CSCI 5561 Project: Gesture Based Control for Commodity Hardware (Gesture ReSIFT)

Grant Matthews
University of Minnesota
matth536@umn.edu

Alex Lorimer
University of Minnesota
lorim007@umn.edu

Abstract

This paper describes the creation of an algorithm to recognize gestures on commodity consumer hardware. It will detail past work, baseline and proposed methods as well as analyze the quantitative and qualitative results produced by the algorithm.

1. Introduction

The most common and largest standard size computer keyboard consists of 104 keys. However, there are 149,186 possible unicode characters and 1,000s of potential shortcuts within a given host operating system. Currently, the best way to access more than just 104 actions is through the use of context menus and modifier keys, both of which require the memorization of (potentially) complex strings of actions and multiple keypresses and mouse movements. This project's goal is to take a computer's webcam, a typically underutilized input device, and use it to add more than 104 one key (read motion) actions to a user's computing experience.

With the rise of the COVID-19 pandemic, industries and educators around the world moved their operations online. As a result, the demand for video conferencing hardware and software skyrocketed. Even now in 2022 as the world returns to face to face interaction, millions of people still have access and regularly use a webcam for virtual interaction. This project aims to take advantage of this opportunity by building a simple, lightweight gesture control system that utilizes a webcam to add additional controls to a user's computer.

The challenges that are proposed in this project are to make the algorithm lightweight enough to essentially act as a background process that will not slow down the user's computer, make the algorithm accurate enough to establish user confidence in minimizing false positives which can disorient and annoy the user. This report will analyze past work, explain the proposed method for creating a gesture control system, and examine the final implementation to determine its success and viability for the average computer user.

2. Methods and Past Work

2.1. Related and Past Work

There have been many different implementations of gesture recognition in the past. As a way to better understand and accomplish the project's goals, the following were reviewed and summarized below.

One of the earliest forms of gesture recognition predated the introduction of a computer webcam by about 10 years. Gary Grimes created the Digital Data Entry Glove in 1983 which consisted of a glove with sensors to detect if they were touching something. This was able to roughly approximate the 3D position of the wearer's hand in a digital form [4].

Conceptualized in 1980 and realized in 1994, the MIT-LED glove was a series of LEDs embedded into a glove at key hand locations such as joints and fingertips. This glove was then worn in front of a camera capable of tracking and representing the LED dots within a digital space [8].

Early camera based gesture recognition problems were solved using multiple cameras to help with occlusion and to recognize when two fingers might be behind each other. One of the first successful implementations of this technology called DigitEyes was capable of basic gesture recognition on a simple background through the use of 2 cameras [5].

In 1998, four researchers paved the way for modern gesture recognition by developing a method to track hand gestures without the use of multiple webcams or basic backgrounds. They were able to track key points on the hand and develop a list of possible positions for each digit just as before. However, they then compared each possible position against a list of realistic hand positions to determine the most likely position the hand was in. This demonstrated the possibility of single camera gesture recognition without the need for complex tracking [6].

A common form of pre-processing is skin segmentation. This is the process of isolating the colors commonly found in human skin and disregarding colors not a part of the human skin tone to focus on the hand making the gesture. While relatively easy to detect using an RGB camera (like a computer webcam) in a controlled environment, it is far more difficult to detect human skin tones when a complex background with colors similar to skin tones are present [7]. This is because not only are there other textures with a similar color pallet such as wood or leather, but also because of variations in lighting that can cause the skin to appear a different shade.

This has been solved through the use of cameras that capture different light spectrums such YUV or HS(I, V or L). Human skin tones contrast better with surrounding items when viewed through cameras that capture in one of these spectrums, thus making it easier to recognize a hand performing a gesture [1] [2].

Feature extraction is commonly done using SIFT features which is described in greater detail below. Furthermore, classification can be done numerous ways. Because this project isn't trying to classify an unknown gesture, but rather trying to match it with an existing gesture, the classificationon step is more akin to a feature match which is described in the proposed method below.

2.2. Baseline Method: SIFT

The project primarily makes use of the Scale Invariant Feature Transform algorithm (SIFT) which can be broken down into two sub algorithms that together create a list of keypoints. These key points have a magnitude and location, and can be compared to other key points in other images to find similar features between the two images.

The first component of the SIFT algorithm is called scale-space peak selection. Here the algorithm creates multiple octaves of an image, where each octave contains images that are half the size of the previous octave's images. Each octave is then progressively blurred using the Gaussian method to create a 3D stack of same size images. This is done to replicate a human's ability to focus at scale. As Deepanshu Tyagi [9] states, "you might see a sugar cube perfectly on a table, but if looking at the entire milky way, then it simply does not exist." BY shrinking the size of the image, each pixel contains more and more information about the original image, and thus the algorithm is looking at more and more information with each octave.

