Enabling Cursor Control Using on Pinch Gesture Recognition

Benjamin Baldus

Debra Lauterbach

Juan Lizarraga

October 5, 2007

Abstract

In this project we expect to develop a machine-user interface based on the pinch gesture recognition using simple computer vision techniques. Our goal is to make a simple interface that can be run using ordinary hardware and requires little training.

1. Motivation

The use of only basic input devices such as the mouse and keyboard is limiting the creativity and capabilities of the user. We want to expand the ways that people are able to interact with their computers. We want to enable all users to interact more naturally using simple hand gestures to control tasks. These hand gestures can be recognized in real time to allow for quick human-computer interaction.

Bringing the thumb and forefinger together is a natural gesture that can be used for cursor control via a computer vision-based interface. When performing the gesture, the hole formed in the middle of the hand shape divides the background into two different connected components, the large background and the hole itself. Using image segmentation and connected components analysis it is possible to detect this hole, avoiding having to use complicated algorithms and hand shape recognition techniques.

Once the hole is recognized, several procedures can be performed to measure its movement and orientation. All these measurements can be translated into mouse events, letting the user control the cursor.

1.1. Target audience & Need for the application

Anyone with a computer and a camera would be should be able to take advantage of this method. This type of human and computer interaction would also benefit those that have difficulty using the normal input devices. For example people with hand and wrist problems such as carpal tunnel or arthritis have extreme difficulty using a mouse.

For most users they will find that this is a more natural way to interact with their computers. With the camera positioned over the keyboard this also speeds up the transition from typing to interacting with the computer in other ways.

1.2. Previous approaches

Hand gesture recognition for computer control is a popular research topic in computer vision. The first research on in the area began in about 1992, when it first became possible to grab images from a camera in real time, and thus enable effective human-computer interaction. Research in the area has typically fallen within the categories of applications for pointing, presenting, digital desktops, virtual workbenches, and virtual reality. In all cases, the motivation has been to create convenient, intuitive, powerful means of interaction with a computer that can serve as an alternative to the traditional keyboard and mouse.

Approaches to recognizing hand gestures have usually been divided into two types: *model-based* and *view-based* approaches. Model-based approaches use a 3D hand model for tracking, which has many more potential applications, but is a challenging task. The main problems are that it works well only for relatively slow hand motions, and in constrained environments (Stenger, 2006). View-based approaches, however, use pattern classification to identify hand features and determine the current hand pose. The main challenges in this case are to find how to segment the hand from the background, and how to determine which features to extract from the segmented region. Segmentation is often done using skin color detection and segmenting this from a uniform background.

For both approaches, tactics to help in recognizing hand gesture input include:

- using props or input devices, such as a pen or dataglove
- restricting the object information, as through a silhouette of the hand
- restricting the recognition situation, by using a uniform background
- restricting the set of gestures made, and using only one hand (Lenman, Bretzner & Thuresson, 2002)

