MULTIPLE COLOUR DETECTION IN REAL TIME.

Aarushi Juneja -102103781
Aalok -102103846
Daksh-102153043
Hardik Verma-102283044

A RESULT OF HUMAN-AI COLLABORATION

OBJECTIVE:

To detect multiple colours in real time (RGB).

THE BASIC IDEA

- The fundamentals of computer are used to track the three colours – red, green and blue(RGB).

- When the code is run, a window will open using the webcam, and if there is any of these three colours present, rectangular boxes of respective colours (Red for Red, Blue for Blue, and Green for Green) will be displayed around the objects with those colours, along with text indicating the name of the colour on top of the object.

COLOUR DETECTION USING OPEN CV.

• Open CV is a computer vision library developed by intel.

• It is a collection of C++ classes that implement popular image processing and computer vision algorithms.

• Colour detection is one of the most important and challenging computer vision task.

• Our aim is to retrieve and detect the three colours(red, green and blue) from video frame.

USED CODE AND MODULES:

- NUMPY (IMPORTED AS NP)
- CV2 - OPENCV (OPEN SOURCE COMPUTER VISION LIBRARY)
- CV2.COLOR_BGR2HSV
- CV2.INRANGE()
- CV2.DILATE()
- CV2.BITWISE_AND()
- CV2.FINDCONTOURS()
- CV2.RECTANGLE()
- CV2.PUTTEXT()
- CV2.FONT_HERSHEY_SIMPLEX
- CV2.IMSHOW()
- CV2.WAITKEY()
- CV2.DESTROYALLWINDOWS()

```
# Reading the video from the
# webcam in image frames
_, imageFrame = webcam.read()

# Convert the imageFrame in
# BGR(RGB color space) to
# HSV(hue-saturation-value)
# color space
hsvFrame = cv2.cvtColor(imageFrame, cv2.COLOR_BGR2HSV)

# Set range for red color and
# define mask
red_lower = np.array([136, 87, 111], np.uint8)
red_upper = np.array([180, 255, 255], np.uint8)
red_mask = cv2.inRange(hsvFrame, red_lower, red_upper)

# Set range for green color and
# define mask
green_lower = np.array([25, 52, 72], np.uint8)
green_upper = np.array([102, 255, 255], np.uint8)
green_mask = cv2.inRange(hsvFrame, green_lower, green_upper)

# Set range for blue color and
# define mask
blue_lower = np.array([94, 80, 2], np.uint8)
blue_upper = np.array([120, 255, 255], np.uint8)
blue_mask = cv2.inRange(hsvFrame, blue_lower, blue_upper)

# Morphological Transform, Dilation
# for each color and bitwise_and operator
# between imageFrame and mask determines
# to detect only that particular color
kernal = np.ones((5, 5), "uint8")
```

```
# For red color
red_mask = cv2.dilate(red_mask, kernal)
res_red = cv2.bitwise_and(imageFrame, imageFrame,
                          mask = red_mask)

# For green color
green_mask = cv2.dilate(green_mask, kernal)
res_green = cv2.bitwise_and(imageFrame, imageFrame,
                            mask = green_mask)

# For blue color
blue_mask = cv2.dilate(blue_mask, kernal)
res_blue = cv2.bitwise_and(imageFrame, imageFrame,
                           mask = blue_mask)

# Creating contour to track red color
contours, hierarchy = cv2.findContours(red_mask,
                                       cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)

for pic, contour in enumerate(contours):
    area = cv2.contourArea(contour)
    if(area > 300):
        x, y, w, h = cv2.boundingRect(contour)
        imageFrame = cv2.rectangle(imageFrame, (x, y),
                                   (x + w, y + h),
                                   (0, 0, 255), 2)

        cv2.putText(imageFrame, "Red Colour", (x, y),
                    cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                    (0, 0, 255))
```

USED CODE AND MODULES:

```
# Program Termination when 'esc' is pressed
cv2.imshow("Multiple Color Detection in Real-TIme", imageFrame)
if cv2.waitKey(10) & 0xFF == 27:
    cap.release()
    cv2.destroyAllWindows()
    break
```

