Task 1 Report

Computer Vision

up202008462@fe.up.pt José Luís Cunha Rodrigues
up202008866@fe.up.pt Guilherme Cunha Almeida
up202005097@fe.up.pt Pedro Jorge da Rocha Balazeiro

April 8, 2024

1 Introduction

Object detection is one of the various applications of computer vision. The system presented in this report aims to detect the presence of LEGO pieces in images and characterize their position and color.

2 Approach

This section outlines, in an ordered fashion, all the steps followed in the main pipeline to identify the objects.

2.1 Preprocessing

Before delving into the actual processing of the images, it is necessary to first go over some adjustments. From the samples in the dataset, there are a lot of different characteristics that may impact the way the image is processed. This includes shadows, noise, and different backgrounds such as quad-ruled paper.

As a first step, the image was resized to 500x500. The reason being this could ensure uniformity across samples and reduce computational costs, without losing too much information.

Moreover, a median filter was also applied. This method was chosen over Gaussian filtering because even though the images had high quality, some random dust spots were being identified as LEGO otherwise. The chosen method also did a better job of preserving edges, which was relevant to this problem.

2.2 Finding Contours

The next step in the pipeline is to try and identify relevant contours in the image, and thus the desired objects. To start this process, edges were identified with the Canny Edge Filter. Then, the latter were dilated, so as to make them thicker, thus connecting the edges that weren't connected. This dilation process helps in closing small gaps between the edges resulting in contours with more cohesive shapes.

This process will yield several contours per brick. Thus, there is a need to filter them, to end up with one per piece. This was done by removing cases where the bounding box was fully contained.

2.3 Detecting colors

In the earlier stages of development, the full image was considered for color identification. The main idea behind this strategy was to aid the detection of the bricks themselves by analyzing the colors. The first method tried was to retrieve the image's histogram and then count the relevant peaks. Following the same rationale, clustering was performed to try and group the image per piece and distinguish the background. The algorithms used for this task were K-means and DBScan. For K-means, the elbow and silhouette methods were used to determine the number of clusters. Both methods were

unsuccessful in doing so as the predominance of the background cluster skewed the results. This was a challenging problem as the clustering algorithms were sensitive to shadows and other changes in color.

It soon became evident that it would be easier to distinguish colors from the already identified contours. To do so, one first had to determine some sort of dominant color for each detection. This was done by simply obtaining the arithmetic mean of the region. The mode could also be used for this metric, even though the results obtained were slightly worse, given the continuous nature of color values.

The problem that follows is the discerning of similar colors. The metric used to estimate color similarity was the ratio between the Euclidean distance and the respective maximum. Using RGB had its limitations, since the obtained distance may not be a good representation of what color similarity means to a human eye. Therefore, different methods of computing this metric were tested, including using RGB Luminance, HSV, and CIELAB color space. The latter was the strategy included in the main pipeline, as it provided the best-performing results.

3 Challenges

Although the steps described in section 2 work for the generality of samples, there is a selected subset in which special attention was required, as described below.

3.1 Shadows

The presence of shadows in the images posed a significant challenge to accurately identifying LEGO pieces. These induce the program to include the dark region in the contour itself.

Initially, applying a threshold to eliminate the darker regions associated with the shadows was tried. However, this approach proved to be ineffective as some pieces were inherently darker than the shadows, leading to the unintentional removal of legitimate pieces.

Another strategy involved using an equalizer to adjust the overall brightness and contrast of the images, aiming to reduce the prominence of shadows. Yet, the results obtained from this method were unsatisfactory, as the shadows still interfered with the contour detection process.

In an alternative approach, rather than attempting to remove the shadows from the original image, an effort was made to reshape the contours after their initial detection. This involved post-processing steps aimed at isolating and removing the shadow sections from the contours. Unfortunately, this approach also proved to be impractical, as distinguishing the LEGO pieces from the shadows turned out to be a challenging task because no good solution was found to separate the shadow section from the piece section of the contour.

In the end, shadows were not fully resolved since no approach was able to remove them completely from the final contours.

3.2 Connected pieces

There are some pieces in the samples that appear touching each other. This induces the program to wrongly identify them as a single object. Even though the median filter described above can improve overall results, it can aggravate this problem by smudging the two pieces together.

One of the techniques employed to try to solve this problem was morphologic operations. The image was binarized by either thresholding or by using previously detected contours. Then, erosion was used to try and separate the pieces. This did not produce great results, even for unblurred images, as the strength of erosion needed to impact the connection was too aggressive for other pieces. One reason for this could be the existence of shadows between touching bricks. One of the operations experimented with was the black hat operation, aiming to highlight dark structures against a lighter background. However, the outcomes fell short of the expectations. One significant challenge arose from the similarity in intensity values between certain pieces and the background. This similarity led to the misidentification of pieces as part of the background, resulting in confusion and inaccuracies in the detection process. Additionally, in images with varying illumination and shadows, the identification of pieces became exceedingly difficult. Moreover, the black hat operation failed to effectively differentiate connected pieces, further exacerbating the inaccuracies in our results. Similarly, top hat operation was explored. It is tailored to enhance bright structures against a darker background. Unfortunately, this

operation also yielded unsatisfactory results. The primary reason for this was the lack of a significantly darker background in the images. The background exhibited a brightness level comparable to that of the pieces, making it challenging for the operation to enhance bright features effectively. Furthermore, images with intense lighting in certain regions led to incorrect identification of bricks due to light reflections, while pieces in darker shadows remained undetected.

