**SAIPS Home Exercise - Defects Detection**

**Nadav Carmel**

**July 2, 2021**

1. **Executive summary:**

A defects detection approach and its results are covered here.

While performance is not satisfactory on this dataset given the implemented approach, a few

improvements are suggested (I had no time to implement them as well), which potentially can gain

more impressive results.

Also, I hope this report can be an opportunity for the company to get the impression on my technical

and analytical skills.

Code is attached and can also be cloned from: <https://github.com/NadavCarmel/defects_detection>

1. **Introduction:**

We have pairs of images (inspected , reference ), where two pairs containing some defects and one pair does not.

Each pair contains images which are highly similar, except of:

* Small relative translation
* The defects that exist in one but (assumed) not in the other.

We are also given with 'labels': the (x, y) coordinates of the (center I assume?) of each defect.

We want to construct an algo which gets such pair of images as an input and return the binary images of the detected defects in the inspected image.

Practically for this exercise, this means 2 main things:

* Align the reference image such that it's as matched as possible to the inspected one.
* Subtract the aligned reference image from the inspected image, and by that, transform the problem into an (weekly supervised) anomaly detection problem, where each pixel is either anomalous (of a defect) or not.

1. **Methodology:**
   1. **Step 1 – Image Alignment:**

There are many methods for image alignment with available implementations on opencv etc.

If our images had also relative scale / rotation to calibrate, we could benefit from (for example) landmark-based approaches (SIFT etc. for finding the landmarks with following RANSAC for matching the images).

In our relatively simple case, I chose to use 2 approaches ‘out of the box’, and implement one of my own for this alignment step:

* + - 1. **Phase-Correlate**: this opencv implementation maximizes the correlation between the 2 images, by transforming them into Fourier-space. The idea is that:

- the multiplication in Fourier space is equivalent for convolution in the original space

- the Fourier transform of an inverted function (f(-x)) is the complex conjugate of the Fourier of the function

- a convolution of f(x) with g(x) is the correlation of f(x) with g(-x)

Thus, by applying the below formula, we get the correlation between the 2 images:

And by looking for the *argmax* of this resulting array we can estimate the relative translation.

* + - 1. **Enhanced correlation coefficient maximization (EEC)**: this opencv implementation (*Georgios et al.*) basically should be very similar to the Phase-Correlate method. It minimizes the L2-norm of the normalized inspected and reference images, thus maximizing the correlation:

But I tried it anyway and got slightly better results than the Phase-Correlate, due to the additional preprocessing of the images (gaussian blurring).

The displayed results are based on this alignment.

* + - 1. **Convolutional filter matching**: this approach, which I thought of and implemented in the code (see: ./src/align\_by\_convolutional\_filter.py), works by running a k-by-k convolutional filter, centered around pixel , on the reference image, and minimizing the squared distance to the corresponding pixel in the inspected image.

Say the reference image is translated by +2 pixels in x direction and +3 in y direction w.r.t. the inspected image, the learned filter will be all zeros except of ‘1’ -2 pixels in x direction and -3 pixels in y direction from its center.

In practice, this is formulated as a liner system of equations (#strides equations, k^2 variables), and is solved straightforward (using a pseudo-inversion).

- The downside is that we cannot calibrate a translation larger than k/2.

If translation is too significant, we’ll be getting too many degrees of freedom in our system of equations and probably ill-conditioned matrix.

I’ll cover a possible improvement (which was not implemented) for this issue in Recommendations section.

* + 1. **Step 2 – Anomaly Detection:**

Now, that we have our reference images all aligned (to a reasonable degree), comes the second part of classifying pixels.

The idea I had implemented here was to subtract the inspected image from the reference image and get a new ‘error’ image, from which classification can arise.

Intuitively, a high-valued pixel in this new image can imply a defect (since the 2 original images are assumed to be very similar aside of the defects), but we need to somehow use our labels and optimize classification threshold.

By this representation, the problem is transformed into a weekly supervised anomaly detection problem.

The approach I took to find the optimal threshold using the ‘error’ image was:

1. Calculate the mean and variance of the non-defected pixels
2. Calculate the mean and variance of the defected pixels
3. Assume each pixel is drawn from a normal distribution with either defected / non-defected distribution params.
4. Repeat those steps over the 2nd pair of images (‘case 2’)
5. Average those 4 statistics () over the 2 sets.
6. On inference (say ‘case 3’ pair), construct the ‘error’ matrix, and compare the likelihood of the defected and non-defected distributions per each of the pixels.

See: ./src/estimate\_defects\_model.py and ./src/inference.py for the implementation.

* 1. **Results:**

In this section I let the images speak for themselves, and do not get into the commonly used metrics (TPR, FPR, Accuracy, precision etc.).

The main reason is that these metrics will not be accurate, as the defects are not properly labeled (not all pixels of the defects are marked in the given data, and the number of defected pixels is in-fact much larger).

As a bottom line, it seems the results are not good enough, with many pixels (especially on some patterns-line-edges) detected as false-positives, and some defected pixels (especially on small defects) which are not detected.

I will cover some suggested ideas to enhance performance in the Recommendations section, which I had no time to implement.

Per each case, there are 6 images attached:

1. inspected image
2. reference image
3. shifted reference image
4. diff between the inspected and the shifted reference images (‘error’ image)
5. probability of each pixel to be considered as defect according to the model (P\_defects)
6. defects prediction binary image (prediction\_mask)

* **Case 1 images:**
* **Case 2 images:**
* **Case 3 images:**

1. Displayed below are 6 images per each of the 3 cases:
2. the inspected image
3. the reference image
4. the shifted reference image
5. the diff between the inspected and the shifted reference images (err)
6. the probability of each pixel to be considered as defect according to the model (P\_defects)
7. the defects prediction binary image (prediction\_mask)

# Example format for a technical report on WQM

# TITLE

1. The title should clearly and briefly indicate what the report is about.

## Executive Summary

1. Summarises the report in plain English. The Executive Summary should be able to be read and understood independently of the main body of the report. It is often easiest to write after completing the rest of the report.
2. **1: Introduction**
3. Background to the report, including discussion of previous studies in the area or related studies. The introduction should also clearly outline the reason for the study/report, including objectives and any working hypotheses.
4. **2: Methodology**
   * 1. **2.1. Data analysis**
5. An outline of how the data obtained from sampling and laboratory analysis were analysed, including details of any statistical tests undertaken and their assumptions and limitations.
   * 1. **2.2. Modeling flow**
     2. bla bla..
6. **3: Results**
7. A description of the results obtained without any explanation or interpretation of them. Visual aids such as graphs, tables and maps may be used to summarise the results. If raw data is to be included in the report, it should be placed in an appendix rather than in the results section.
8. **4: Discussion**
9. The results of the study are interpreted and implications of these results in terms of the project objectives are discussed. A discussion of how the results relate to other studies is also common. An evaluation of the methodology used may also be undertaken.
10. **5: Conclusions**
11. Summarises specific conclusions drawn from the results in terms of the project objectives and working hypotheses.
12. **6: Recommendations**
13. Recommendations as to future management actions and/or studies are made based on the findings of the study.

# References

1. Lists the literature cited in the report.

# Appendices

1. May contain detailed information such as data tables, laboratory reports or photographs.