
cv::findingCoins();
Ian Dudder & Karran Singh

Project Goals

Project Idea: Develop Coin Recognition Software

Objectives:

1. Recognize elliptical objects in an image.
2. Determine which elliptical objects are coins and which are not.
3. Determine if the coin is heads up or tails up.
4. Determine the type of each coin (i.e. penny, nickel, dime, etc).
5. Report the value of the coins in the image as a collection.

Methodology

Our strategy to find coins in an image is:

1. Use Canny's Edge detection to identify notable features in the image.
2. Find the contours in the image to identify the major objects.
3. Draw a best-fit ellipse around the contours to determine which objects are elliptical and which are not.
4. Compare each elliptical shape to a heads-up and tails-up template for each type of coin.
5. Based on the number of matching edges (i.e. lines/features), conclude whether or not the object is a coin.

Finding Contours

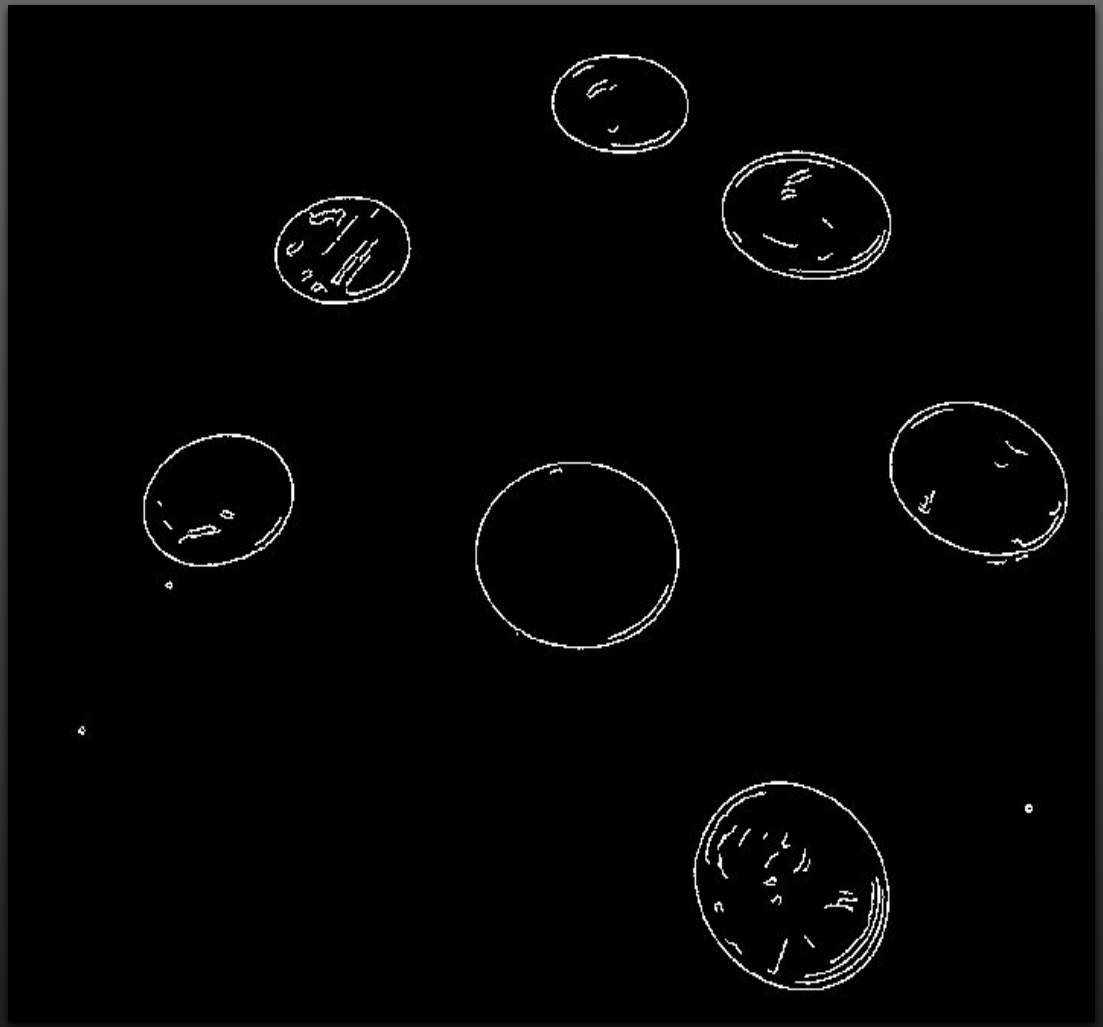
Benefits of using cv::findContours() for shape recognition.

