Convert the image to grayscale

The image was converted to grayscale because the method used to detect the bill borders—adaptive thresholding—only works on single-channel images. This happens because binarization needs to decide whether each pixel becomes either white (255) or black (0), based solely on intensity, not color.

Adaptive threshold

- This method was used to better distinguish the background from the bills in case the image has lighting variations.
- For greater precision, the threshold value (a limit number that determines whether a pixel turns white or black) was calculated using a Gaussian-weighted sum of the neighboring pixel values.
- The constant cv2.THRESH_BINARY_INV was used because the colored bills were the objects of interest. Normally, darker pixels (with lower values than the threshold) would be turned black, but it was preferable to make the bills white. This is because the findContours() function wraps white objects on a black background. So, by inverting the binary output, the bills turn white and the original white background becomes black.

Apply morphology to the binarized image

- Morphological operations were applied to the binarized image to close small holes inside the bills and make them easier to detect with computer vision.
- This technique works by sliding a kernel over the image, which helps close gaps inside the white regions (bills). As a result, the contour detection later on doesn't mistakenly split one bill into several objects.

Finding the bills' contours

- Contours were found using findContours(). The mode cv2.RETR_EXTERNAL was chosen to extract only the external contours of the bills, ignoring any inner ones. cv2.CHAIN_APPROX_SIMPLE was used to simplify the contour representation by removing redundant points.
- This function returns a list of arrays with the coordinates of each contour. To help visualize the contours detection, they were all drawn in red.

Cutting bills delimited by contours

- Using the coordinates of each detected contour, the corresponding region of each bill was cropped from the original images.

- This ensures that OCR is applied only to unmodified bill regions, free from all the contours drawn before.

Gaussian blur

- Each cropped bill image was blurred using a Gaussian filter to remove small unnecessary details that might interfere with OCR.
- The blur helps the OCR focus on the large printed value.

OCR application

- The detected text by ocr was compared to common confusions in numbers and characters such as s with 5, o with 0 and I with 1, and it was corrected.
- Then it was verified if a string of a possible number was within the detected text. This was stored in detected_value and if it was true, it was stored in a dictionary with the bill value as the key and the number of occurrences as the value. The detected value was also added to the total sum of money.

Green contours and text

For better visual representation, a green rectangle was drawn around each detected bill, and its value was displayed in blue text above it.

Final results

- The number of each type of bills in ascending order and the total amount of money was displayed in console.
- Finally, the created image was saved in a .jpg file.