Finger Photo Editor
Project Proposal

Nadia Makan
11.11.09

Intro

The Finger Photo Editor is a unique tool for editing digital photographs. Instead of having to use a mouse to select editing options and areas of the picture, the user will do all of this with their fingers, making this editor a much more natural tool to use. This project breaks up into two main parts: gesture recognition using the camera as a sensor, and the graphics involved with the manipulation of the photograph on the screen. The camera will be pointed downward at a black rectangle on a white table surface. The black area will be divided into two sections, left and right, in which the user's corresponding hand will reside. The left hand will be able to select editing options down the left side, including: Crop, Rotate, and Restore. Then the right hand will select, for example, the locations of two opposite corners of the desired cropped picture. The user will be given visual feedback on the screen of where his left and right index fingers are pointing. Because the main focus of this project is the natural interface for the manipulation of digital images, using an actual digital photograph would present unrelated challenges and therefore will not be implemented unless there is time after the rest has been implemented.

Functionality

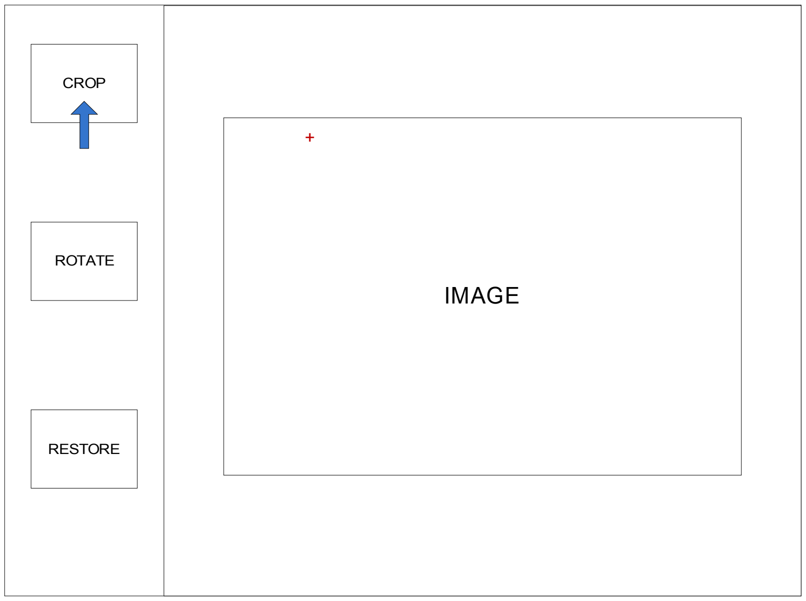

Figure 1. *This is what the user will see on the screen. The arrow and plus sign will move in response to movements of the user's left and right fingers, respectively, on the black paper.*

As mentioned above, a camera will be pointed at a black rectangle on a table top. The rectangle will be divided into virtual sections that will be displayed on the screen as shown in *Figure 1* above. The blue arrow in the left section marks the position of the tip of the user's left finger. The red plus sign in the right section marks the position of the tip of the user's right finger.

In order to crop the image, the user needs to first make sure the arrow is not pointing at any editing option by moving their left finger. Then they will move their right finger so that the plus sign is sitting where they would like to start cropping. This position will become one corner of the cropped picture. The user will then move their left finger so that the arrow's tip is within CROP's rectangular boundary. Keeping the left finger still, they will drag their right finger to where they want the opposite corner of the cropped picture to be. As they do this, they will see a rectangle drawn on the screen between the initial right finger position and the current right finger position (See Figure 2). This will help them visualize the final image. When they are satisfied with the boundary they have formed, the user will keep their right finger still, and move their left finger so that the arrow is no longer on CROP, or any other editing option. The rest of the image will be cut away, and the cropped image will be displayed on the screen.