- Provides the exact locations of objects.
- Allows us to create a bounding rectangle to extract a sub-image from.
- Allows us to create a best-fit ellipse to identify elliptical objects.

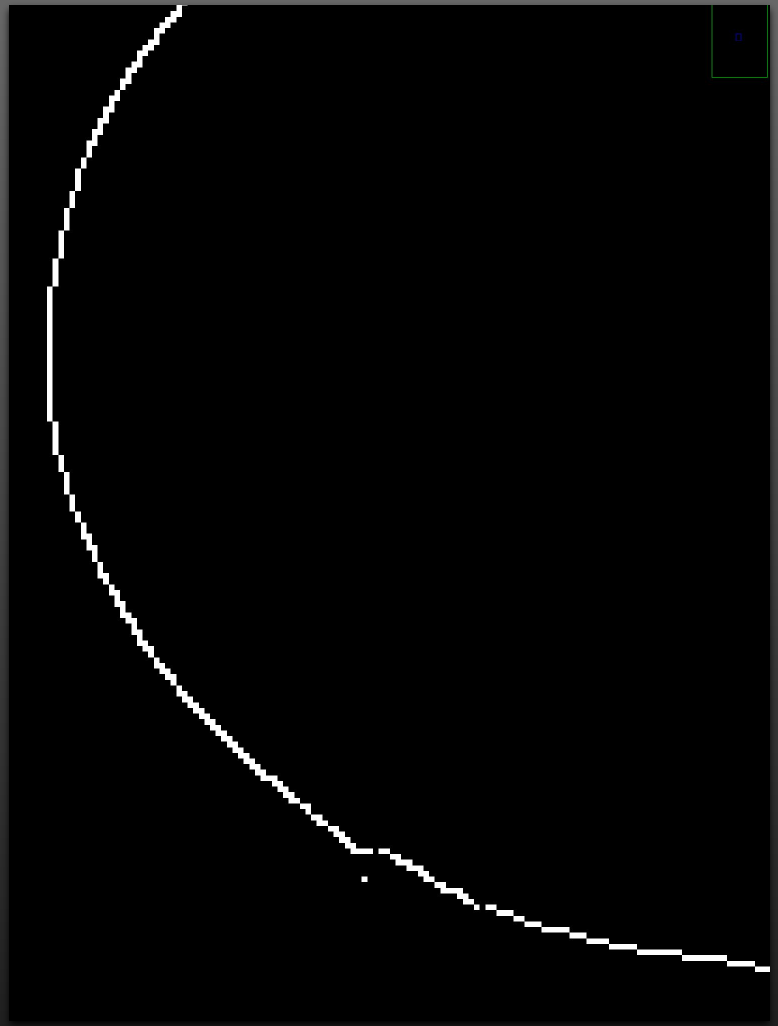
Finding Contours

Problem: Contours required continuous lines to detect a shape. Any break in a line would cause an object to go undetected.

