Computer Vision - Project 1

Exploration

The first step of our exploration was the use of the Canny edge detection algorithm and Hough Lines. We experimented with different threshold values to see their impact on the results.

Unfortunately, by itself, edge detection was not nearly enough to obtain good results. We moved on to exploring different segmentation algorithms, such as: Otsu's Method, Grabcut Algorithm, Adaptive Threshold, and K-means. We had some good results with some of them, but the K-means was by far the most consistent one, and hence, it was used in our final algorithm.

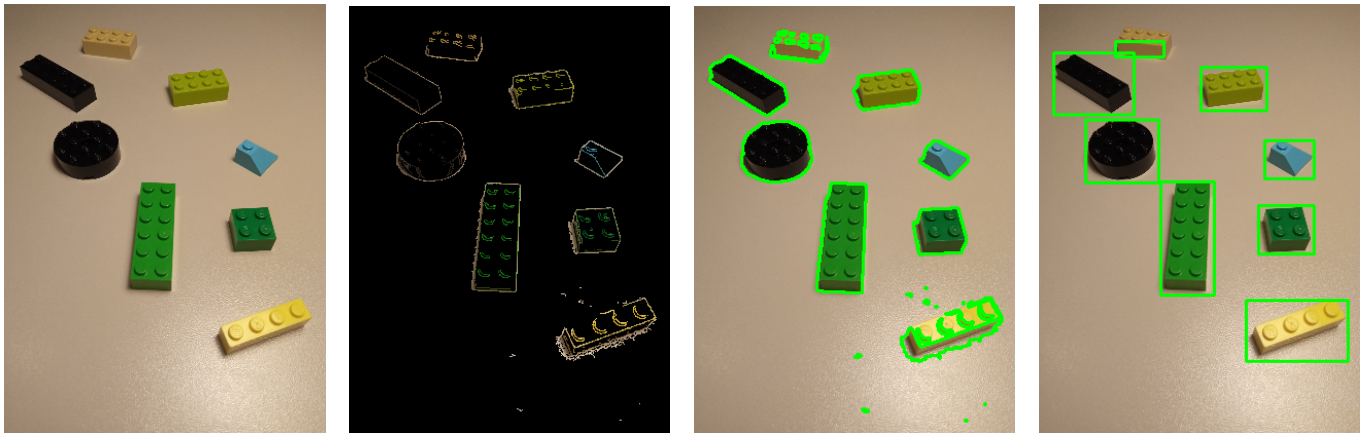
One of the main problems we had in the development of our program was adjusting the values of the parameters for the algorithms used. This was especially visible regarding the size of the kernel used in applying a Gaussian blur. The ideal values were different for each image, as such, we decided to implement step 4b which drastically improved our results.