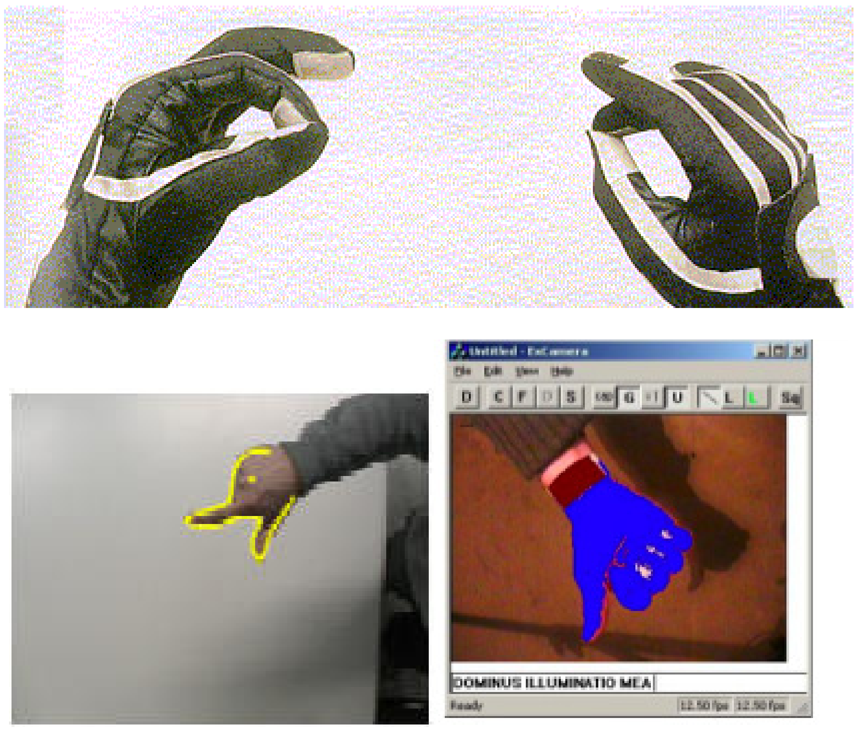


Figure 1: Some previous research approaches, using data gloves, edge detection, and skin color segmentation.

Cursor Control

Several applications for cursor control have been created using 2D tracking. These methods all use a single camera to track a hand, and have typically used pointing gestures as the means of replacing the mouse. An early example is *Finger Mouse* (Quek, Mysliwjec & Zhao, 1995), which used a camera to view the hand from above and recognize hand gestures to control the mouse. One drawback of this system was that the user had to press the shift key with their other hand in order to ‘click’, making the system not completely gesture-based. Later applications found ways to solve this problem and create click gestures, such as O’Hagan (1997) who defined clicking as when two fingers were extended, and von Hardenberg and Berard (2001) who defined clicking as when a one second delay in hand movement occurred.

1.3. Related work

While previous applications have used complicated model or view-based approaches to do hand gesture tracking, the goal of our project is to implement this in a simpler, more intuitive way. We also wish to study strictly hand gestures that are done over the keyboard instead of away from the computer, since this is a more natural and easy place to use gestures. To fulfill these goals, we will be basing our project off of the research of Andy Wilson of Microsoft Research, who has created a way to detect pinching gestures over the keyboard using simple image processing techniques.

Wilson’s technique makes use of the fact that the keyboard (in his case, a common black keyboard model) is much darker than human skin tone. The camera, set on top of the computer monitor looking down, records a picture of the keyboard when the application is first started. When the hands appear over the keyboard, it then uses this image to be able to distinguish when the user, pinching their thumb and forefinger together, has created a ‘hole’ over the keyboard. This hole is distinguished in several steps, including obtaining a binary segmentation of the scene, computing the connected components of the background pixels, and identifying components of significant size, which are the ‘holes’ of a pinching hand.

The centroid of this hole is the point used for cursor control. Cursor movement is enabled when the user first pinches their fingers, and ends when the pinching stops. To create mouse clicks, the interface uses the rapid closing and opening of the thumb and forefinger as the mouse-down and mouse-up events. Double clicking can also be done this way, by creating the motion twice. For dragging, a quick open-and-close gesture of the thumb and forefinger is used to initiate dragging, and is ended when the fingers are separated.