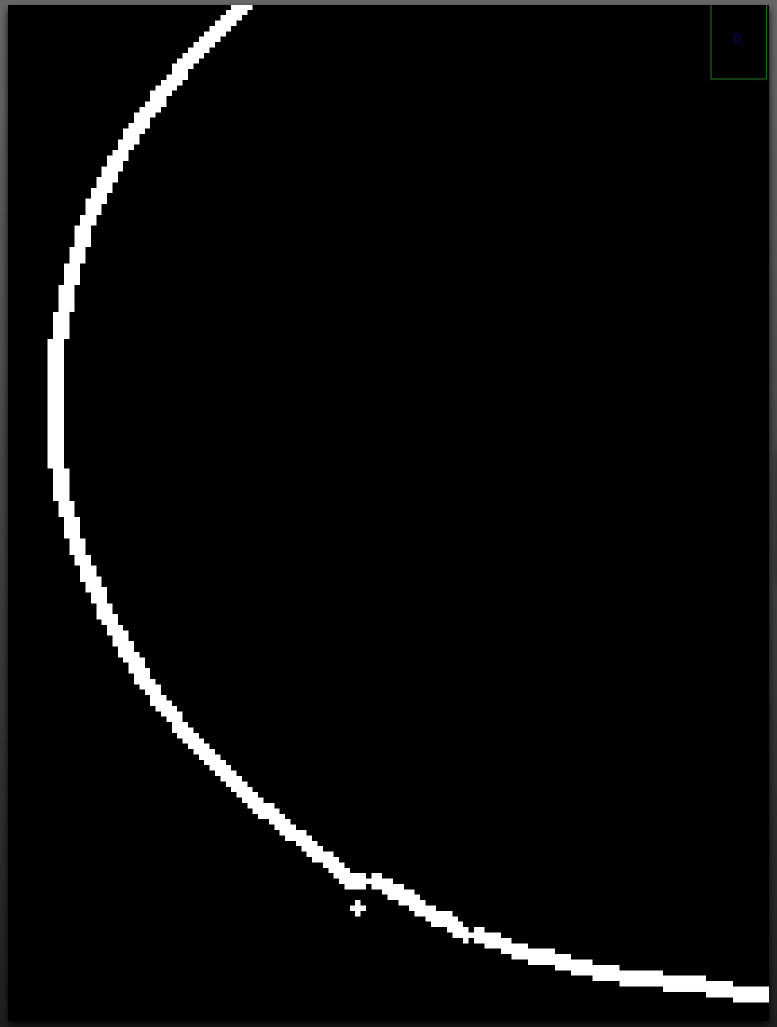
Solution: Dilate the image to re-connect any broken lines and identify all the important shapes.