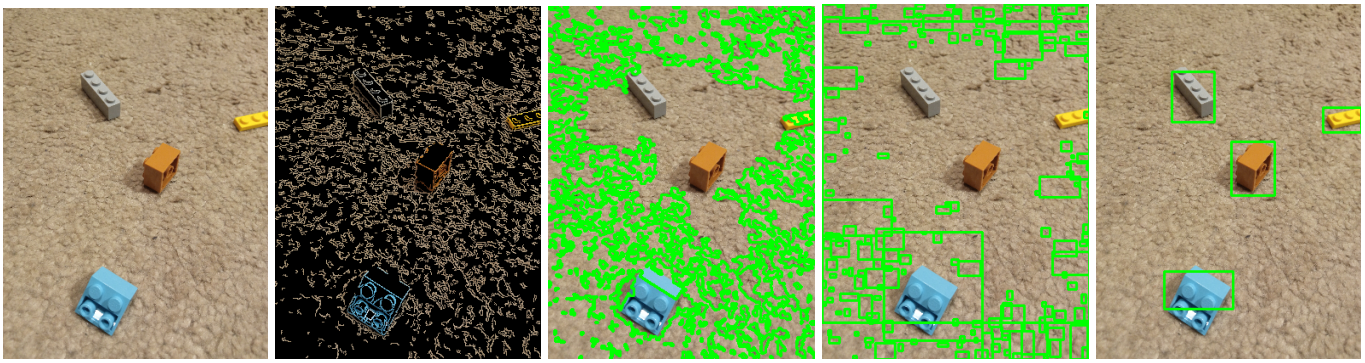
Model Algorithm

1. Gaussian Blur (small)
2. K-means and Canny Edge Detection
    a. Multiple iterations of K-means with different values/seeds
    b. Apply Canny Edge Detection Algorithm for each one of them
    c. Taking the edges for each iteration and choosing the overlapping ones according to a certain threshold.
3. Contours
    a. Get contours based on the edges
    b. Connect nearby edges with a kernel
4. Bounding Boxes
    a. From the previous step, the algorithm tries to find the respective bounding boxes
    b. If there are too many of them, it applies a bigger blur and restarts from step 2.
    c. Filters some overlapping and small bounding boxes
5. Color Detection
    a. Take an histogram of the most common range of colors of the original image (the most frequent one must be the background color range)
    b. For each bounding box found earlier in step 4:
        i. Removes the background color range
        ii. The remaining colors must belong to the lego pieces. Take the most frequent one and store it in a colors list.
    c. Compare each pair of colors in the colors list using manhattan distance. It groups up the colors if they are too close to each other (hence the same color hue), ensuring only different lego colors are kept in the colors list.

Results

The following images represent the multiple stages of the program running. In the second sequence, there is one more image, due to the extra iteration with a larger blur.

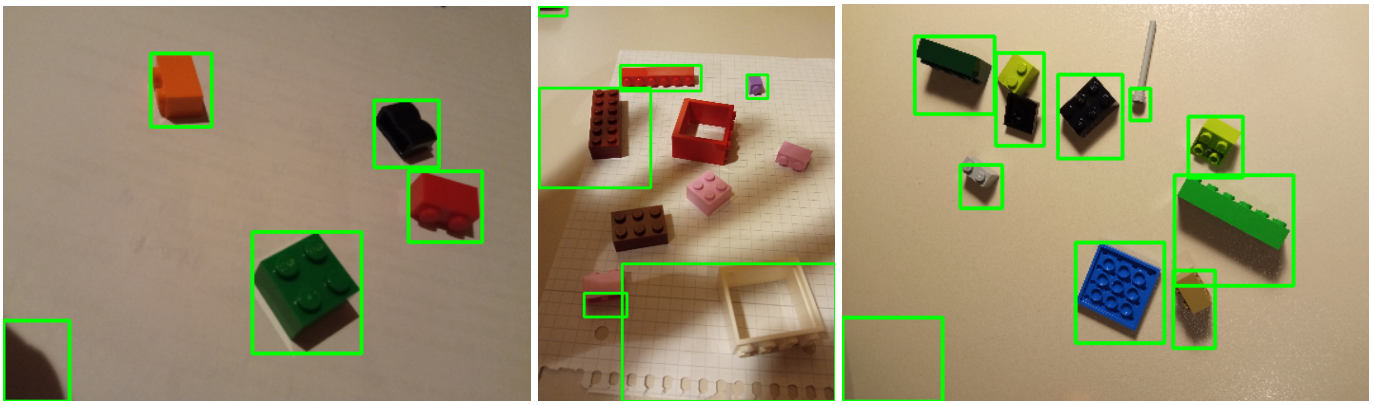




Known Problems

The model used has a couple problems:

- The results take a while (40s) to compute. This happens mainly due to the K-Means (Step 1) and Color Detection (Step 5).
- There are some detection problems related with false positives in shadows and corners or others where the Canny edges overlap with the legos and huge bounding boxes are defined.
- Some light colors (gray, pink) or that are very similar to background (gray, white, transparent) are sometimes not found as pieces.
- Pieces that are too close are sometimes considered a single piece (like in image 3) due to shadows



Group

João Alves - 202007614 Marco André - 202004891 Rúben Monteiro - 202006478