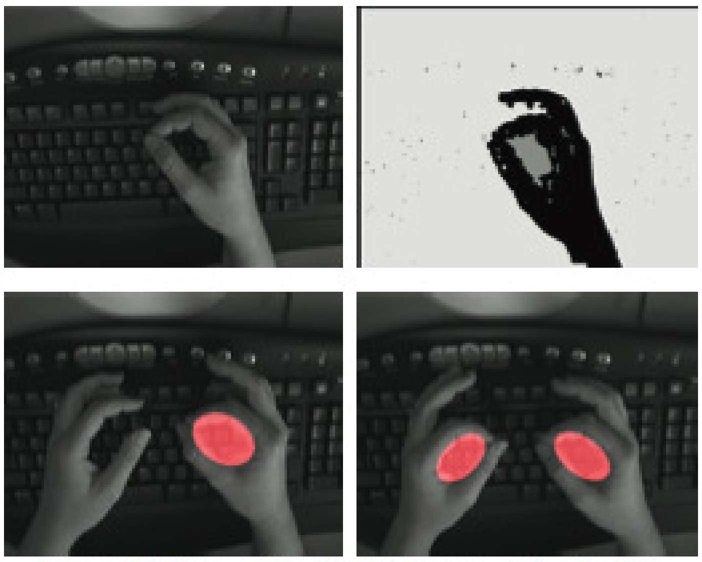


Figure 2: Upper left: Typical image of hand with thumb and forefinger together, acquired from camera above the keyboard. Upper right: Image segmentation indicates background pixels. The hole formed by the thumb and forefinger is detected as a distinct connected component (shown here darker in color). Bottom left: Ellipsoidal model of hole component for the closed hand, shown in red. Bottom right: two hands are detected.

Wilson’s technique also allows for more complicated interaction, such as translation, rotation, and scaling. These changes can be computed from the ellipsoidal hole created when pinching. Scaling, for example, is done when the size of the ellipsoid gets larger or smaller as the hand moves either toward or away from the camera.

This technique can also be implemented using two hands to create pinching gestures. This is especially useful in enabling special interactions, such as zooming in or out by moving the hands together or apart. Moving one hand faster than the other will rotate the view about an axis determined by their motion, which each hand being ‘pinned’ the map during the rotation.

This method has several advantages over other more complex hand tracking methods. For one, the pinch gesture is more precise than tracking an extended index finger, since it is less ambiguous as to when the gesture has been made. The simple image processing performed for this method is more straightforward than other techniques, in that it does not need to know anything about the precise hand shape or where the fingertips are.

1.4. Previous experience

Ben Baldus is a senior in Computer Science. Over the past four years he has gained experience programming in Visual Basic, C, and C#. Ben has been programming in C++ for over 5 years. Though he has been involved in several unique projects and created several windows applications in his free time, he is really excited to bring his skills to this interesting and novel project.

Juan Lizarraga has successfully completed the first 4 out of 6 years of Telecommunication Engineering at the *Universidad Pública de Navarra* (Spain) and he is now in his 5th year at Iowa State University. During this time he has attended courses in several fields such as Circuit Design (analog and digital), Microwave Devices, Fiber Optics, Computer Networks and Signal Processing. The latter has made him gain experience in MATLAB programming through several projects on audio signal processing, digital filters and FFT algorithms. He also has some experience in C, Pascal and Java programming.

Debra Lauterbach is a senior Computer Science and Psychology. She has experience programming in C, C++, and Java, as well as knowledge of web programming languages. She has been involved with the Human-Computer Interaction program since her sophomore year, doing research related to eye tracking and later with OpenGL graphics programming for the C6, and taking classes and seminars in HCI. She plans to go on to graduate school in HCI to study user interface design.

2. APPROACH

2.1. Equipment

- **Webcam**

It will be necessary to use a standard USB webcam with a minimum resolution of 640x480 pixels at a frame rate of 30Hz.

This device will be installed on the top part of the monitor pointing toward the keyboard. The relative position of the camera and keyboard must be as perpendicular as possible, in order to reduce the tilt angle of the camera.