Canny’s Edge Detection

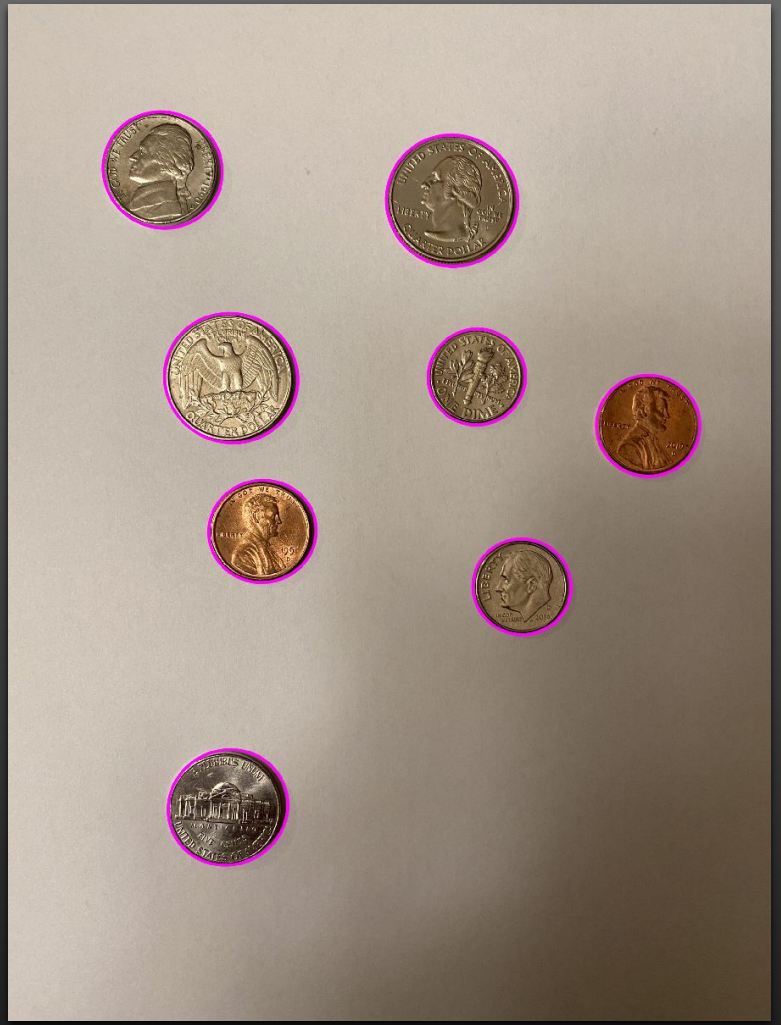


Broken edges



cv::dilate

Good results with dilation



Template Matching

Goal: Compare each elliptical object to a template of each type of coin to determine if the object is a coin.

Process:

- Scale down the template image to be the same size as the contour object.
- Compare the contour object to each rotation of the template until a best match is found.
- Compute the percentage of matching features and use the data to make a conclusion.

Template Matching

Patch

Edge Image

Steps:

1. Create a sub-image (patch) for each contour found.
2. Create an edge image from the patch.

Template Matching

3. Compare the contour object to the coin template images. Try every rotation of each template and keep track of how many edges match at each orientation.

