Group 5

6388030 Kulawut Makkamoltham
6388040 Ariya Phengphon
6388133 Pitchaya Teerawongpairoj
6388196 Sasima Srijanya

Background Removal

1. Read foreground image and background image
2. Convert foreground image to HSV for read the color of the image
3. Define upper and lower HSV threshold for creating a green mask later.
4. Create a green mask.
5. Inverse mask to make green screen to black and object (people) to white.
6. Resize the background image to have the same size as the foreground image.
7. Use an inverse mask (5.) to extract foreground image and background image by using cv2.bitwise_and().
8. Combine foreground mask (already remove green screen) and background mask (already remove object).
9. Convert BGR color (matplotlib) to RGB.
10. Show the result image.

Output # 1

Output # 2

Output # 3

Money Count

1. Read money image (BGR).
2. Convert the BGR image to Grayscale image.
3. Circle detection (Coin):
   - 3.1. Define the cv2.HoughCircles() to find circles in the image. We use matplotlib to estimate the area radius of the circle and adjust parameters slightly to make it able to detect wanted circles.
   - 3.2. We use NumPy's ogrid function to create two grids of coordinates y and x with dimensions matching the shape of the input image. Then, we used the circle formula $(x-a)^2 + (y-b)^2 = r^2$ to check whether the pixel is in the circle area. Next, we use a boolean mask and extract blue, green, and red from the image and store in r, g, b variables. After that, we will find average value of blue, red, and green and change to uint8 type. Noted that using this method is the optimized version. The original version use enumerate to iterate the image and filter those ones that are in the circle area. The for loop can take up to 1 minute to process the 2000 x 1500 image.
   - 3.3. Get coin value by:
     - 3.3.1. Condition#1) Setting cutting threshold of all channels Red, Green, and Blue (Threshold of white: 180 and black: 50)
     - 3.3.2. Condition#2) If it does not match with the first condition, then find the maximum value of each channel (Red, Green, and Blue).
       Since rgb color of each channel is similar, then the minimum and maximum color value shouldn't be much different. From the observation, the minimum value that determine red, green, or blue is lower than the maximum value at 60. Therefore, we picked 50 to be safe to find the color dominant. (e.g., If one pixel has RGB is (1,28,159) then it should return the coin value of Blue (20).
     - 3.3.3. Condition#3) Else, return the coin value of Gray (100).
   - 3.4. We use for loop to draw circles, find average circle rgb value from calling avg_circle_value(img, circle)function from 3.2, and call the get coin value function and add to the result variable.
4. Rectangle detection (Bank note detection):
   - 4.1. First, use median blur and morphology (custom for each image) to filter out the noises from the table and preprocess the image.
   - 4.2. After image preprocessing to remove noises and make the object separated. Then, it detects edges from grayscale image using Canny edge detection.
   - 4.3. After detecting edges, it needs to scope the image using MORPH_DILATE and MORPH_CLOSE to close the unconnected detected edges from the image. The number of times on looping is manually calculated and tested. (To remove noise from image - wooden table, medianBlur() with other morphology if needed are used to enhance the edge detection)
   - 4.4. Then, use those edges to find contour and draw the bounding box around the detected edges. Moreover, since the rectangle does not work perfectly as it has

some unused components (not Bank note detection), we need to estimately filter width and height value. (Select only the long detected rectangle)

4.5. After getting each rectangle surrounding each Bank note detection, then we want to scope down the image to the center (Pokemon focused) to use those cell colors in processing.

4.6. Then, we decided to use HSV rather than RGB because it is easier to extract and working on each color (HSV: use Hue instead of R, G and B channels) and check each pixel with the setting condition to classify the color to assigned Bank note detection price.

4.7. We calculate the ROI value of Pokemon focused and convert to HSV. Then we use converted ROI to calculate average color. Next, we use average color to separate HSV channels and print out the average color to check H. After that we match H with the given conditions and sum up to get the total amount of Bank note detection.

5. Combine total amount of Bank note detection and amount of Coin detection. Finally, display the ultimate totals.

Output


```
C:\Users\DELL\Desktop\4th year 1st semester\digital image processing\midterm exam coding>python -u "c:\Users\DELL\Desktop\4th yea
r 1st semester\digital image processing\midterm exam coding\money_count"
201 226 240 1
15 86 53 11
58 97 176 31
50 89 171 51
32 33 33 101
88 107 117 201
158 87 99 206
164 74 87 211
200 216 220 212
209 232 240 213
159 77 78 218
126 18 21 223
Coin counted:  223
Area: (981, 779) to (1040, 809) Center: (923, 750), Average Color: [ 59 110 145]
Area: (702, 781) to (758, 820) Center: (646, 742), Average Color: [ 78 110 151]
Area: (1277, 778) to (1341, 815) Center: (1213, 741), Average Color: [ 74 118 157]
Area: (407, 796) to (462, 855) Center: (352, 737), Average Color: [118  99 162]
Area: (1558, 753) to (1620, 774) Center: (1496, 732), Average Color: [ 42 140 152]
Area: (1552, 291) to (1612, 312) Center: (1493, 271), Average Color: [ 41 146 162]
Area: (431, 311) to (490, 370) Center: (373, 253), Average Color: [117 102 164]
Area: (1258, 303) to (1315, 356) Center: (1201, 250), Average Color: [106 122 139]
Area: (706, 284) to (764, 321) Center: (648, 247), Average Color: [ 74 109 175]
Area: (993, 269) to (1049, 295) Center: (938, 243), Average Color: [ 52 120 156]
Bank + Coin counted 4013
```