisfactory from a didactic point of view cannot, at present, be used permanently as an alternative to the mouse.

References

[1] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," 2019.

[2] M. Oudah, A. A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: A review of techniques," *Journal of Imaging*, vol. 6, p. 73, 07 2020.

[3] K. J. T. L. J. Z. J. . Y. Y. B. Chen, Z. H., "Real-time hand gesture recognition using finger segmentation," *TheScientificWorldJournal*.

[4] L. B.-G. L. H. Yeo, Hui-Shyong, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," *MultimediaToolsandApplications*.

[5] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[6] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[7] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," vol. 81, 04 1981.