### recovering 3d information with a signle camera

Haoran (Simon) Liang

Rose-Hulman Institute of Technology, Imaging Systems Minor Project  
Email: [liangh@rose-hulman.edu](mailto:liangh@rose-hulman.edu)

#### Abstract

Most of the applications require stereo cameras in order to recover 3D information of a scene. However, after knowing information such as the camera’s intrinsic properties and the real-world coordinates of an object, it is possible to recover the 3D information from an image of the given object taken by a single camera. In this paper, a Python-OpenCV application for recovering 3D information with a single camera has been proposed. The application takes input from a single webcam, then tracks a square in the scene. After knowing the corners of the square, the recovery algorithm is applied to recover the 3D information.

**Keywords—** 3D information, mono-camera, Python, OpenCV

**1. Introduction**

The purpose of this study is to explore hand gesture classification and develop an algorithm that will identify these gestures from a live video feed. Algorithms like this can be very valuable in many different applications such as interpreting sign language, human and computer interfacing and robot control. Using a camera to interpret sign language can be useful for teaching people sign language and giving feedback. Another implementation could be for faster easier human to computer interfacing, by leaving hardware like the mouse out of the loop. In robot control it may be easier precisely control a robots motion without the use of joysticks or other control devices.

Previous work in this area includes, hand gesture recognition, the hand as an interface device and 3-D hand tracking. The use of a stereo camera to recognize the hand and estimate the orientation of the hand in 3D space can be found in [3].  [2] Uses stereo camera to track a hand and recreate its 3D pose. [1] Covers blob and ridge detection to find the palms and fingers in a 2D image. Since two out of three designs use stereo cameras, and only one paper actually requires precise segmentation on fingers, the only technique that is brought over to our design is to use distance transform to figure out where the palm is located.

Some of the challenges involved with gesture recognition come with the vast variety of positions and postures the hand can occupy in the image plane. The same hand pose can occupy any coordinates on the screen as well as any gesture can be located on the (x,y) coordinates of the image plane. Another issue is the variety of colors; shapes and sizes of different hands can throw off the robustness of a spatial or color classifier.

Furthermore locating individual fingers, that may be overlapping or in close proximity, can be very difficult to identify even for an edge detector with color classifier. The center point of the palm also shifts with finger movements, making is a challenge to track the noisy input. Another difficulty comes with translating the movement from a low resolution video to a high resolution computer screen. Translating 160 x 120 video feed to a 1920 x 1200 pixel screen using a proportional translation results in large jumps in movement and noise distorting the mouse position.

**2. Process**

The program uses three mains steps for processing the video stream, first is hand segmentation, then gesture recognition, and finally mouse movement.

**2.1. Hand Segmentation**

The camera takes raw images in RGB space. For simplicity, the program converts the raw image to gray-scale. By simply applying a threshold on the gray-scale image, the hand can easily be separated from the black background. Figure 1 shows the raw image and the result image by applying a threshold on the gray-scale image.

|  |  |
| --- | --- |
| a) | b) |

Figure 1. a)The raw image of the hand from the camera b) the resulting image after applying a threshold to the gray-scale image of the hand.

In order to locate the center of the palm, the Euclidean distance transform is applied to the mask, and the coordinates where the maxima are found. The resulting image after the distance transform is shown on Figure 2.



Figure 2. The resulting image after applying the distance transform on the mask. The brightest the pixels are those that are farthest away from the edge of the connected component.

Simultaneously, a corner detection algorithm, found in [4], finds the global and local maxima of the edge curvature on the mask. The corner detector will first apply canny edge detector on the image, then find the curvature along the curve. Figure 3 shows the basic idea of behind the corner detection algorithm.



Figure 3. Examples of detecting the (a) round corners and (b) the obtuse corners using [4]’s corner detection algorithm. (c) and (d) are the corresponding curvature plot of (a) and (b).

