Detection Methods for Gesture Recognition

Alex DiStasi
Ithaca College
953 Danby Rd
Ithaca, NY 14850
adistas1@ithaca.edu

Beth Dellea
Ithaca College
953 Danby Rd
Ithaca, NY 14850
edellea1@ithaca.edu

ABSTRACT

This piece documents our endeavor to use gesture recognition through color detection to create a magical experience. Color detection used the normalized look up table method and gesture recognition utilized KNN matching. Ultimately, the program shifted into an interactive art experience.

Categories and Subject Descriptors

I.4 [**Image Processing and Computer Vision**]: [Color detection, color tracking, gesture tracking]

General Terms

Normalized Lookup Tables, Color Tracking, KNN Character Recognition

Keywords

Computer Vision, Color Detection, Gesture Detection, Normalized Lookup Table, KNN

1. INTRODUCTION

Gesture Recognition has many uses in computer vision applications. Gestures are defined as dynamic movements that often carry significance in humans [2]. Recognizing specific dynamic human gestures could be used for conveying information and device control [1]. There are several methods for going about gesture recognition including vision-based, glove-based, and tracking-based methods[2]. In this paper, we will describe how we tried to implement our own vision-based gesture recognizing program using color detection, color tracking, and character recognition.

2. RELATED WORKS

Laviola's research dives into gesture recognition using glove-based and vision-based recognition. Hand position and hand posture can be used to recognize hand movement and gesture. Gloves can be useful in tracking hand posture and position in a 3D space. Virth and Bishop discuss 2D collision detection. Collision detection in real time can be successful if a program can keep track of the object's position. Their methods show that one of the simplest collision detections methods involves checking for a collision through a list of objects every time the main object moves. Vezhnevets, Sazonov, and Andreeva provide comparison and analysis on several skin color detection methods which included Normalized Lookup Tables (LUT). LUT is a form of non parametric skin distribution modeling. This method detects a color by first training the program to understand the distribution of colored pixels. The program can then take a pixel and then determine the probability of that pixel being a certain color. Pérez, Hue, Vermaak, and Gangnets' research focused on probability based color tracking [3]. One of the benefits of color tracking in real time is that a color can be tracked despite large motions, motion blur, and partial occlusion [2].

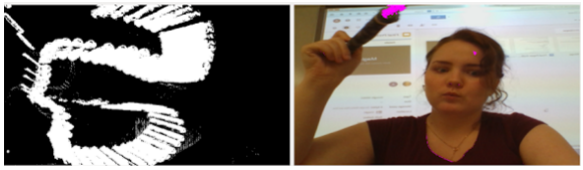

Figure 1: The path of the green wand is shown in white.

3. COLOR RECOGNITION

We used a bright green colored wand for tracking tracking gestures. In order to get the program to recognize the neon green color, we used a Normalized Lookup Table. The distribution of pixels is then used to detect specified colors in an image. The first step is separating the color values into bins. Each bin relates to a specific range of color values[6]. Next, each pixel of the image is analyzed and distributed into the appropriate bin. The bins track the number of times times a pixel falls within that bin range. The count is used to create a probability distribution that determines the likeliness of a pixel being classified as a color. We used a live feed and selected the area of the space that contained the green color we were tracking. The selected region was then sent to a histogram and the values were stored in an XML file. We designed our program to read this file and automatically recognize the bright green color when the program is being run.

Once we could locate the bright green color, we focused on saving the previous positionings of the marker's location

Figure 2: Figures created to train the program and compare the user-created paths to.

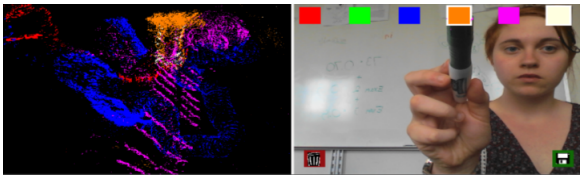

Figure 3: The path color changes when the green marker collides with the boxes at the top of the screen.

in order to track it. Using individual frames from the live input, movement and gesture can be analyzed[1]. To do this, we first set a variable that would keep track of the current time. Since our program uses a live feed, we compare pixel differences between the feeds at different time increments. The program kept track of how many pixels were changed between the two images; if there were more than thirty pixels changed, a file named after the current time is created. The program also creates a new mat that saves a PNG image of the path at the time of saving. Figure 3 shows the path of the bright green marker since the program first begins running.

4. CHARACTER RECOGNITION

Our next step was to take the user-created path and get the program to recognize it as a gesture or shape. An initial version of the tracking program would save the user-created path every time the detected color was not in the camera feed. Using this feature, we created a variety of training paths, following the basic shapes of circles, triangles, and 'N' and 'S' shapes. Gaussian blur and erode and dilate filters were applied to the paths before they were saved to remove as much non-path noise as possible. These saved images were examined to find the most viable training images, which were combined into one file to train for recognition.

The recognition process followed a basic KNN training and recognition algorithm, similar to that used by Rasamimanana, Fléty, and Bevilacqua [4]. Each shape was given a corresponding letter, with circles getting "O," triangles getting "T," 'N' shapes getting "N," and 'S' shapes getting "S." Multiple examples of each shape were provided in an attempt to prepare for user-created shapes, as seen in Figure 2 With a wider variety of each shape, we assumed the program had a better chance of recognizing shapes that did not perfectly match the examples.

The main program was adjusted to attempt KNN recognition on the path every time the wand went out of frame, with the intent of adding a specific effect to the feed based on which gesture was made. Unfortunately, we were not able to draw precise enough shapes for the learning to be successful; though it regularly recognized shapes in the paths drawn, the results were infrequently accurate.

