Counting M&Ms - Whats possible with Computer Vision

This lesson will focus on the understanding of using computer vision to count the number of occurrences each M&M color. The program will be able to recognize the different colors of M&Ms and draw a contour around the identified color. The program will also count the number of times the color occurs and prints it on to the video feed. Lastly, the program must do this in real time.

Importing libraries

We will need to import two libraries: numpy and openCV. Numpy is a library use for numerical calculation and analysis. OpenCV is the library use for computer vision, this library will allow us to find the colors of the M&Ms. Libraries help by making programming easier by allowing us to use properties and methods, rather than having to code them ourselves.

In [2]:

```
import numpy as np #imports the numpy library and sets np as an alias to use throughout t
he code
import cv2 #imports the openCV library
```

Capturing a Real-Time feed from the camera

The following code will use openCV to create videoCapture of the system's camera. The number within the videoCapture is the camera that will be used. Because we used the built in camera, we are passing zero.

In [3]:

```
cap = cv2.VideoCapture(0) #cretes cap which is the real time capture feed

# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")
```

Creating a Contour

A countour is a continuous line around points/pixels that share similar color profile or intensity levels. This will be used to isolate the colored M&Ms from the rest of the image. The function bellow creates a contour using the a color mask (describes in the "Digital Image Processing" section) to isolate that specific color. The parameters for the function are:

1. colorMask(string) -> this is the name of the color being passed
2. mask(matrix) -> this is the mask of the color being passed
3. B(integer) -> this is the blue value of the color
4. G(integer) -> this is the green value of the color
5. R(integer) -> this is the red value of the color

The function also returns a counter which is the number of contours created

In [4]:

```
def contourColor(colorName, mask, B, G, R):

    counter = 0 #initates a counter and sets it equal to zero

    #creates a contour using the mask along with a hierarchy
    #findContours has the following parameters:
        # - mask: this is the color mask that will be used to create the contour
```

```
        # - cv2.RETR_EXTERNAL: this is the hierarchy that will be used. This specific one
will be used to
        #                      minimize the number of contours created inside an existin
g contour
        # - cv2.CHAIN_APPROX_SIMPLE: this is the approximation method. This method will a
llow the program
        #                            to create a contour by using 4 points rather than u
sing 764 points.
        #                            This is done to save memory.
    contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL,
                                           cv2.CHAIN_APPROX_SIMPLE)

    #this loop will iterate through all contours created and will add frames and text to
each one
    for pic, contour in enumerate(contours):

        area = cv2.contourArea(contour) #sets area equal to area of a contour

        #the following loop will only create the frames and text to contours withh an are
a larger than 300
        if(area > 300):

            #This will set x, y, w, h equal to the bounding rectangle of the contour.
            #The "rectangle" was created using the four-point mentioned previously
            # x = the x coordinate where the rectangle begins(top left)
            # y = the y coordinate where the rectangle begins(top left)
            # w = the width of the rectangle
            # h = the height of the rectangle
            x, y, w, h = cv2.boundingRect(contour)

            #a rectangle is created from the boundingRect and placed on the video capture
            #cv2.rectangle parameters include:
            # - imageFrame: this is the video capture where the rectangle will be placed
            # - (x,y): this the beginning point of the rectangle
            # - (x + w, y + h): this is the ending point of the rectangle
            # - (B,G,R): this is the color of the rectangle
            # - 2: this is the rectangles thickness
            colorImageFrame = cv2.rectangle(imageFrame, (x, y),
                                            (x + w, y + h),
                                            (B, G, R), 2)

            #text is places on the top left part of the rectangle
            #cv2.putText parameters include:
            # - imageFrame: this is the video capture where the text will be placed
            # - (x,y): where the text is going to be placed in relationship to the video
capture
            # - cv2.FONT_HERSHEY_SIMPLEX: this is the font of the text
            # - 1.0: this is the font size
            # - (0,0,0): this is the color of the font
            # - 2: this is the thickness of the font
            cv2.putText(imageFrame, colorName, (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                        (0, 0, 0), 2)

            #this iterates the counter by one everytime a contour is created
            counter = counter + 1

    return counter
```

Displaying the Counter on to the Video Capture

Displaying the amount of times a color of M&M appears is done by counting the number of contours created for the specific color and displaying it as text onto the video capture

In [ ]:

```
def counterText(x, y, color, colorCounter):
    #THe parameters include:
    # - imageFrame: the video capture where the text will be places
    # - color + ": " + str(colorCounter): this is the text that will appear on the captur
```

```
e,
    #                                       the string combined the color passed and the n
umber of contours as a string
    # - (x,y): where the text will be placed in the video capture
    # - (0, 0, 0): this is the color of the text; in the example it is in black
    # - 2: this is the thickness of the font
    cv2.putText(imageFrame, color + ": " + str(colorCounter), (x, y),
                cv2.FONT_HERSHEY_SIMPLEX, 1.5,
                (0, 0, 0), 2)
```

Main Program and Color Limits

We have created the functions we need to track the color of the M&Ms however, we need the code to detect the specific colors, for this we need image processing and thresholding. The function bellow will be responsible for the color detection and displaying the contours onto the video feed.