3.3 Corners

Some samples in the dataset contain corner sections that could be mistaken for bricks. These instances include table edges or the transition between a paper sheet and the table surface.

Regarding the methods employed, the authors encountered challenges in accurately distinguishing between these scenarios and actual pieces situated in the corners. As a result, due to the predominance of non-brick scenarios, corner contours were uniformly excluded from further analysis.

4 Results

The program's quality was accessed by running it against the samples and calculating the relative error for the number of detections and colors on each sample. The results obtained for the entirety of the dataset are presented in Table 1. These metrics were used to assess and improve the program's quality empirically. Moreover, some image results can be found in Annex B.

Table 1: Overall results for the program

	Average Relative Error	Total
Detections	94.46%	(280/298)
Colors	90.40%	(243/248)

5 Conclusion

The developed system shows promise in identifying LEGO pieces in images, despite encountering various challenges. However, optimizing the program to perform better for specific images may impact its performance overall. Thus, addressing these challenges without compromising overall performance remains a complex task. Nevertheless, the achieved results meet our objectives.

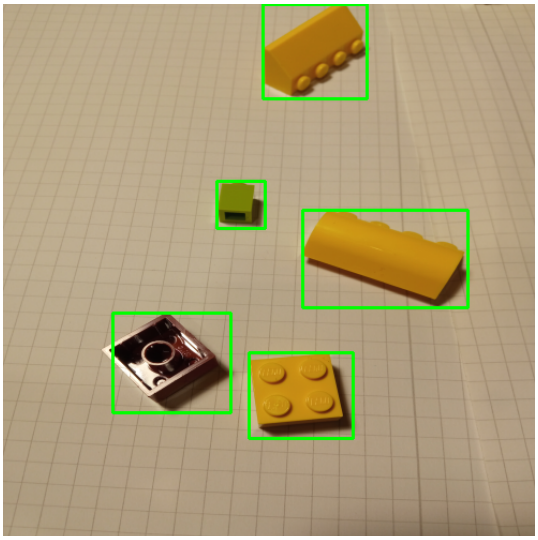
A Program Instructions

The described program is included in a single Python file and can be executed by running it as:

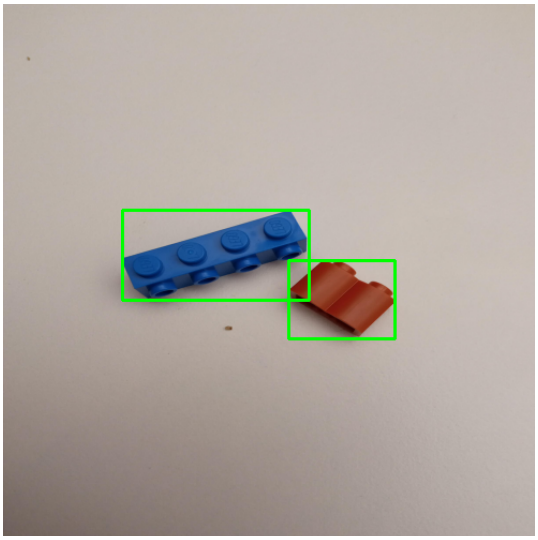
```
python main.py <input_file> [<samples_dir>]

# Example
python main.py input.json samples
```

B Image Results

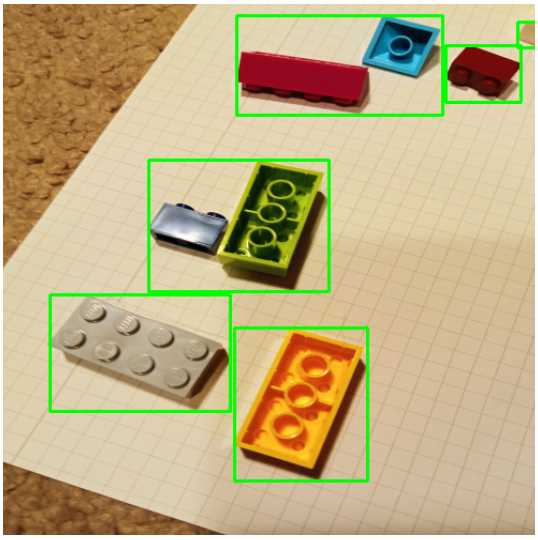


(a) Quad-paper background

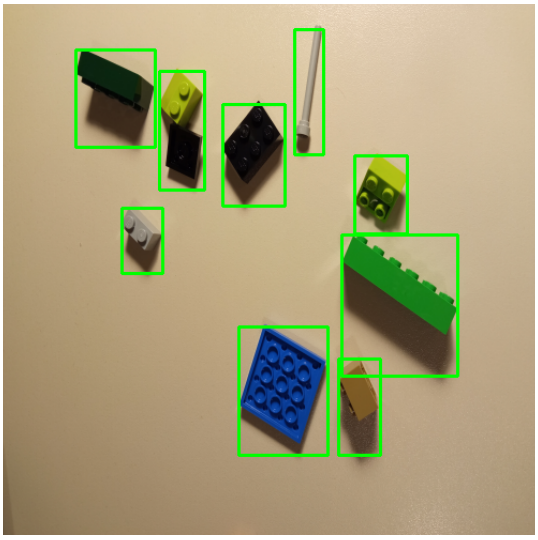


(b) Dust Spots

Figure 1: Background and Noise Resilience



(a) Corners and connected pieces



(b) Shadow detection

Figure 2: Illustration of different challenges