5. COLLISION DETECTION

Once we concluded that we could not get the program to recognize the user-created paths as shapes, we decided to take the project in a different direction and turn it into an art program. We continued to use the LUT to track the green color and turned the wand into a drawing tool. The programs ability to save and display the path of the green color on a black background could be used to show the user's

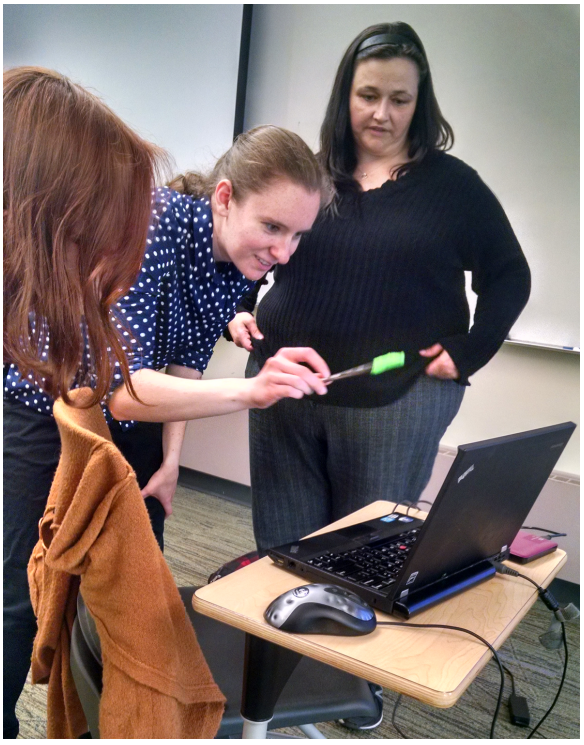

Figure 4: A user test of our art program performed by Caitlin Wormsley and Denise Fullerton. Caitlin is using one of the smaller brush options.

drawing. The user-created path is shown in a white color. To transform our current project into an art program, we wanted the user to be able to choose the color of path. We created different colored boxes at the top of the live feed, as seen in Figure 3.

The program can recognize if the green wand is colliding with one of the boxes by using a collision detection method that checks if the position of the green wand is within the space the square occupies [5]. If there is a collision, the path of the marker would then be the same color as the box. This allowed the user to choose the color of the path and use multiple colors in one drawing session.

6. CONCLUSIONS

This program has evolved beyond its original magical concept into an application which is more widely usable. The gesture recognition program required high levels of precision in order to ensure accurate recognition and results, while most slip ups in the art program will only serve to take

the creative piece into a new direction. The project demonstrates the use of color recognition through look up tables, color tracking, and collision detection. Each of these processes enhances the project and evolves it, from merely a recognition tool to a basic drawing tool, and finally to an art experience.

As seen in Figure 4, our final program is engaging not just for those actively creating art, but also for spectators. Though the initial goal is not the final result, what we have created is an interactive art experience which elevates the typical paint application to a more realistic and interactive level. We have found that the experience of the program is reminiscent of that of actually painting. Moving a "brush" and creating art is an experience we have digitized, made more portable, and more affordable.

7. IMPROVEMENTS

There are several improvements we would make with our project. One current issue is the color detection method. Although the method is mostly accurate, there is still the case that non green pixels get passed as green pixels. This creates noise that interferes with our collision detection. The noise pixels would set off the collision detection flags causing the program to change the color of the brush or clear the screen. We noticed that the lighting in the room would greatly affect results. The kind of lighting impacted the color detection very strongly. Fluorescent lights, natural sunlight, and more yellow lights all changed the color the wand read as, causing it to fall in and out of the histogram's range.

Another improvement we would like to make for our project is getting the KNN recognition to work. We believe that using a brush with a finer point would allow the user to create more precise shapes, which could improve the accuracy of the recognition. We could also move further in the directions researchers attempting to widely recognize handwriting go.

One issue we had with the KNN recognition was finding an appropriate point to check the user-drawn path against the pre-made paths. Our original method compared the two paths only if the green wand was moved off the screen. Once the program was not able to detect the green color, a check would be done. The issue with this method is that moving the wand off screen still leaves a small streak in the history of the path which throws results off. The way the we handled the issue during the project was to cover the camera after completing the gesture, while holding the wand steady. This process often involved the assistance of another person to cover the camera effectively. A future goal is to improve this method or come up with a new one, such as finding an appropriate interval to routinely check for gesture matches to maximize accuracy and minimize wasteful processing. As a future goal, we would also like to incorporate audio recognition for further user control of the program. Another goal would be to add visual effects after a successful gesture.

8. REFERENCES

[1] C. Harshith, K. Shastry, M. Ravindran, M. Srikanth, and N. Lakshmikhanth. Survey on various gesture recognition techniques for interfacing machines based on ambient intelligence. *International Journal of Computer Science & Engineering Survey IJCSES*, 1(2):31âĂŞ42, 2010.

[2] J. J. LaViola, Jr. A survey of hand posture and gesture recognition techniques and technology. Technical report, Providence, RI, USA, 1999.

[3] C. V. J. G. M. Pérez, P.and Hue. *Computer Vision — ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I*, chapter Color-Based Probabilistic Tracking, pages 661–675. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[4] N. H. Rasamimanana, E. Fléty, and F. Bevilacqua. *Gesture in Human-Computer Interaction and Simulation: 6th International Gesture Workshop, GW 2005, Berder Island, France, May 18-20, 2005, Revised Selected Papers*, chapter Gesture Analysis of Violin Bow Strokes, pages 145–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[5] J. M. V. Verth and L. M. Bishop. *Essential Mathematics for Games and Interactive Applications, Second Edition: A Programmer's Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition, 2008.

[6] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *IN PROC. GRAPHICON-2003*, pages 85–92, 2003.