Taran Soden (Team 12 CV)
10/8/2021
SEED Lab – Fall 2021
Mini Project Computer Vision Description

The team decided to utilize a blue rectangle as our marker for this mini project.

Our code utilizes an OpenCV video capture object. The first frame taken from the camera is used to determine the image height and width using the shape method which will be utilized later in quadrant calculation. All other frames after the first go through a detection algorithm.

First the frame goes through color detection. The frame is converted from BGR to HSV using the cvtColor method. Blue is detected through a mask created using the inRange method with [80,50,0] in HSV being the lower blue limit and [145,255,255] as the HSV upper blue limit. This mask is applied to the original image using a bitwise_and method. The resulting frame, called maskedHSV in the Python script, is composed of only blues detected in the original frame.

Next the frame goes through shape detection. The maskedHSV frame (post color detection) is converted from HSV to grayscale using two cvtColor method operations since HSV cannot be directly converted to grayscale in OpenCV. The gray version of the masked frame is denoised with a gaussian blur and a threshold image is created from the denoised frame using the threshold method. Shape contours are determined by running the threshold image through the findContours method. For all contours found in the image, if a shape is found that has four sides and has an area greater than 250 pixels squared, then the center position is determined of this contour. This is done with the following statements:
cx = int(M[‘m10]/M[‘m00’])
cy = int(M[‘m01’]/M[‘m00’])

Occasionally a contour would be detected that would have a 0 area, so the position is only determined if the area is greater than the 250 pixels squared threshold, else a divide by 0 error would be thrown by the above statements. This also helps reduce the frequency of false marker detection dramatically.

The contours are drawn onto the original frame so the user can see what the camera sees and the function det_quad to determine the quadrant the marker is in.

Function det_quad takes in the x and y positions of the contour, the image height, and the image width. The midlines of the image are determined by finding where half of the image width is and where half of the image height is. For this project traditional quadrant labels were used such that quadrant 1 is the top right of the image, quadrant 2 is the top left, quadrant 3 is the bottom left, and quadrant 4 is the bottom right. If the determined center of the contour has an x position greater than the vertical midline and a y position less than the horizontal midline (since the origin of the image is in the top left corner of the frame) then the marker is determined to be in quad 1. Quad 2 is determined if cx is less than the vertical midline and cy is less than the horizontal midline. The other quads are determined in a like manor. When a quadrant change is detected, the quadrant number (either 1, 2, 3, or 4) is sent to the Arduino via I2C code provided.