Finally the pixel is compared to the 28 other pixels around it in the 3D stack, and if it is a local maximum then it (or it's relative position in the original image) is added to a list of potential key points. This list is refined during the second component of the SIFT algorithm, key point localization.

Key point localization is used to weed out pixels that do not lie on an edge and thus are not as important when describing the image. Here, the algorithm takes in the list of potential key points as generated above and runs each pixel through an edge detector, discarding the pixels that are not along an edge. The final list of pixels is then considered the key points of the image [3].

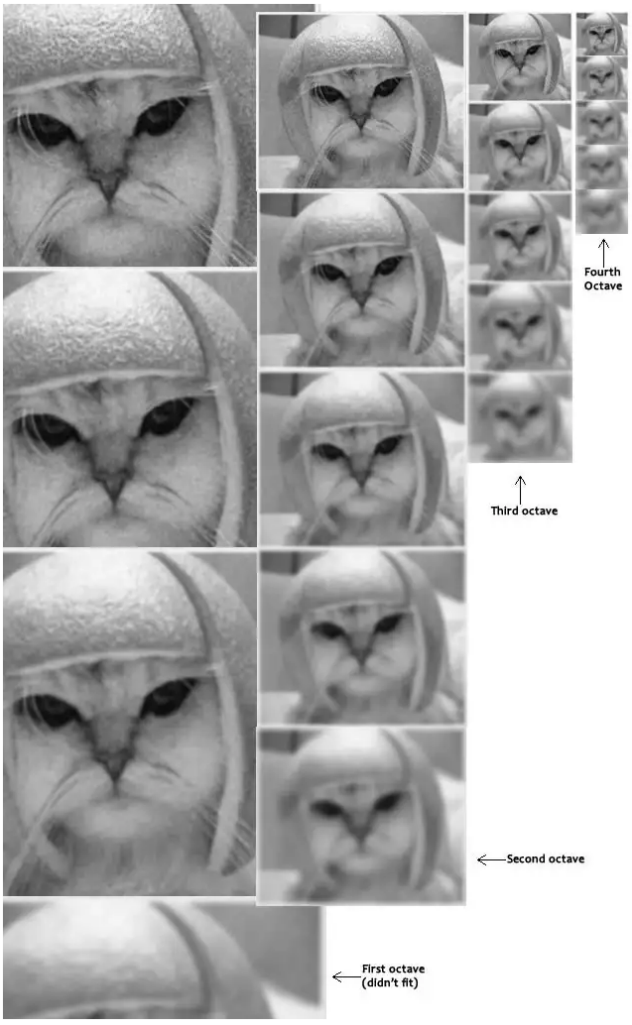

Figure 1. Example of Octaves used in the SIFT Algorithm (from [9])

2.3. Proposed Method

One of the project's most important goals is creating an algorithm that is capable of running on commodity or everyday consumer hardware in real time. Therefore this project's method needs to be a relatively simple algorithm that is accurate without requiring GPU acceleration or multi computer computation.

Our baseline method and approach uses the Scale Invariant Feature Transformation (SIFT) feature matching algo-

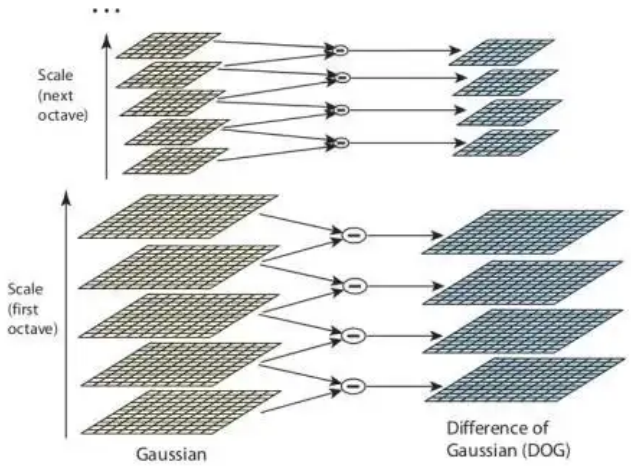

Figure 2. Octave Stacking for calculating Local Maxima (from [9]

