**Your final project document should include the following sections:**

* A project summary.
* 1 prototype picture
* If your project involves some hardware
* Brief documentation on Project design, Implementation, and Testing
* Project design
* Breakdown of accomplished project prototype functions
* Texts, diagrams or pictures would be all fine
* Task: How to get hand recognition?
  + Simplest answer, we didn’t implement anything that will tell the program that the object in the footage is a hand. In fact, any object can be used, and the program will identify this as a hand. A work around was implemented. A square box is used and placed onto the screen. Anything inside of this box will be seen in the program as a hand. It is essentially up to the user to display a hand in this box.

A picture containing shape

Description automatically generated

* Task: How is the size of the yellow box determined?
  + The yellow box must be small enough to give users on weaker systems a decent frame rate. The yellow box also must be big enough for the user to put their hand into the field of view without leaving their initial position.
* Task: How to get the shape the hand?
  + Use of opencv2 functions was essential to forming the hand. The original footage is converted into a grayscale image and then smoothed using a median blur to remove some noise. The main part is applying a threshold to the image which may be different for each user to convert the image into a binary image, which is an image of only black and white pixels. Lighting is important to get an accurate shape of any hand for the step. During testing of we found that a backdrop that had a high brightness value worked best (i.e., a white painted wall).
* Task: Gave the user the ability to change the binary image’s threshold.
  + The original values to set the threshold seems to work well when the image has decent lighting. However, the original values can be set to different values if the original values don’t work for the user.

A picture containing graphical user interface

Description automatically generated

* Task: How to remove noise in the footage?
  + To smooth the image, median blurring was used to remove some noise. However, not all noise was removed while processing the footage. The median filter was set to a 3x3 mask. The reason for this was to prevent loss of framerates for the user. If a weaker system uses the program, it may become sluggish if a bigger mask was used to remove some noise. The main noise removal was a work around. If the binary image had multiple white blobs in it, it would form contour points around all white blobs. A function was used to only take the biggest white blob that had the largest set of contour points. This is leads to the assumption that the biggest white blob in the image is a hand.

A picture containing text

Description automatically generated

* Task: How were gestures found?
  + Specific gestures were not found. Instead, we opted for counting how many fingers were extended in the hand (i.e., one, two, three, four hand signs).
* Task: How were extended fingers counted?
  + This was yet another work around. The fingers themselves are not counted. Rather the defects between the fingers are counted. There can be multiple defects in an object. We noticed that one of the values returned when finding all defects in a set of contours was the distance from the defect to the farthest point. The defects we were interested in were all more than four digits long (i.e., 4000+, 5000+, etc.). So we counted all the defects that were greater than this value. The finger count may not be accurate in some situations. So, a white marker shows the user where the defects are when it is detected, and a green line in between fingers. This work around also doesn’t work well with the fifth finger, which is the pinkie. The fifth finger needs to form a defect with the palm of the hand in some situations to be detected.

Shape, arrow

Description automatically generatedShape

Description automatically generated

* Distribution of development tasks
* Each group member can be involved in one or more development task categories - Design, Implementation, Test.
* Joshua Santos
  + Design, implementation and testing of the computer vision features of the prototype.
  + Features include live camera feed that is actively being converted into grayscale and binary footage. The implementation of contour points and lines, thresholding in binary images, blurring to smooth images, and noise reduction in images. The use of multiple functions to detect defects in objects using binary images.
* Link to a 2-minute Youtube video
* For a prototype involving hardware, demo your prototype
* For a software only project, demo your software and walk through the accomplishments of the project
* A copy of your source code
* Source code readme
* Source code structure
* How to use your prototype
* How to use:
  + Prepare for use:
    - To use the yellow box must be placed on a bright background. For instance, a white wall or bright backdrop. The lighting inside this yellow box must be good otherwise the user will notice noise in the binary image.
  + Using the Binary Threshold Trackbar:
    - The Trackbar has values set that were good at the time of the program’s inception. The values are a range with Lower and Upper boundaries. The range of pixels are [0,255] for a grayscale image. For example, if the “Lower” slider is placed at 0 and the “Upper” slider is placed at 135, then only the values in the following range are allowed to be white in the binary image.
  + Where and how to place hand:
    - The hand must be placed inside of the yellow box. The palm of the hand or back of the hand must be facing the camera. Fingers are not detected, rather the gaps between your fingers are detected as defects in your hand. A white marker will show where the defect detections are on your hand. Also, a green line between the points will be shown for more confirmation. You may need to adjust your hand or fingers a certain way to count fingers.
  + To end the program:
    - To end the program, simple press the key ‘q’ to end and close all windows in the program.