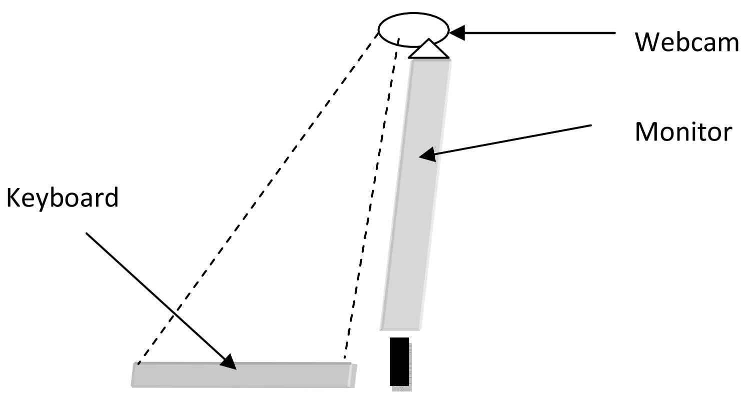


Figure 3: Example Setup

The scene captured must fit the keyboard.

- **Software**

  - Windows XP: this operating system developed by Microsoft is a stable and trustworthy platform for running applications made with Microsoft Visual Studio.

  - Microsoft Visual Studio 2005 is Microsoft's software development product for programmers. It includes several programming languages such as Visual Basic, C++, C# and J#. In our case we will develop our project in C/C++.

  - OpenCV is a C++ based computer vision library developed by Intel. It focuses mainly on *real-time* image processing, and gives the user a large amount of tools that provide extensive functionality on this field. In our project we will use OpenCV for capturing the video stream.

- **Ordinary PC**

  As with any other program this one will need a platform on which to run. Any standard PC running Windows XP should be able to run our project. The minimum hardware requirements are those of the operating system and the required software. There is just one hardware requirement that does not depend on the software, but is necessary because it makes the algorithm more accurate. It is necessary to have a black, or at least dark-colored, keyboard.

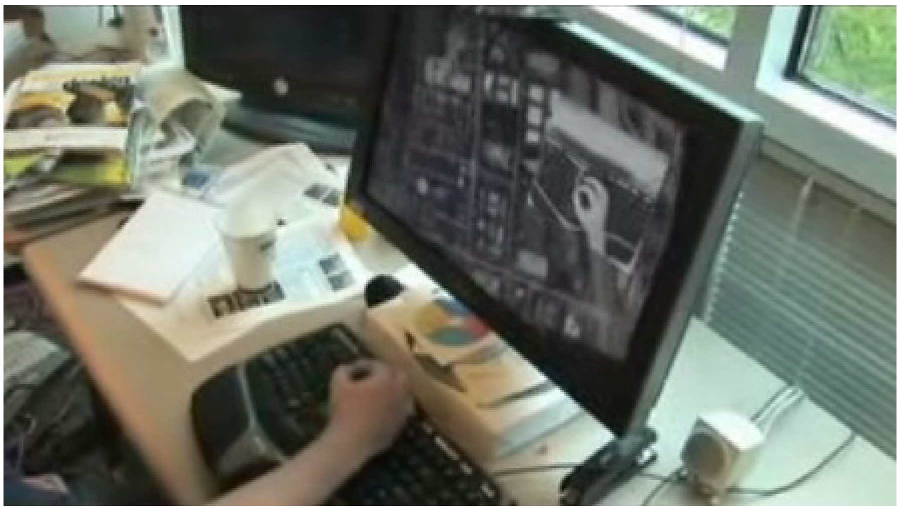


Figure 4: Real-life setup example.

2.2. Algorithms

- **Algorithm for detecting pinch gesture**

This algorithm hinges on the image segmentation and connected components analysis. When the pinch gesture is made it is possible to distinguish a hole in the middle of the hand shape. An algorithm for detecting connected components is applied. This results in the detection of the hand as the biggest connected component, but when analyzing the background it has been split into two connected components: the hole formed when the thumb and forefinger are brought together and the large background component.