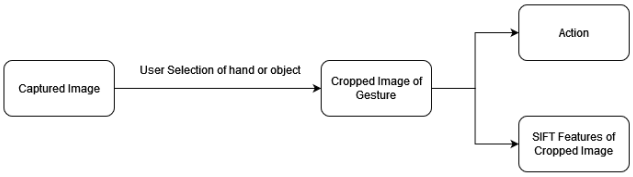

Figure 3. Flow Diagram for adding a gesture

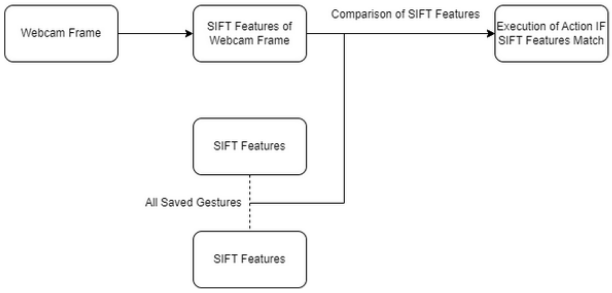

Figure 4. Flow Diagram for checking gesture matching

rithm to figure out what gesture our user is making. There are a few reasons why SIFT was selected over other potential algorithms. The most important of those reasons is the speed at which it is possible to extract SIFT feature points from an image. As described in the results section, extraction of SIFT features from even a relatively large image can be done very quickly. This allows the algorithm to process each frame produced by the user's webcam against all the saved gestures multiple times per second, enabling real time registration and execution of a gesture's saved action.

While using SIFT features allows for a quick algorithm, use also allows for flexibility in the size of the images being compared. Each gesture is a user cropped sub frame of a larger webcam produced frame but each time the algorithm checks for a gesture it considers the whole webcam frame. Many traditional and "fast" image comparison algorithms require identical image dimensions between the two images. Use of one of these algorithms would require splitting each webcam frame into sub images and then checking each of those images against each of the gesture images. With the goal of processing speed in mind, it was decided that an algorithm that isn't reliant on matching dimensions would work best.

The method used by the project algorithm can be split into two parts, the creation and storage of a gesture and the continuous checking and execution of a gesture. When creating a gesture, the user has two seconds to position their hand or other object in the webcam frame. The user's webcam then captures a reference frame, and asks the user to draw a bounding box around the hand or object being used as the gesture. The algorithm then uses OpenCV to extract a list of SIFT features from the cropped image and then saves those features along with a user defined action to a list of saved gestures. This process can be seen in more detail in Figure 3.

The continuous recognition algorithm to detect a gesture and execute its corresponding action analyzes each frame (or a parameter defined number of frames per second) to determine if a gesture is being performed by the user. Each time a frame is analyzed, the algorithm first uses the OpenCV library to generate a list of SIFT features from the frame. The algorithm then iterates over the SIFT features from each of the gesture images and selects the image with the greatest number of matching features. If the confidence of the algorithm is above the default or defined user threshold, the algorithm executes the gesture's action.

Our proposed method also wanted to increase our confidence when we think a gesture is being made over the conventional baseline SIFT feature matching method. The proposed method does a brute force SIFT 2 feature K-nearest neighbor feature match between our ground truth gesture image and the input from our camera. We then filter out the list of key points K-nearest neighbors for any points that are "too far away" from each other for each feature match, meaning that if the two nearest neighbors differ too much in terms of their feature descriptors, then they were unlikely to have been a match. After this, the average coordinates of the filtered matched features are grabbed, then we we check to see how many filtered features are left, if we have over a threshold then grab a padded image around the central coordinate from the camera input image, crop it, and then repeat the process of a 2 feature K-Nearest neighbor feature match between our cropped image and our ground truth image, though this time through we have a relaxed ratio multiplier between the distances because we are more confident that we are looking at the part of the input image