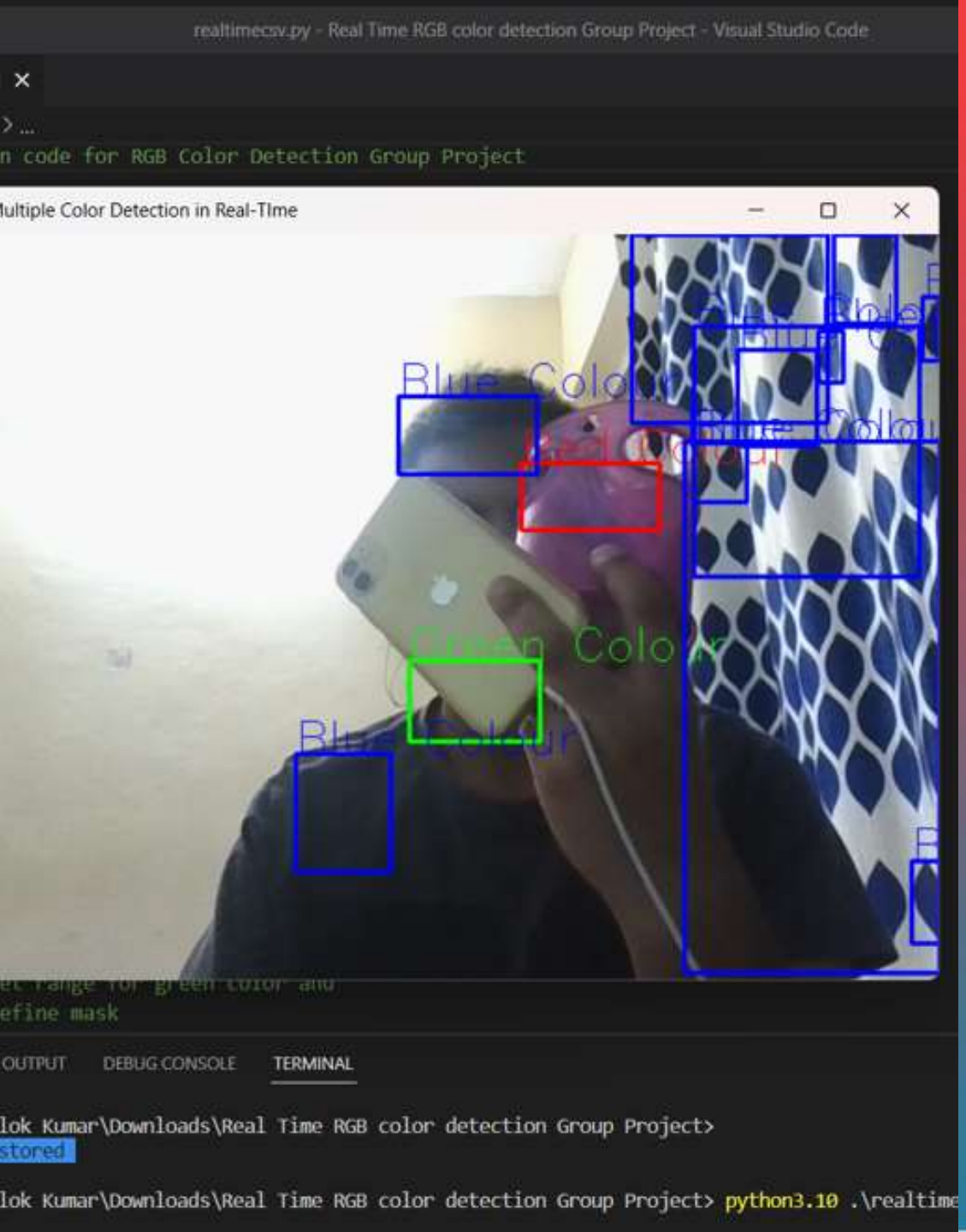
```
# Creating contour to track red color
contours, hierarchy = cv2.findContours(red_mask,
                                       cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)

for pic, contour in enumerate(contours):
    area = cv2.contourArea(contour)
    if(area > 300):
        x, y, w, h = cv2.boundingRect(contour)
        imageFrame = cv2.rectangle(imageFrame, (x, y),
                                   (x + w, y + h),
                                   (0, 0, 255), 2)

        cv2.putText(imageFrame, "Red Colour", (x, y),
                    cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                    (0, 0, 255))

# Creating contour to track green color
contours, hierarchy = cv2.findContours(green_mask,
                                       cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)

for pic, contour in enumerate(contours):
    area = cv2.contourArea(contour)
    if(area > 300):
        x, y, w, h = cv2.boundingRect(contour)
        imageFrame = cv2.rectangle(imageFrame, (x, y),
                                   (x + w, y + h),
                                   (0, 255, 0), 2)

        cv2.putText(imageFrame, "Green Colour", (x, y),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    1.0, (0, 255, 0))

# Creating contour to track blue color
contours, hierarchy = cv2.findContours(blue_mask,
                                       cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)
for pic, contour in enumerate(contours):
    area = cv2.contourArea(contour)
    if(area > 300):
        x, y, w, h = cv2.boundingRect(contour)
        imageFrame = cv2.rectangle(imageFrame, (x, y),
                                   (x + w, y + h),
                                   (255, 0, 0), 2)

        cv2.putText(imageFrame, "Blue Colour", (x, y),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    1.0, (255, 0, 0))
```

WORKING AND IMPLEMENTATION:

This code segment is used to create contours around objects of specific colors (red, green, and blue) in an image, and draw rectangles around those objects with corresponding color labels using OpenCV library in Python.

A loop iterates over each contour found in the mask image using the "contours" variable. For each contour, the area is calculated using cv2.contourArea() function, which gives the area of the contour region.

The process is repeated for each color, so that contours are drawn and labels are put for objects of red, green, and blue colors separately, using their respective masks and corresponding color codes for rectangles and labels.



WORKING OF MODEL:

Overall, this code segment captures video frames from a webcam, converts them to HSV color space, creates binary masks for red, green, and blue colors, performs morphological transformations, and then applies bitwise AND operation to detect and highlight the regions of the respective colors in the video frames.

SOME OF THE COMMON USES OF REAL-TIME MULTIPLE COLOUR DETECTION INCLUDE:

- Computer Vision and Image Processing
- Medical Imaging
- Augmented Reality and Virtual Reality
- Education and Research
- Industrial Automation
- Art and Design
- Human-Computer Interaction

CODE REPOSITORY:

HTTPS://DRIVE.GOOGLE.COM/FILE/D/1FMO7YIBWWF33UI1J77X-SYWWQFSBRJLX/VIEW?USP=SHARE_LINK

THANK YOU