**Windows Manager Responsive to Hand Gestures**

Lydia Gu and Yunjie Ma

**Project Proposal**

## Introduction

Our project idea comes from the Major Motion Picture, *Minority Report*, where Tom Cruise can manipulate windows on a computer screen simply by moving his hands. Our project will mimic this system by using a camera to record the hand movements. We will create a simple windows manager that outputs a video display to a projector. The image will be projected onto a table where the user can then make hand motions in front of the display to manipulate the windows. A camera will record in live time the location and shape of the hand, which will be interpreted as commands for the windows manager. The windows manager will then update the image and output the new frame to the projector.

## Physical Setup

We will be using two additional pieces of equipment: a camera and a projector. Both the camera and the projector will be situated above a white table counter. The image will be projected onto the table top so that a user can see the windows manager and move his hands over the screen. His movements will then be picked up by the camera above and sent to the labkit.

The shape of the hand will determine the command the user want to call. The user will be able to control the screen in four ways: opening a window, resizing the window, moving the window, and minimizing the window to a taskbar located at the bottom of the screen. To open a window, the user will point to the corresponding icon with his pointer finger while keeping the rest of his fingers closed in a fist. The user moves his hand to where he wants the window to open. Moving a window will require two hand positions. A hand with its fingers completely out signals the beginning of moving. The hand closes to a fist and moves to the desired spot. To release the window, the hand opens again to drop the window at the new location. Resizing a window will be similar to opening a window. The user places his pointer finger over a corner of the window. He then drags his finger to where he wants the resized corner to be. To minimize a window, the user's hand will be open, palm facing him and hand perpendicular to the screen. A downward motion will signal the actual minimization.

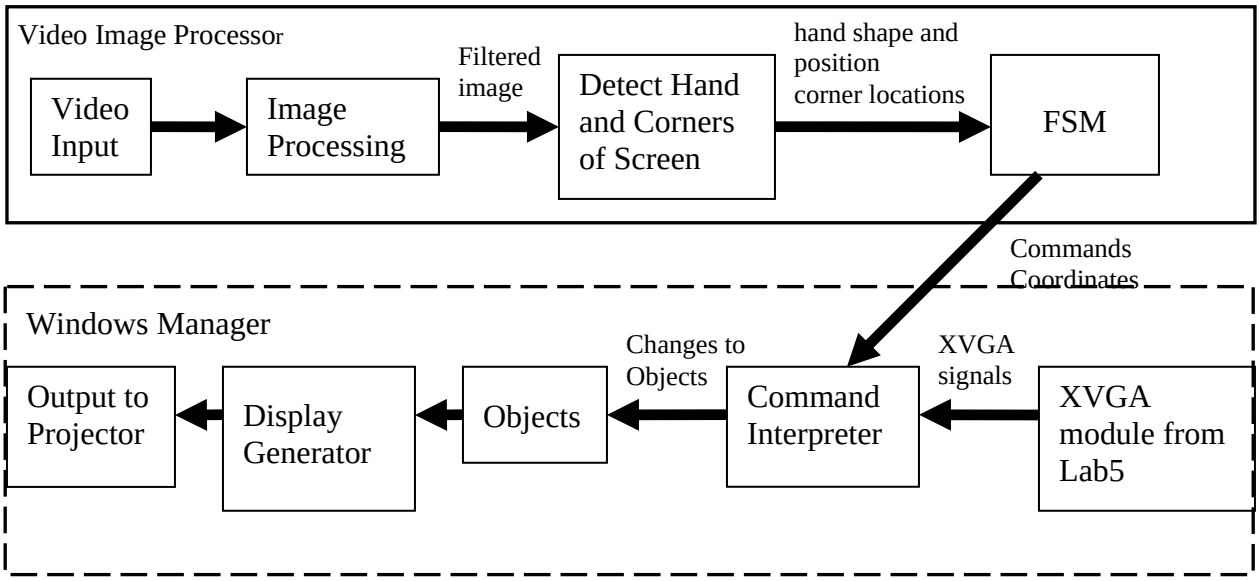
Depending on how easily the camera detects flesh color, we may use a glove for the hand detection. Regardless, the color of the hand must not show up in the image projected by the screen to ease the signal processing. To detect where the pointer finger is pointing to, we might use a bright sticker at the end of the pointer finger.

To account for different hand sizes, we will have a configuration step in the beginning. The system will record how many pixels each hand gesture requires and use that for detection.

Verilog Implementation

The system will be divided into two components, video image processor and windows manager. The signal from the camera will go into the image processor. The image processor outputs a command and a location to the windows manager. The windows manager takes the command and applies it to the window. The windows manager then outputs a video signal to the projector.

Diagram of Verilog Modules




*Video Image Processor*

Lydia will be in charge of the video processor. The processor will take in the video signal at 30 frames per second. It will first go through a module that locates the border of the screen and divides the image into pixels. Each pixel will have an x and y coordinate. Following convention, the top left corner will be (0,0). The next module detects the location and gesture of the hand. To detect the gesture, the module will sum the number of pixel that's the same color as the hand. Each gesture will require a unique range of pixels to implement. The number of pixels will indicate which gesture the user is implementing. To find the location of the hand, we will average the pixel coordinates to find the center of the hand.

The video processor will also have a FSM module. The reason behind the FSM is that the commands usually require more than one hand gesture. By using a finite state machine, the module will be able to remember previous gestures to output the correct command. Our finite state machines will most likely use case statements. If we end up using lots of RAM, then we may have to be careful with the size of the case statement.

*Windows Manager*

Yunjie will be in charge of the windows manager. Like lab 5, the manager will have a XVGA module that constantly generates XVGA signals to the projector. The current implementation will have a 1024 x 768 screen refreshed at 30 frames per second.

The windows manager will have a module that takes the commands from the video processor and update properties of each object (window, icon, or taskbar) by writing to the memory blocks. This module will also be an FSM. The states are determined by the action the screen needs to display: idle, moving, resizing, and minimizing. More states may be included for initializing each action.

One memory module will be used to store the location, type, size, and priority of each object. Because objects will need to overlap, the module will need to choose which window will be on top, depending on the priority level. The colors of the objects should not overlap. Each object will have a three bit priority indicator.

Another module will take the data from the object module and create the video display based on the objects data. This display will be written to a frame buffer, where it will be read and displayed by the video projector. The amount of memory needed for the frame buffer will be approximately 20 MB, the size of one RBG frame.

Testing and Debugging

Each major section of the project should be tested independently before they are interfaced. The video image processing can be tested by testing each hand gesture and seeing if the correct hand shapes and locations are output. The FSM for the image processor can be tested by itself. Hand shapes and locations can be simulated and the FSM outputs can be analyzed using ModelSim. The command interpreter and objects module can also be simulated with ModelSim. Finally, the image display module can be tested with the projector. After each module is tested, they can then be interfaced.