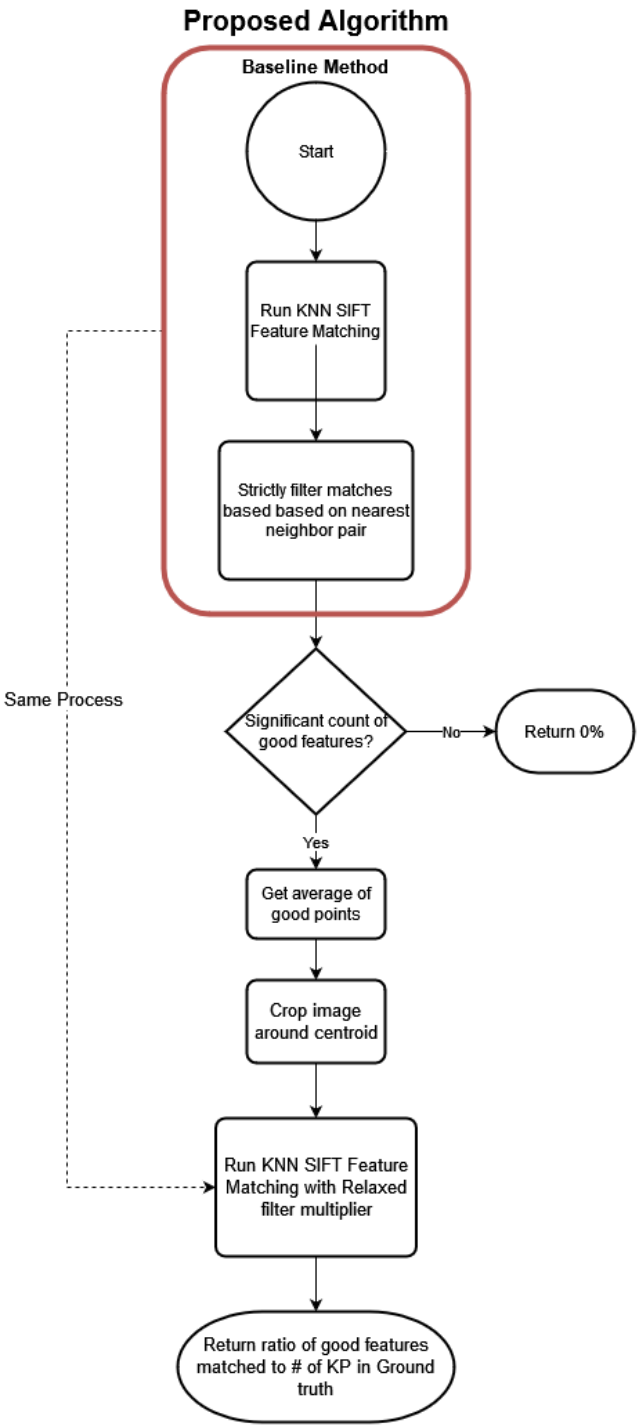

Figure 5. Proposed Algorithm Flowchart

that would contain the gesture.

A design choice that was mentioned was determining whether or not to proceed to the second layer of the algorithm which is the cropping of the input image and running SIFT feature matching on the cropped sub image. The reason that this was done is because while we were doing our initial investigation and observing the baseline SIFT feature matching method, we saw that in a complex environment like a room full of different lighting features and objects in the background would create false positives in the initial pass with a small matching list size. Adding this base requirement helps reduce unnecessary work on false positives and ensure a greater confidence score when running our proposed method.

The reason we crop the image is when we do the initial feature match on the image that comes from the camera, the section that we want to look at or may contain our feature may not contain a set of features large enough when compared to the ground truth image to warrant significant enough evidence that the two gestures are the same. Another reason is that this increase in accuracy comes with minor to negligible increases in run time or work required by the algorithm to process.

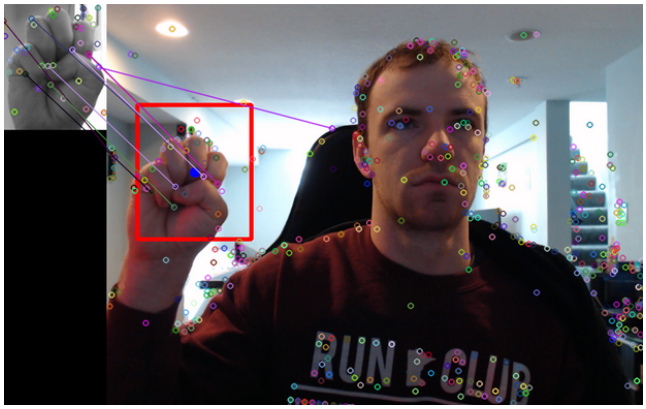

Figure 6. Skeleton algorithm application showing SIFT feature matching for ground truth image and camera input

3. Results

3.1. Quantitative

For our testing, we used a skeleton application that utilizes both the baseline and proposed method to compare confidence scores between the two algorithms based on a single baseline image and the camera input.