Applying the corner detector to the image creates a set of interest points, which includes the fingertips and the area between the fingers. In order to locate the fingertips, all other points must be separated from the set. A convex hull algorithm is used to find the fingertips. This method is effective since the points between the fingers will always be bound by the convex polygon bounded by the fingertips and the wrist endpoints. Figure 4 shows the results of applying the convex hull algorithm to the set of interest points found by the corner detection algorithms.

|  |  |
| --- | --- |
|  |  |

Figure 4. The resulting image of the corner detection algorithm and results after applying convex hull algorithm to the set of points found by the corner detector.

**2.2 Gesture Recognition**

Gestures are determined by the number of fingers fully extended in the image. The number of fingers present in each image correspond to a different commands. Table 1 contains commands represented by different gestures and the sample image of the gesture.

|  |  |  |
| --- | --- | --- |
| Number of Fingers | Corresponding Command | Sample Image |
| 5 | Pause Mouse |  |
| 3 | Move Mouse |  |
| 2 | Left Click |  |
| 1 | Right Click |  |

Table 1. The list of existing commands based on the number of fingers registered, and the sample images of the gestures

**2.3 Mouse Movement**

When three fingers are registered in the system, the mouse will move corresponding to the movement of the center of the palm in the camera. Since the area for the hand to move is limited and the camera resolution is low, a conversion must be made to relate hand movement captured by the camera to the mouse movement on the screen in order to allow ample motion to properly operate the computer. After various experiments using different scaling method, parabolic translation formulas were found to be the most effective means to ensure a proper range of motion on the screen.

Equation 1. The Parabolic Movement Formula

where is the coordinate of the mouse and is the vector of the hand movement from the previous frame to the current frame. This translation allows for accurate movement for the slow motion, and the wide range for the fast motion. Hence, the mouse movement translation guarantees the fine control and the range of the mouse movement.

**3. experiments**

This project required use of a camera mounted facing down, to capture the live video feed of the hand. This camera mount was designed to hold the camera 18” high and it leaves room for hand gestures below. The mount consisted of six laser-cut acrylic parts, each with tabs to connect them. A black backdrop was used to help the algorithm extract the hand mask, by making a consistent easily distinguishable background. The webcam was set to 160 x 120 pixel resolution; the computer screen it was interfaced with had 1920 x 1200 resolution.



Figure 5. Shows the experimental set up with the camera mount and computer.

The algorithm and physical set up was tested by playing a simple computer game, minesweeper, where the goal is to use right and left clicks to “safely” mark out a mine field. This game was successful completed 2 out of 50 games, the left click seemed to accidentally be executed when a right click was desired. Furthermore normal completion time for the user was around 45 seconds, whereas with the algorithm it took 170 seconds.

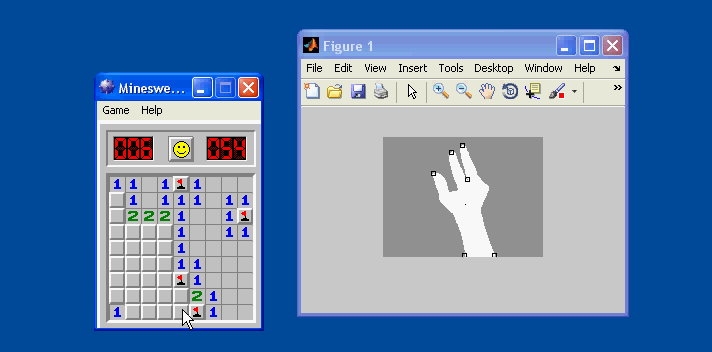


Figure 6. Shows a game of minesweeper being played with the video feedback.

**4. Discussion**

The algorithm is able to consistently locate the center of the palm and count the proper number of fingers present. The center point, as well as fingertips, remains relatively constant with only slight unintentional movements over a small range of pixels. These slight errors are unnoticeable when making large movement. However, when making small movements the error becomes more noticeable and the cursor moves back and forth enough to either void movement or make the slight adjustments overshoot the intended target. For general computer use, where range of motion is expected to vary from across the screen to slight movements, the capability of the algorithm remains a viable alternative for traditional mouse. More precise small movements are still too noisy for practical use because of the shifting of the center point of the palm.