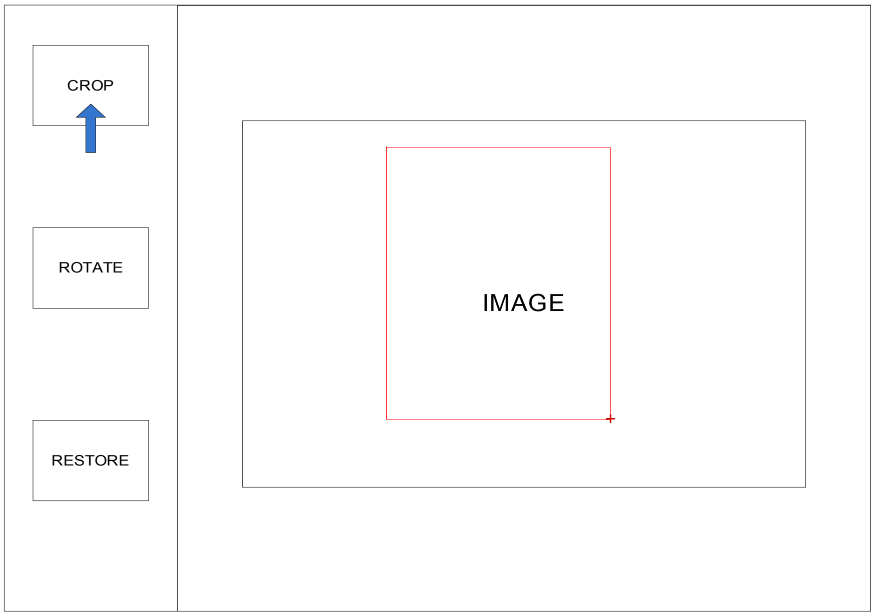

Figure 2. *Red outline to help visualize final cropped image*

To rotate the image, the user will first make sure the arrow is not pointing at any editing option. Then they will move their right finger so that the plus sign is near the corner they would like to move. Then keeping their right finger still, they will move their left finger so that the arrow's tip is within ROTATE's boundary. A circle will appear over the first corner they selected. They will then move their right finger to another corner, where they would like to move the first corner. Once they have selected the new corner, they will move the arrow off of ROTATE, and the image will rotate the desired amount (some multiple of 90 degrees). For example, if the first corner selected is the top right corner and the second is the bottom right corner, then the image will be rotated 90 degrees clockwise.

If the user has made an edit that they wish to undo, they can use the third editing option to restore the image to it's original form. To do this, all they need to do is to move their left finger so that the arrow's tip is within RESTORE's boundary, and hold it there for three seconds.