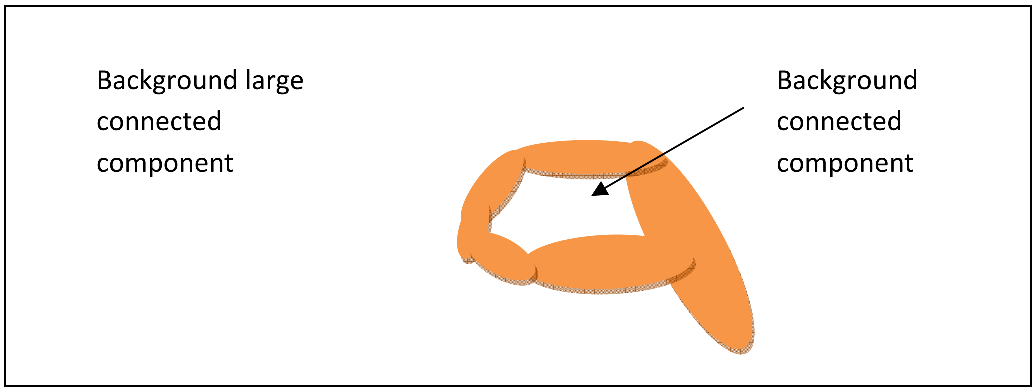


Figure 5: Connected Components Analysis

There are several image segmentation techniques, but the one we are going to use is comparing the current image to a background stored picture. If the pixel has changed significantly in value then it can be labeled as ‘hand’ but if it doesn’t then it can be labeled as ‘background’.

The criterion to identify which is the hole and which is the remaining background is the size of the connected components. The background will be the largest one and the hole the smallest, but we have to take into account that there could be more than two connected components as shown in the following figure:

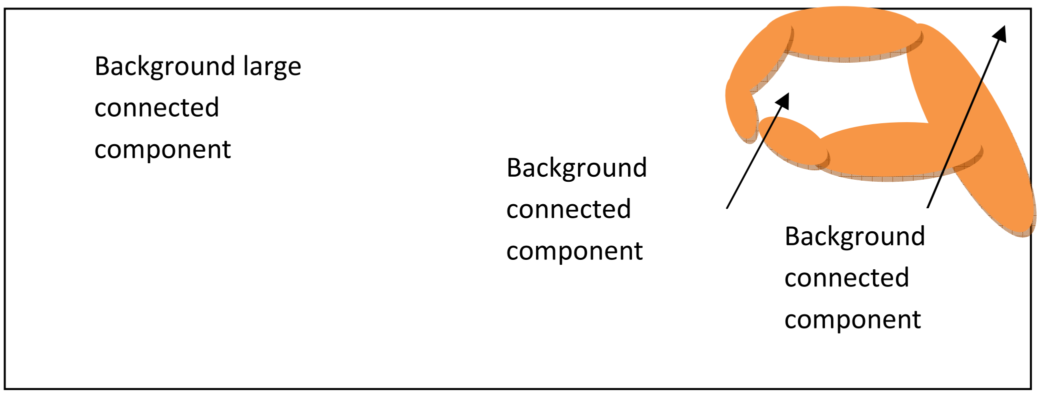


Figure 6: Eliminating connected components on the image edge

So we will define a hole as any background connected region, of significant size, smaller than the largest background connected region, but that has no pixels in the image border.

- **Algorithm for tracking**

Once the pinch gesture is detected it is possible to find the centroid of the connected component (hole). If this process is repeated frame by frame then the program will be able to track the hand and use this movement as the cursor control.

- **Algorithm for detecting clicks**

The click detection is based on the rapid closing and opening of the thumb and forefinger. This can be done by measuring the time between two consecutive pinch gesture detections.

- **Algorithm for detecting rotations**

It is possible to calculate an oriented ellipsoidal model over the connected component. Changes in this model will let our program track changes in orientation that can be used, for example, in programs that allow users to rotate objects such as maps or figures.