Contour Patch

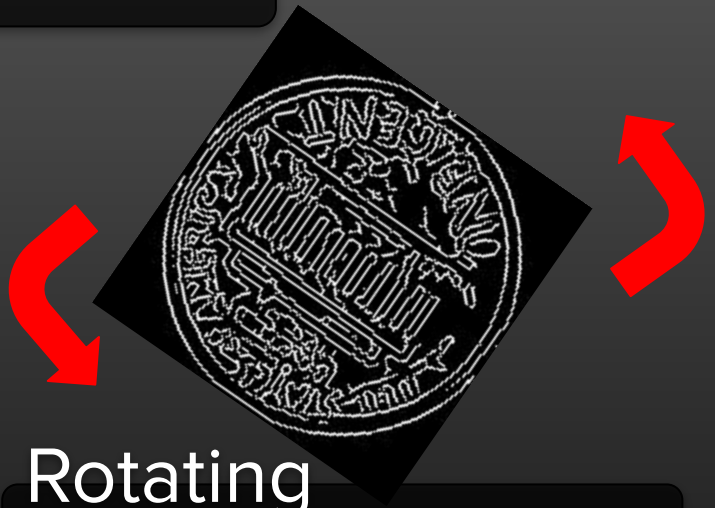
Rotating Template

Template Matching

4. Remember which coin's template gave the best match.

Patch

=



Rotating Template

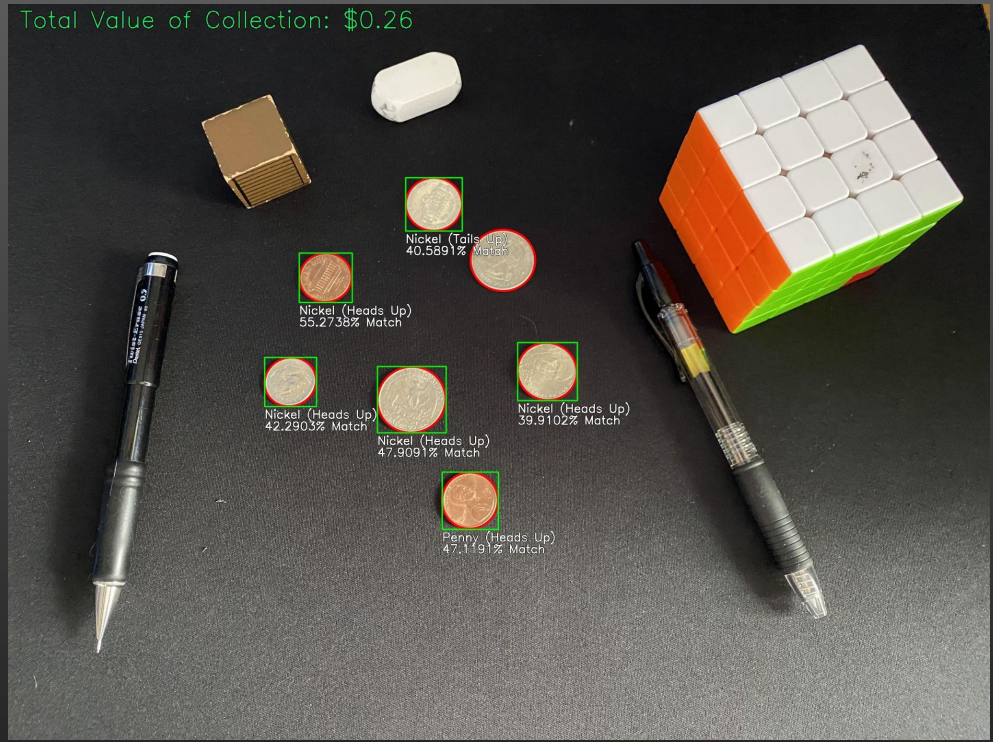
Results

After implementing our program to follow our methodology, we ran our program with 10 test images. Based on the program's output, we evaluated our results based on three criteria:

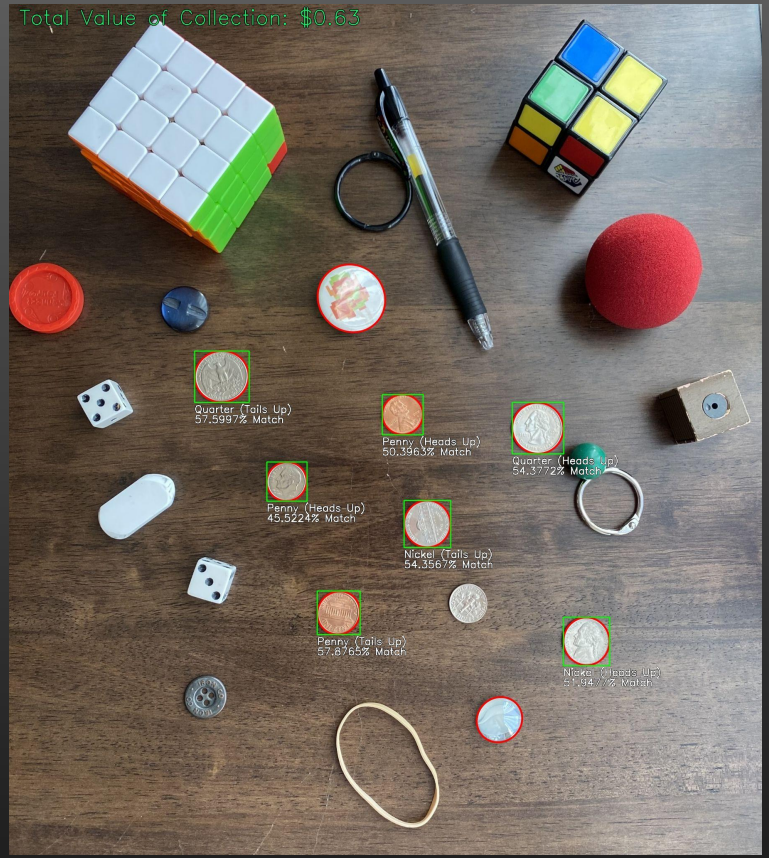
1. Coins Recognition Rate: the ratio of coins found to coins not found.
2. Orientation Accuracy: the percentage of coins whose orientation (heads-up/tails-up) was correctly identified.
3. Coin-Type Accuracy: the percentage of coins whose type was correctly identified.

