Assignment 1: PCB Defect Detection

Visual Computing

During the production of printed circuit boards a number of things can go wrong. This can introduce defects in the boards, leading to malfunctioning PCBs. In this project you will create a program than can find if there is an error in a PCB and where on the board the defect is located.
In the files attached to this project you will find a `template_images` folder. This folder contains images of 12 correctly manufactured PCBs. There are a set of `test` folders which include images of PCBs with defects and degradations. These test images are named {`pcb_id`}_{`fault_type`}_{`fault_number`}`.jpg`, {`pcb_id`} can be used to identify the correct template image.
The goal of this project is to use Structural Similarity to compare the faulty PCBs to the correct ones to find the differences between them and to thus find the errors in the faulty one. Structural Similarity is a perceptually based metric used to compute the difference between two images. Create a `results` folder in the `test` folders and save for each input image a results image where the located defects are indicated by drawing their bounding box, as shown in Figure 1. To compute the Structural Similarity you can use the function implemented in skimage.

1 Noise removal (based on curriculum week 3)

In the first part of the project you will only be comparing images of PCBs that have the exact same pose. So a pixel in the template image depicts the exact same part in the test image. You can thus compute the SSIM without needing to transform the image. This process will be relatively straightforward for `test_1`, the input images of the three folders `test_2`, `test_3` and `test_4` contain different type of noise however. This noise will impact the Structural Similarity computation and will prevent it from localising defects. Analyse the input images and try to determine what type of noise is applied to the image. Remove the noise by applying the appropriate filter for that type of noise. Then try to localize the errors again with the structural similarity.
Filtering the noisy image can remove the noise but will also introduce some degradation which will cause the SSIM to fail. This problem can be avoided by degrading the template image by the same amount as the test image, making them depict the same content again.

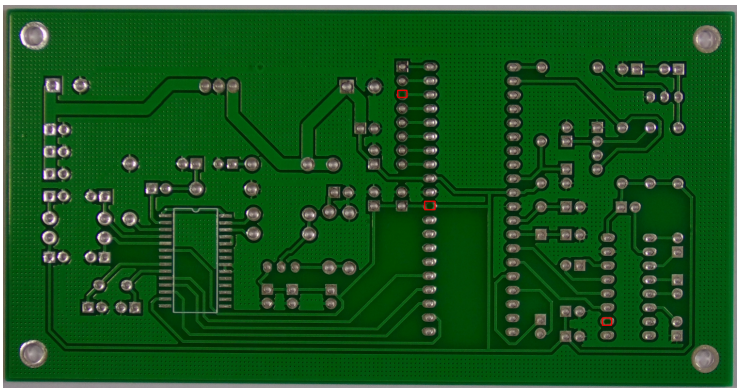

Figure 1: PCB with some errors highlighted.

2 Image Alignment: Feature matching (based on curriculum week 5)

In reality, the pose of the template images and the images of the defects will not be perfectly aligned. For example, the images can be rotated or perspectively transformed. Examples are given in the `test_5` folder. Both images must be first aligned in order for the SSIM method to work. The goal of this step is to detect and match corresponding features between both images. Implement the following steps:

1. Experiment with different feature detectors (SIFT, SURF, ORB, AKAZE, $\cdots$).
2. Implement (using OpenCV) brute force, K-nearest neighbor and FLANN feature matching.
3. Identify and filter only good matches (distance smaller than certain threshold).
4. Make a visualization of the matching features.

3 Image Alignment: Homography (based on curriculum week 6)

The last step is to calculate the transformation between both images and warp the image to let them align. Implement the following steps:

1. Use these good matches of the previous step to construct a homography between the two images.

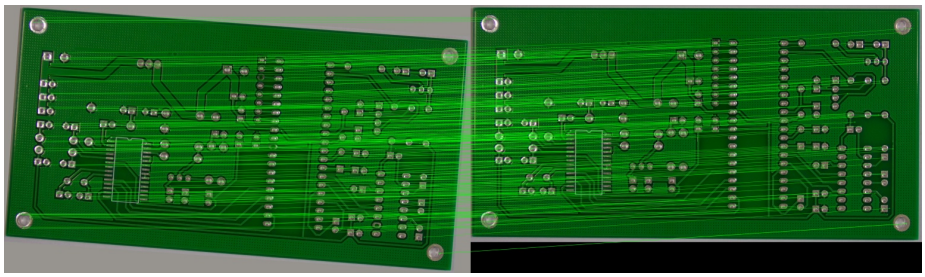

Figure 2: Feature detection and matching.

2. Identify outliers using the found homography.
3. Use the homography to warp and align the image with the template image.
4. Play around with the different interpolation methods.
5. Additional test images with both variations in orientation and noise are given in `test_6`.

Reporting

- In each test folder you should create a `output` folder containing the input images with the detected defects highlighted as shown in Figure 1. You should also create a `processed` folder, this folder contains the test images after you have processed them before performing the SSIM computation.
- **Write a brief report detailing how you approached this project.** The report should be one to two pages and should explain what computer vision techniques you used and why you used them.
- Your program code, the outputs and the brief report should be **committed and pushed to the GitHub repository of the assignment before Sunday 30th of March at 18h00**.