From 7, we can see how the confidence increases with our cropping and reapplication of SIFT feature matching on top of a strict SIFT K-NN feature match test. The scores for the figure above are direct comparisons between the values within the same column. The red line near the bottom of the table indicates the cutoff of if we are going to run our proposed method on a cropped image. When the Single SIFT (blue) dots yield a translated confidence [1] below this red line, we are not confident that the initial pass of sift

[1]Because the number of good and filtered key points determines pro-

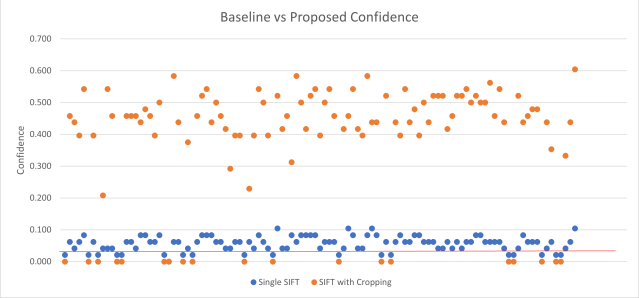

Figure 7. Comparison between baseline SIFT feature matching and proposed cropped SIFT feature matching confidence levels

feature matching yielded enough points near our desired input gesture to do the crop and retest of SIFT on the cropped image, therefor we just return a confidence score of zero.

The second goal that our proposed method needed to achieve was that it does not slow down the machine that it is using. From looking at the delay time between each frame, for the two algorithms, we are not seeing a significant difference in time between frames for each method. Though this may not make sense as since we are running SIFT a second time, since the data set that we are running SIFT on the second time is a smaller cropped image, its run time is dramatically lower than SIFT on the initial full resolution image from the camera. This is also exaggerated by the fact that the cropped SIFT algorithm does not fire every check/frame, thus bringing down the average run time of the added work.

3.2. Qualitative

There are two main metrics within the application of our algorithm within our application program that contribute to the feel and trust on the impact of the algorithm on the application.

The first is reducing the number of false positive action activations. Looking back at figure 7 we can see that the proposed algorithm contributes to an increase in confidence over the baseline which allows us to increase the required confidence for an action to fire. While also testing the baseline algorithm, we encountered a number of situations where less complex [less features] gestures would cause an increase in the activation of an action while no action was present at the screen, or the area section it determined was a gesture was seemingly random in the background.

The second contributing factor is the responsiveness of the application to fire an action after a gesture is initially made. From interacting with the application, both methods had a quick response time after a gesture was made within a negligible human perception time frame. This can be attributed to the lightweight nature of both algorithms compounded with a camera that can video pictures up through 30 frames per second. Even if the algorithm does not get a strong enough confidence score on the first or second frame, the time to try for the next frame can be around 0.061ms from our time testing, which is a negligible amount of time in comparison.

ceeding with the proposed algorithm, this number directly translates to the confidence score returned in the test

4. Conclusion

In this project, we observed the shortcomings of using SIFT feature matching as a gesture recognition algorithm and improved on this by adding another layer of SIFT feature matching using a cropped feed from the initial SIFT pass which helped us find our potential gesture to crop around. Our proposed method showed an increase in confidence score over the baseline method while also adding negligible work on top for the computer to process, thus keeping the responsiveness of the application at a higher standard.

Given enough time, we can expand on the usability and control that the user has over the software if we are able to implement motion tracking of a gesture. Simple motion activated gestures like waving a had to move forward or backwards between applications, or adding a feature to load an save gestures/commands from a directory or file would be very usefulto the repeated usability of the tool created here.

References

[1] L. Torres A. Albiol and E. Delp. Optimum color spaces for skin detection. in: Proceedings of the international conference on image processing, 2001. 2

[2] M.A. Mottaleb A. Senior, R.L. Hsu and A.K. Jain. Face detection in color images, 2002. 2

[3] Crystal M. Dunxu H Ahmed, E. Skin detection-a short tutorial. encyclopedia of biometrics by springer-verlag berlin, heidelberg, 2009. 2

[4] J. Davies and M. Shah. Recognizing hand gestures. eccv-94, 1994. 1

[5] Xu Guangyou Huang Yu and Zhu Yuanxin. Extraction of spatial-temporal features for vision-based gesture recognition, 2000. 1

[6] Y. Kuno N. Shimada, Y. Shirai and J. Miura. Hand gesture estimation and model refinement using monocular camera-ambiguity limitation by inequality constraints. proceedings of third ieee international conference on automatic face and gesture recognition., 1998. 1

[7] P. Premaratne Q. Alshebani and P. Vial. An embedded door access based on face recognition system: A survey. to appear in (icspcs), 2013. 2

[8] J. Rehg and T. Kanade. Digiteyes: Vision-based human hand tracking. proceedings of european conference on computer vision, 1994. 1

[9] Deepsanshu Tyagi. Introduction to sift(scale invariant feature transform), 2019. 2, 3