Spotting Differences Between Affine Invarient Images and Counting the Number of Object Differences

Wuyeh Jobe
Carnegie Mellon University Africa
Kigali Innovation City, Kigali, Rwanda
jwuyeh@andrew.cmu.edu

Abstract

Spotting differences between images have wide ranging applications including fraud detection in various facets of human activity. However, this is not trivial especially when the images are distorted. In this paper, we discuss some of the ideas that were tried and what finally gave promising results. We also discussed the challenges encountered and the limitations of the solution.

1. Introduction

Spotting differences between two images is an immense challenge, especially if the images are distorted. This is because there may be fine grain differences that arise due to changes in lighting, color, and other external factors that create noise. Such noise makes it hard to distinguished between what is important or not. In order to solve this problem, the images were aligned and then template matching was used to find the difference, instead of the traditional way of subtracting the pixels. For this task, the main challenge faced is getting the best Homography matrix that will produce the least amount of noise. This is because any slight change in rotation can cause huge amount of noise that may be impossible to separate from actual differences. The Oriented FAST and Rotated BRIEF(ORB) is used for feature detection.

2. Approach

The approach for solving this problem includes aligning the images, using template matching to find the differences and finally using contours to count the number differences between the images.

2.1. Aligning the Images

To align the images, key points on both images are detected using ORB. ORB, unlike SIFT and SURF is open source and does not require a license to use. Additionally, it is fairly accurate and fast, hence widely used. We need this key points (at least four of them) to generate the Homography matrix. Once the key features are detected in both images, the images are matched together. The features detected are sorted by goodness of match and a percentage of the best matches is selected to find the Homography matrix. After the Homography matrix is calculated, it is used to align the target image.

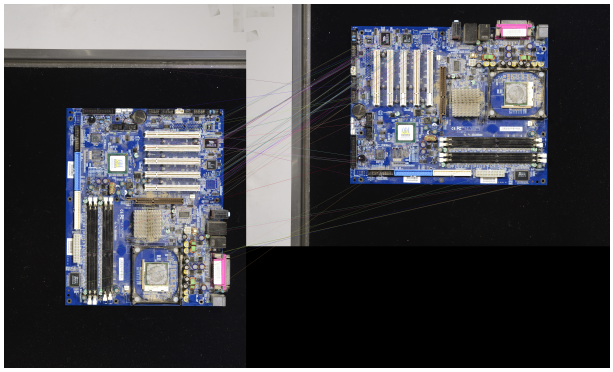

Figure 1. Showing strong matches between two Images

2.2. Finding the difference between the images

To find the difference between the aligned images, we loop through the images using a window. For each window on first image, we correlate it with the window on the second image using template matching. The OpenCV template matching function returns a value (for matrices that are of the same size) based on the metric used. The window size is determined through a trial and error bases and does not exceed a width and height of 5 to avoid stitching objects together. For this purpose, the cross correlation measure is used so that we get the brightest set of pixels in the image that are the same shape to the template. Hence, a large correlation score is returned (mostly over 0.9) when there is very little difference between the two images for a partic-

ular window. Initially, a matrix with zeros is created (let's call it the mask). Whenever, the cross correlation score is less than a certain threshold (meaning a strong match is not found and there is a huge difference between the two at that particular window), the corresponding location on the mask is shaded white. By the time the loop is done, all the places that have difference are highlighted. This approach prevents using dilation which makes it possible to avoid objects being stitched together for unique use cases.

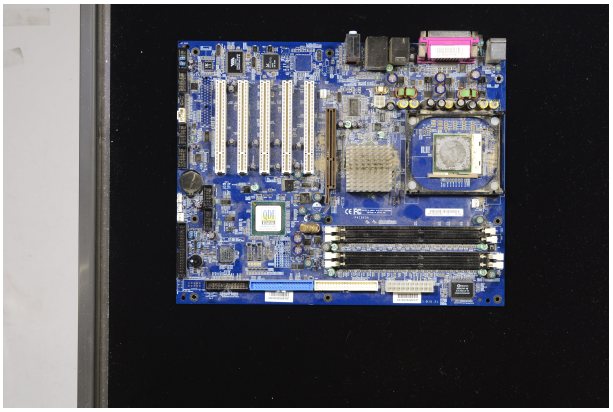

Figure 2. Image A

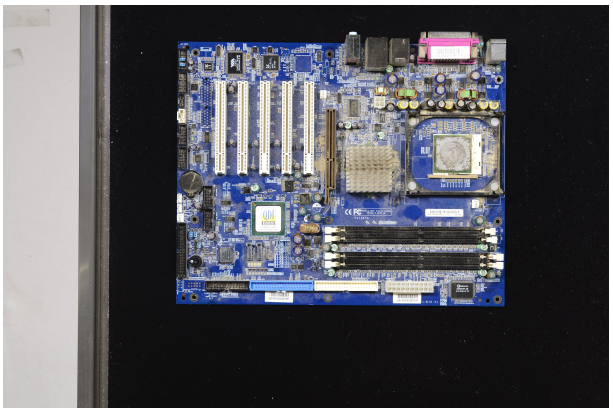

Figure 3. Image B

2.3. The Real Challenge - Find the number of objects

The most important part of the work happens after this mask is generated. Since a certain threshold is used, the pixels representing a missing object can be divided into small components (essentially creating abject with holes). This makes it hard to find the count of objects using contours since the small components are counted separately.

One of the approaches tried is to use convex haul to connect the small components together. However, the implementation of this is based on contours which does not combine the small components. The convex haul is effective when you want to combine all the components in an image.

Figure 4. Difference using template matching

Another technique tried is to find the average contour area and remove all contours with an area less than the average. Although this significantly removes unnecessary noise, it is not effective since it may remove essential objects.

Another approached tried is the watershed technique. The idea of this is to apply some filtering (e.g. laplacian filtering) to sharpen the image, then apply a distance transform function, and then a watershed method to isolate objects in the image from background.It was not really clear why it did not work. However, we suspect that this is effective if the space between the objects are really close, since most of the applications online are in this direction.

Figure 5. Mask on image

The final approached tried is to play around with the cross correlation threshold and also the size of the window. This was done in a trial and error bases and tested with the test images. However, this has limitations in that it may not be robust for certain use cases. An improvement to this

will be finding the characteristics of each image to be able to come up with threshold and a window size that suits a particular image.

3. Conclusion

Finding the differences between two images, especially if they are distorted is not a trivial challenge. In this paper, we explained some of the approaches tried and the challenges.