The main limiting factor of the algorithm prototype lies in the hardware used for testing. The camera’s auto-adjustment feature limited the accessible area of movement for the hand to approximately a 16”-by-12” rectangle. Any further motion is likely to cause the camera to begin an auto-focusing process which disrupts the algorithm by causing additional corner points to be detected which the algorithm classified as fingers. The auto-focusing feature also limits the amount the camera height can be adjusted. Raising the camera increases the usable movement area and reduces noise acting on the center point and fingertip locations, but also increases the likelihood of activating the camera’s auto-adjustment.



Figure 6. Corners being located during the camera’s auto-adjust.

Another recurring error is the loss of interest points, especially finger tips and the area between fingers, during fast motions. The loss of these points is due to the image taken by the camera being blurred causing some fingers to blend together or making the fingertips seem flat; either error can prevent them from being identified as a corner. Since this error alters the pictures from what the program is designed to detect, no simple adjustment can be made to compensate for this motion. The only methods that can be used to fix this would be to increase the lighting so the camera can use a quicker shutter speed, which may only allow a slightly faster motion based on the camera’s capabilities, or use a high-speed camera.

|  |  |  |
| --- | --- | --- |
| C:\Users\nelsonlr\CSSE 463\Final Project\Lose Interest Points\start.png | C:\Users\nelsonlr\CSSE 463\Final Project\Lose Interest Points\move.png | C:\Users\nelsonlr\CSSE 463\Final Project\Lose Interest Points\stop.png |

Figure 7. a) shows initial points classified, b)fast motion losing interest points resulting in a left click, c) points being relocated.

**5. Future Work**

Noise in the placement of interest points is expected from virtually all measurements made from a real time image stream. The main noise that affects the operation of the algorihm is noise on the center point which makes fine motion difficult. A method that appears especially promising for preventing this noise is implementing a Kalman filtering. Kalman filtering reduces noise by producing estimates of the true values of two measurements and their associated calculated values by predicting a value, estimating the uncertainty of the predicted value then computing a weighted average of the predicted value and the measured value. The measurement with the lowest measured weight is given to the value with the least uncertainty.

Another improvement would be to expand the number of gestures that can be identified. Currently, the algorithm only has the capacity to track six different states, each defined by the number of fingers, zero to five, present in the image. In order to increase the number gestures that can be identified by gesture mouse, a more complex tracking system is needed. One of the most robust methods for providing additional states to be used as gestures is to apply a system for tracking individual fingers. In order to identify and track fingers, more interest points would need to be located. One of the major issues concerning the implementation of such a system is the finding unique points that are not affected by independent movement of finger joints or the muscles. Hence, a robust hand model would also be useful for further development of the gesture mouse.

An additional improvement could be using a moe effiecient compiler language, such as C with OpenCV to boost the performance of the algorithm. Since C programs are compiled into machine code, and are run directly by the processor, the overall process would be much quicker thn having the MATLAB kernel run in the background to interpret each line of code in the process.

**6. References**

[1]Bretzner, L., I. Laptev, T. Lindeberg, S. Lenman, and Y. Sundblad. “A Prototype System for Computer Vision Based Human Computer Interaction.” 2001. January 20, 2011, <<http://www.nada.kth.se/cvap/abstracts/cvap251.html>>

[2]Rehg, James M. and Takeo Kanade. “Visual Tracking of High DOF Articulated Structures: An Application to Human Hand Tracking.” Proceedings of the 3rd European Conference on Computer Vision (ECCV '94), May, 1994. pp. 35-46.

[3]Sepehri, Afshin, Yaser Yacoob, and Larry S. Davis. “Employing the Hand as an Interface Device.” Journal of Multimedia, Vol. 1, No. 7, Nov./Dec. 2006. <http://www.academypublisher.com/jmm/vol01/no07/jmm01071829.pdf>

[4]Xiaochen He, and Nelson H. C. Yung. "A Corner Detector Based on Global and Local Curvature Properties." (2009). *MathWorks - MATLAB and Simulink for Technical Computing*. Web. 4 Feb. 2011. http://www.mathworks.com/matlabcentral/fileexchange/7652.