Module Descriptions

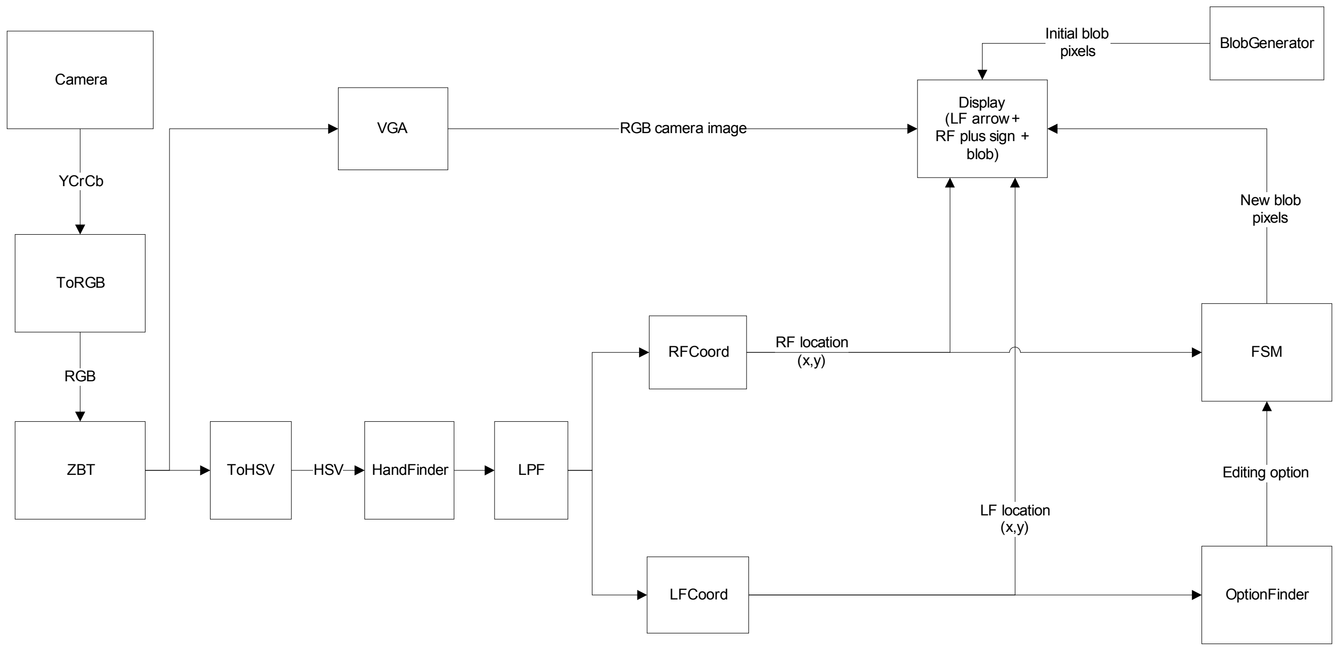

Figure 3. *Block Diagram for the Finger Photo Editor*

ToRGB

This module is fed an image from the Camera whose pixels are represented in the YCrCb color space. Here the YCrCb values are converted to RGB values. The new RGB representation of the image is then stored in a ZBT RAM.

ZBT

From the ZBT, the RGB image is sent both through a VGA cable to the screen (for testing purposes) and to the ToHSV module.

ToHSV

This module's job is to convert the pixels' RGB values to HSV values. This will make it much easier to detect the 'flesh'-colored blobs in the picture.

HandFinder

This module will go through the HSV values for each pixel, looking for Hue values that lie in a specified range of 'flesh' color values. It will output an image specifying which pixels are 'hand' pixels, and which are not.

LPF

Since the camera is not perfect and the lighting in the room is variable, the HandFinder will probably identify some pixels as 'hand' pixels that aren't actually part of a hand. This Low-Pass Filter module will filter out the lone random 'hand' pixels so that only the bigger blobs are recognized as hands.

RFCoord & LFCoord

These modules are responsible for finding the tips of the right and left fingers, respectively. They will do this by starting at the top of the image and moving down, line by line, until they find a long enough line of 'hand' pixels. RFCoord will only look at the right section of the image, and LFCoord will only look at the left section. They will then output the (x,y) coordinate of the middle of this line to represent the position of the corresponding finger tip.

OptionFinder

This module figures out which editing option the user wants to use by looking at the coordinate of their left finger tip. If this point lies within one of the three rectangular editing option 'buttons', the corresponding option will be output to the FSM.

FSM

The FSM module takes as input the Editing Option from the OptionFinder module and the RF location from the RFCoord module. It outputs the new/edited pixels of the 'blob' to be displayed on the screen, so that the user can receive visual feedback of where their fingers are and what their cropped or rotated image looks like.