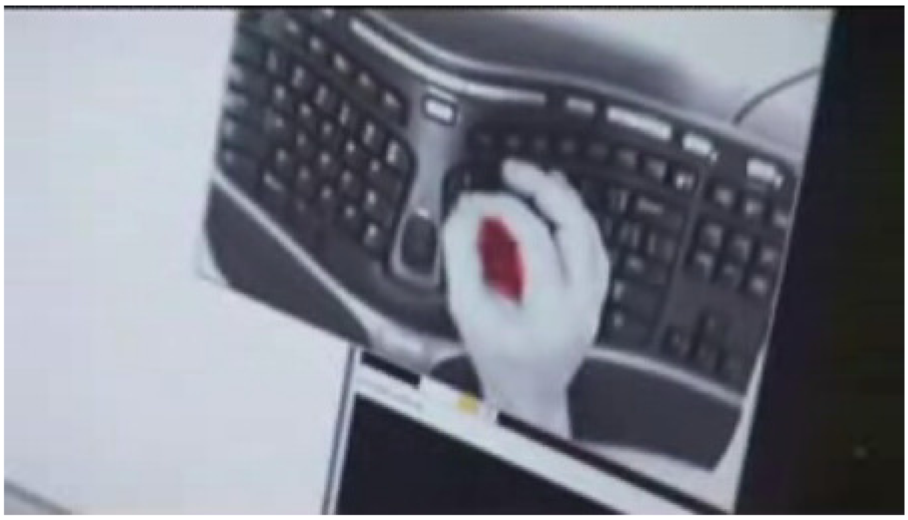


Figure 7: Pinch detected on one hand and displayed onscreen

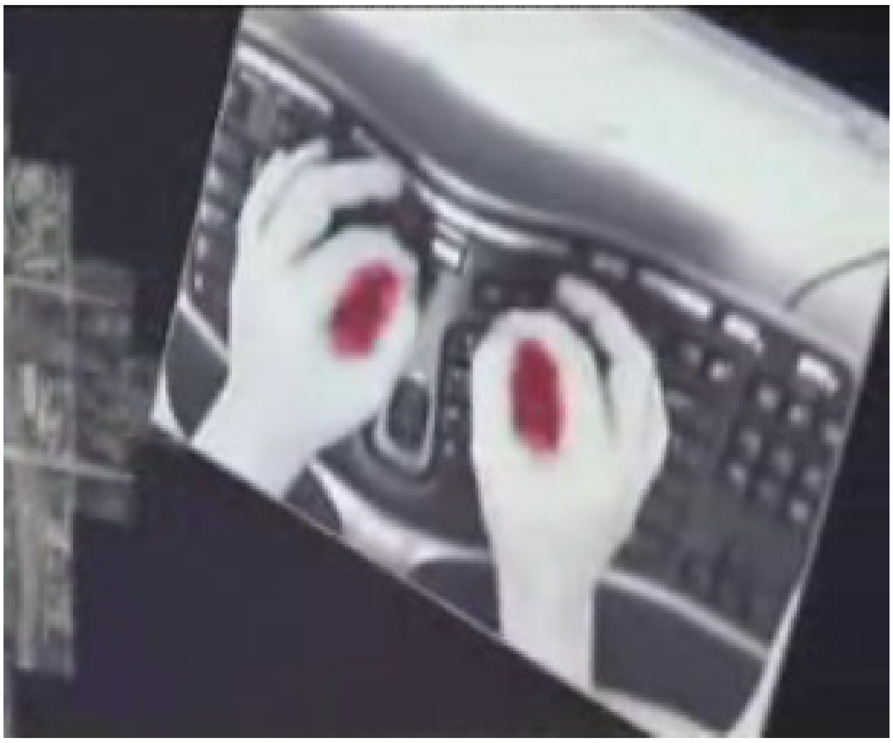
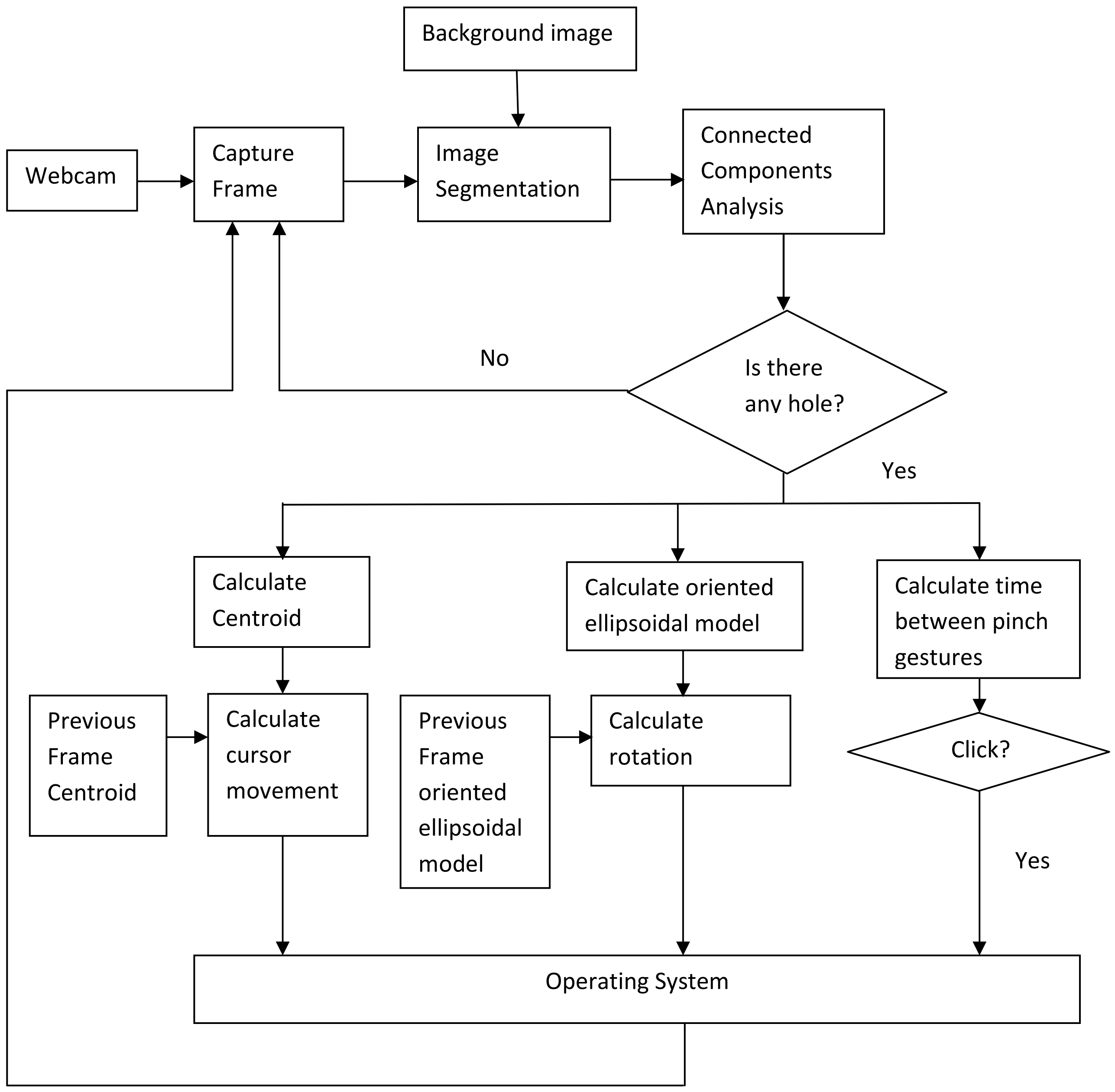


Figure 8: Pinch detected on both hands.

2.3. Dataflow


Background image
Webcam
Capture Frame
Image Segmentation
Connected Components Analysis
No
Is there any hole?
Yes
Calculate Centroid
Calculate oriented ellipsoidal model
Calculate time between pinch gestures
Previous Frame Centroid
Calculate cursor movement
Previous Frame oriented ellipsoidal model
Calculate rotation
Click?
Yes
Operating System

3. Evaluation Methodology

3.1. Tests

Testing should include several key methods. First, the system has to work for many different people. Second, the system should not recognize false gestures. Third, the system should work in more then one stable environment.