In [ ]:

```
while True:
    ret, imageFrame = cap.read() #this will set imageFrame to equal to the video capture
of the camera

    #convert RGB color used by the video caputre to HSV to better
    hsvFrame = cv2.cvtColor(imageFrame, cv2.COLOR_BGR2HSV)

    #kernal that will be used for dilating the image, think of this as
    kernal = np.ones((5, 5), "uint8")

    ##########################################################################
################

    #The following is using HSV values to detect a color. The system will check if the pi
xels match the
    # ranges. If they do, a mask will be created; the mask will basically be a two-colore
d image
    # (usually black and white) where black is the areas we ignore, and white is the area
s we focus on.
    # The mask will be dilated which will remove discontinuities within the focused area

    #Set range for orage color and
    # define mask
    red_lower = np.array([160, 170, 100], np.uint8)
    red_upper = np.array([180, 255, 255], np.uint8)
    red_mask = cv2.inRange(hsvFrame, red_lower, red_upper)
    red_mask = cv2.dilate(red_mask, kernal)

    #Set range for orage color and
    # define mask
    orange_lower = np.array([5, 140, 170], np.uint8)
    orange_upper = np.array([13, 230, 255], np.uint8)
    orange_mask = cv2.inRange(hsvFrame, orange_lower, orange_upper)
    orange_mask = cv2.dilate(orange_mask, kernal)

    #Set range for blue color and
    # define mask
    blue_lower = np.array([88, 80, 100], np.uint8)
    blue_upper = np.array([120, 255, 255], np.uint8)
    blue_mask = cv2.inRange(hsvFrame, blue_lower, blue_upper)
    blue_mask = cv2.dilate(blue_mask, kernal)

    #Set range for green color and
    # define mask
    green_lower = np.array([32, 70, 100], np.uint8)
    green_upper = np.array([60, 255, 155], np.uint8)
    green_mask = cv2.inRange(hsvFrame, green_lower, green_upper)
    green_mask = cv2.dilate(green_mask, kernal)

    #Set range for yellow color and
    # define mask
```

```
    yellow_lower = np.array([27, 80, 180], np.uint8)
    yellow_upper = np.array([30, 255, 255], np.uint8)
    yellow_mask = cv2.inRange(hsvFrame, yellow_lower, yellow_upper)
    yellow_mask = cv2.dilate(yellow_mask, kernal)

    #Set range for brown color and
    # define mask
    brown_lower = np.array([2, 125, 50], np.uint8)
    brown_upper = np.array([12, 255, 80], np.uint8) #15, 255, 150
    brown_mask = cv2.inRange(hsvFrame, brown_lower, brown_upper)
    brown_mask = cv2.dilate(brown_mask, kernal)

    ##########################################################################################
#################

    #The following code segments are use to created the contour of each color, notice how
we are using the
    # functiion that was created previously. Each parameter is passed and the function do
es the rest. Also
    # notice how we used the other function as well to put the text onto the video.

    # Creating contour to track red color
    redCounter = contourColor('Red',red_mask, 0, 0, 255)
    counterText(50, 50, "Red", redCounter)

    # Creating contour to track orange color
    orangeCounter = contourColor('Orange', orange_mask, 0, 100, 255)
    counterText(50, 100, "Orange", orangeCounter)

    # Creating contour to track blue color
    blueCounter = contourColor('Blue', blue_mask, 255, 0, 0)
    counterText(50, 150, "Blue", blueCounter)

    # Creating contour to track green color
    greenCounter = contourColor('Green', green_mask, 0, 255, 0)
    counterText(50, 200, "Green", greenCounter)

    # Creating contour to track yellow color
    yellowCounter = contourColor('Yellow', yellow_mask, 0, 255, 255)
    counterText(50,250, "Yellow", yellowCounter)

    # Creating contour to track brown color
    brownCounter = contourColor('Brown', brown_mask, 42, 42, 97)
    counterText(50, 300, "Brown", brownCounter)

    ##########################################################################################
###################

    #The code bellow displays the image and tittles it M&M Counting; it also checks for q
to be pressed
    # to stop the image capture.

    cv2.imshow('M&M Counting', imageFrame)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
```