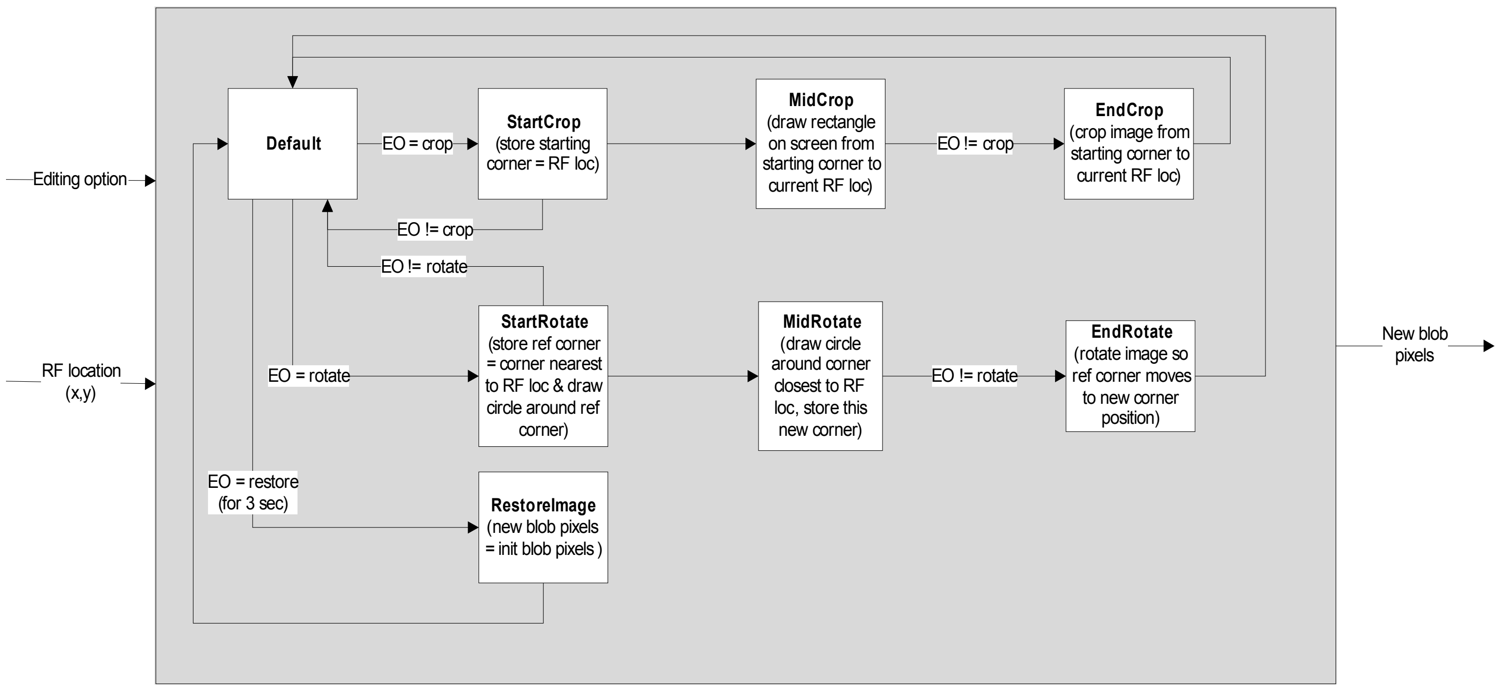
Editing option
RF location (x,y)
Default
EO = crop
StartCrop (store starting corner = RF loc)
MidCrop (draw rectangle on screen from starting corner to current RF loc)
EO != crop
EndCrop (crop image from starting corner to current RF loc)
EO != crop
EO != rotate
StartRotate (store ref corner = corner nearest to RF loc & draw circle around ref corner)
EO = rotate
MidRotate (draw circle around corner closest to RF loc, store this new corner)
EO != rotate
EndRotate (rotate image so ref corner moves to new corner position)
EO = restore (for 3 sec)
RestoreImage (new blob pixels = init blob pixels)
New blob pixels

Figure 4. Finite State Machine

Default State
This is the state the FSM will start in, and return to after an edit has been performed.

StartCrop
When the CROP option is selected, the system will enter this state. The location of the right finger will be stored as a starting point, and then the system will immediately enter the MidCrop state. If CROP is no longer selected, the system will return to the Default state.

MidCrop
While in this state, a rectangle will be drawn between the starting point and the current right finger location. When CROP is no longer selected as the editing option, then the system will enter the EndCrop state.

EndCrop
In this state, the image will be cut down to the rectangle formed between the starting corner and the current right finger location, and this cropped image will replace it on the screen. Then the system will return to the Default state.

StartRotate
When the ROTATE option is selected, the system will enter this state. The location of the right finger will be stored as a reference corner, and a circle will be drawn around it for visual feedback. Then the system will immediately enter the MidRotate state. If the ROTATE option is no longer selected, the system will return to the Default state.

MidRotate
While in this state, a circle will remain around the reference corner, and another circle will be drawn around the corner closest to the right finger location. The location of the new corner will be stored. When ROTATE is no longer selected as the editing option, the system will enter the EndRotate state.

EndRotate
In this state, the image will be rotated so that the reference corner moves to the location of the newly selected corner. Then the system will return to the Default state.

RestoreImage
When the RESTORE option is selected for at least 3 seconds, the system will enter this state. Here, the image on the screen will be replaced with the original image that was generated in the BlobGenerator module. The system will then return to the Default state.

BlobGenerator

This module generates the screen layout (as shown in Figure 1, sans the arrow and plus sign). The image in the center of the screen is the 'blob' that will be edited by the user. It should be an image that is unique at every 90 degree rotation, for testing and debugging purposes. If there is time after all of the basic functionality of the system has been implemented, this module will be replaced by a memory holding an actual digital photograph to be edited by the user.

Display

This module sums the pixels of the screen layout, the 'blob', the arrow, and the plus sign, and displays all of them on the screen.