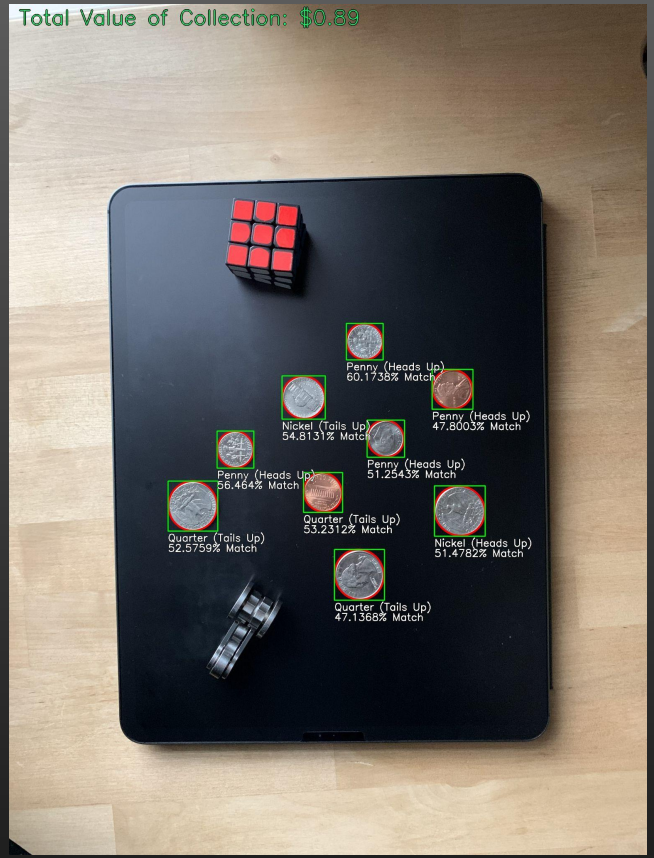
Output Images

Output Images

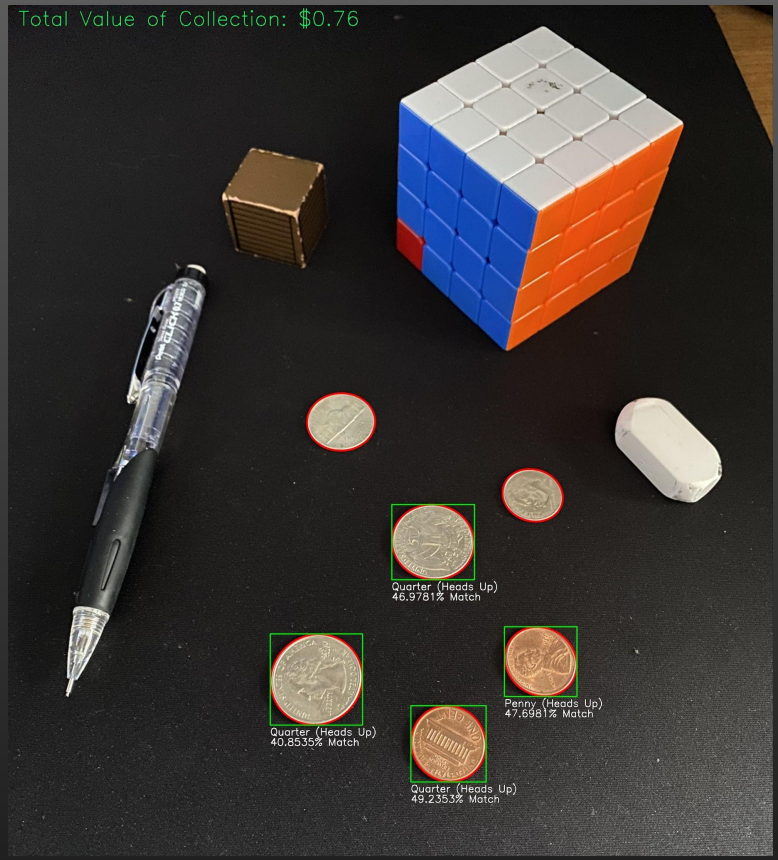


Output Images





Output Images



Output Images

Results

The average identification rates for each of the three criteria after running 10 test images is:

- Coin Recognition Rate: 81%
- Orientation Accuracy: 81%
- Coin-Type Accuracy: 63%

Lessons Learned

Through this project, we learned how to:

1. Identify major features in an image with edge detection.
2. Identify major shapes in the image through by finding contours.
3. Apply dilation to reconnect broken edges.
4. Use templates to identify objects in an image.
5. Annotate images with a bounding rectangle a level of confidence in the identification.
6. Use C++ 17's filesystem to feed our program an entire folder of images.
7. Leverage parallel processing by using multiple threads to process images.