In order to fulfill these test requirements we will need volunteers to test our program. The volunteers will be given a set amount of time to learn how to use the system and to try to fool the camera with false gestures. After they have completed their test the volunteers are required to fill out a survey. If necessary we can capture the video from the volunteer to review any gestures that cause problems. Some of the volunteers will be required to perform their tests under varying environmental conditions.

3.2. Conditions for Success

While simple gestures are easily recognized under set conditions, our program has to be able to work for many varying conditions. In order to be successful, it needs to be able to work for an overwhelming majority of users. The most common gestures such as pinching (clicking) and cursor movement are required to work for all cases. The project also needs to avoid any interference including false gesture recognition. The main success involves the satisfaction of our users. We want this to be a fun and novel way for users to interact with their computer, not a hassle.

3.3. Test Conditions

For our test we will be studying the condition of whether the users can pick up the use of the pinch gesture to control the cursor within a five minute time period. This will include observing how easy it is for them to learn to accurately create a pinching gesture, and how easy it is to interact with the computer through these gestures. We will try to notice if any frustration with the system develops and why this is caused.

3.4. Test Subjects

Test subjects will be made up of our peers at Iowa State University. We will run the test on around 10-15 users, one at a time. We will hope to gain a diverse mix of test subjects through recruiting friends, roommates, and fellow computer science students.

3.5. Results Evaluations

We will evaluate our results based on two things: our notes from observations during testing, and a questionnaire filled out by the test subjects after testing. The questionnaire will have several Likert-style questions asking the user to rate their experience using gestures for cursor control. We will compute statistics for these results over all users to gain an understanding of the average opinion of the system. Combined with our notes from observations during testing, these results should help provide a clear picture of whether gestures are an appropriate alternative to the mouse.

3.6. Future improvement

We envision several possible future improvements for this technology. For starters, in his research, Wilson notes that there are several areas for improvement with his interface, including providing better cues to the user that a gestures has been recognized, such as visual or audible cues, as well as investigating better methods for implementing clicking and dragging than the current pinch-release-pinch gesture.

Additionally, there are many applications to which this technology could be applied, other than just simple cursor control. Some ideas are gestures that would show or hide windows in Windows XP/Vista, or gestures to control browser actions when using the Internet. If appropriate means are found which could implement these applications, they would certainly be feasible to create.

Sources

Lenman, S., Bretzner, L., Thuresson, B. (2002): Computer vision based hand gesture interfaces for human-computer interaction. *Technical Report TRITA-NA-D0209*, 2002, CID-172, Royal Institute of Technology, Sweden.

O'Hagan, R. & Zelinsky, A. (1997) Finger Track – A Robust and Real-Time Gesture Interface. *Australian Joint Conference on AI,* Perth.

Quek, F., Mysliwjec, T. & Zhao, M. (1995). Finger Mouse: A Freehand Pointing Interface. *Proceedings of the International Workshop on Automatic Face and Gesture Recognition.* June 26-28, 1995, pp. 372-377.

Stenger, B. (2006). Model-based hand tracking using a hierarchical Bayesian filter, *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 28 (9), pp. 1372–1384.

von Hardenberg, C. & Berard, F. (2001) Bare-Hand Human-Computer Interaction. *ACM Workshop on Perceptive User Interfaces,* Orlando, Florida.

Wilson, A. (2006). Robust Computer Vision-Based Detection of Pinching for One and Two-Handed Gesture Input. *UIST '06*. October 15-18, 2006, pp. 255-258.

Wu, Y. & Huang, T.S. (1999). Vision-based gesture recognition: A review. In A. Braffort et al., editor, *Gesture-Based Communication in Human-Computer Interaction,* 102-116, 1999.