Assignment 4 – Pokémon Analysis

For this assignment, I relied heavily on connected component labeling and template matching to decipher text. This took care of identifying HP, CP, and stardust. For the big semicircle and level location, I relied on Hough Circle Transform to detect them. Lastly, to determine the Pokémon’s ID, I did RGB histogram comparison. There is some preprocessing before these steps to reduce the search space, such as cropping a patch from the original image where the objects of interest are likely to be and removing values of the wrong colors. The cropping part was driven by manual tuning and validation through my own sight. My template matching was extremely simple. All I did is use a few samples of known characters as binary images and subtract them from the binary images created from putting a bounding box around the labeled connected components.

**HP/CP/Stardust**

When finding HP, I knew that three important characters to find are ‘H’, ‘P’, and ‘/’. My goal was to find these three characters occurring on similar rows of the image. Afterwards, I discarded all connected components matched with non-digit characters. Then, I only kept the ones to the right of the slash, since we are supposed to be finding max HP rather than current HP. The CP values were found with similar heuristics to HP, focusing on locating ‘C’ and ‘P’ characters on similar rows and then only examining close digits to the right. Stardust is an interesting story because the correct value is often surrounded by false positive text. To get around this, I realized that stardust values will always have at least two zeros. Once I find these zeros, I just look for nearby digits on the same row that are also within a reasonable distance between columns.

**Circle Location/Level**

Finding the level circle location is simply done through MATLAB’s built-in ‘imfindcircles’ function where the radius range I am searching through is based off image proportions that are manually tuned. I had an issue with the eyes of the Pokémon being detected, so I whited out most of the area beneath the level arc to prevent these false positives. The bigger semicircle was not being detected by that technique with an adjusted radius range, and my own implementation of Hough Circle Transform was slow and not as accurate as guessing, so the result I return for this value is always horizontally centered and 35% of the way down from the top of the image. It’s a shame I couldn’t get this to work, but the guess is usually right on the target since it is very predictable.

**ID**

Finally, to determine the ID, I used RGB histogram comparison with 32 bins across 3 channels. This concept is straight from my work in the third assignment, where I used this technique along with the bag of words. My approach here was greedy and did not use k-nearest neighbors to match Pokémon to their ID.

**Results**

My algorithm performs well on high resolution images where numbers are clearly visible and not touching one another. In low resolution images, connected component analysis fails terribly due to broken up characters and those that are touching neighbors. On the validation set, I achieved the following results…

|  |  |
| --- | --- |
| **Metric** | **% Accuracy** |
| ID | 87.94 |
| HP | 87.23 |
| CP | 76.60 |
| Stardust | 74.47 |

**Circle Detection Results Sample**

The Good:



The Not Good:

