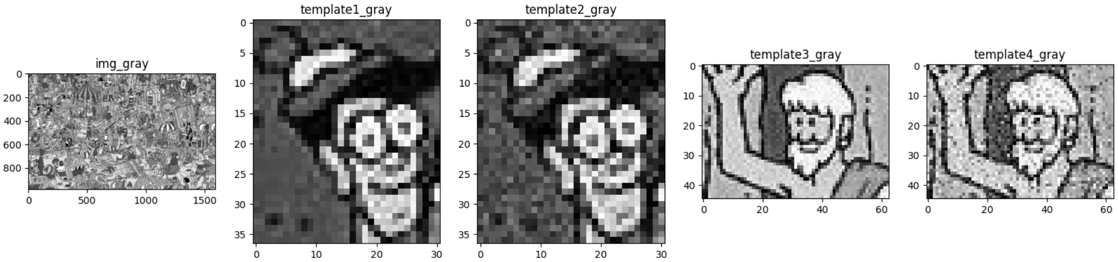
tp

October 19, 2023

1 Practical course Computer Vision

- Clabault Tom
- Magnin Constantin
- Ruivo Bastien
- Spataro Mathis



2 Pattern matching

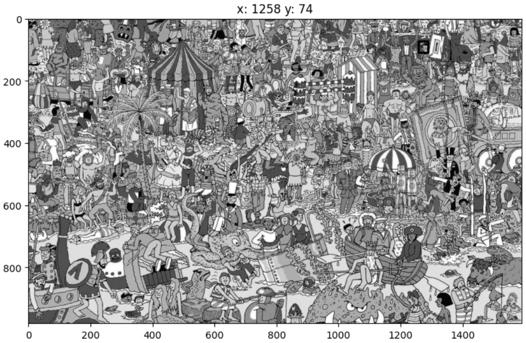
2.1 What did we do

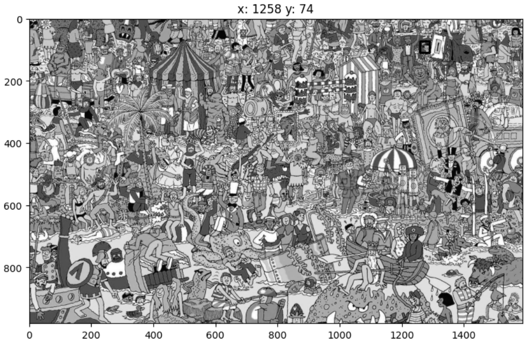
We used the waldo dataset to find waldo in a picture, for this we used the following steps: - iterate over the image - for each pixel, get a region of interest (ROI) of waldo template size - compute the corresponding metric between the ROI and the template and store it - return the location of the ROI with the lowest metric

At first, we tried the sum of squared distances, but it didn't work as you can see in the following images because we loaded the images with matplotlib, which altered the image too much to be recognized by SSD, the problem was fixed in this notebook, that's what we tried multiple metrics, see below.

2.1.1 Sum of squared distances

It is defined by $SSD(x,y) = \sum(roi_{xy} - template)^2$ - Where ROI stands for ``Region of Interest'' at x y, which is a region the size of our template taken in our image. - Template is the image we try to find in the target image.

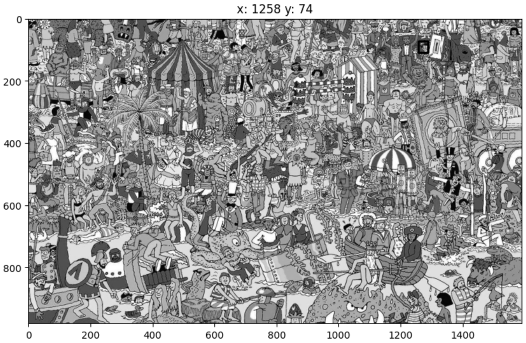


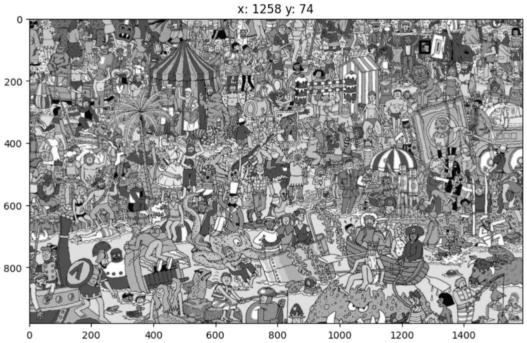


- It's able to find waldo in this case, so it's a good metric to use for pattern matching images that are fix.
- But it is important to note that SSD is sensitive to change in luminance and size of the image, that is why we got problem when we load the image with matplotlib. The image was modified and we were unable to find the pattern, unlike with NCC and ZMC, but in exchange it's less computing expansive than NCC and ZMC.

2.1.2 Normalized cross correlation

It is defined by $NCC(x,y) = \frac{\sum(ROI - average(ROI)) \times (template - average(template))}{\sqrt{\sum(ROI - average(ROI))^2 \times \sum(template - average(template))^2}}$

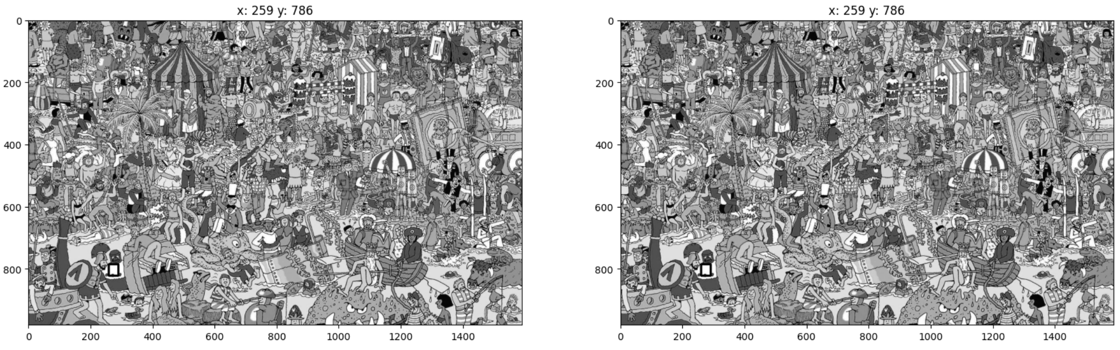




- It's able to find waldo, with and without noises so it's a good metric to use but it's painfully slow as we can imagine due to all the division, this is a solution to keep in mind. But in simple case, it's easier to start with SSD. And if it's missing the template too many times, we can try this metric.

2.1.3 Correlation

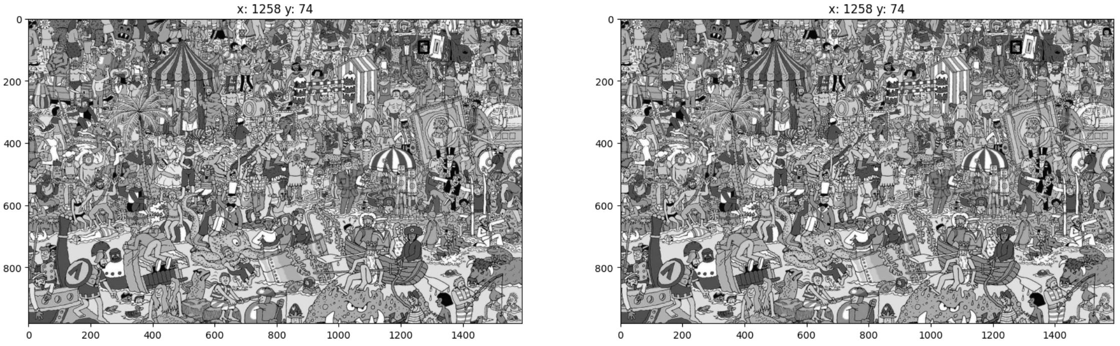
It is defined as follow $C(x,y) = \sum ROI \times Template$



- It's not able to find waldo, it's too sensitive to the background / and noise that we may find in the image but it's faster than the normalized cross correlation and SSD. However, and as we can see in the formula if we interpreted it correctly from the course, the more higher the value of the ROI is, the more higher it will be and that's why the program find a white roi as the best candidate. I don't know if this metric is really suitable for founding a pattern.

2.1.4 Zero mean correlation

It is defined as follow $ZMC = \sum(ROI - average(ROI)) \times Template$



- It's able to find waldo, with and without noises so it's a good metric to use and it's faster than the normalized cross correlation, however it is known to make false detection in other case and it is slower than SSD.

2.2 Benchmark

```
-----------------------
------ Grayscale ------
-----------------------
Time for find_pattern_Corr: 5.420857424900168s per iteration
Time for find_pattern_ZeroMeanCorr: 15.267262754500189s per iteration
Time for find_pattern_NormCrossCorr: 33.68204152429971s per iteration
Time for find_pattern_SumSquaredDist: 7.548036304599736s per iteration
```

```
-----------------------
--------- RGB ---------
-----------------------
Time for find_pattern_Corr: 8.567348977800066s per iteration
Time for find_pattern_ZeroMeanCorr: 24.510718908499985s per iteration
Time for find_pattern_NormCrossCorr: 49.98489026099996s per iteration
Time for find_pattern_SumSquaredDist: 10.864691014100027s per iteration
```

3 Benchmarks

Because several methods are able to find Waldo (noisy template or not), the deciding factor here will be the time it takes to compute the operation on the ``Where is Waldo'' image that we're using.

Here are the timings that we measured for the search of the template in the entire image

3.1 Grayscale Waldo template

- Simple Correlation: 13.9s average. Not able to find Waldo
- Zero Mean Correlation: 30.19s average. Able to find Waldo
- Normalized Cross Correlation: 70.99s average. Able to find Waldo
- Sum of squared distances: 16.09s average. Able to find Waldo.

3.2 RGB Waldo template

- Simple Correlation: 17.99s average. Not able to find Waldo
- Zero Mean Correlation: 36.77s average. Able to find Waldo
- Normalized Cross Correlation: 88.94s average. Able to find Waldo
- Sum of squared distances: 19.61s average. Able to find Waldo.

All in all, the best method to find the template in this exact scenario is going to be the Sum of squared distance because it's the fastest method that can find Waldo (the simple correlation is slightly faster but it cannot find the template so it's of no use here).

Using an RGB template instead of a grayscale doesn't result in any significant improvement when it comes to finding Waldo. However, doing the computations in RGB rather than grayscale results in an 20% to 30% slowdown so there's really no point to searching the template in RGB here (even though it could yield better results for other applications).

4 Image transformation

For image transformation, we first took with hand for point on the bus, and then we built the homography matrix.

4.1 Homography matrix construction

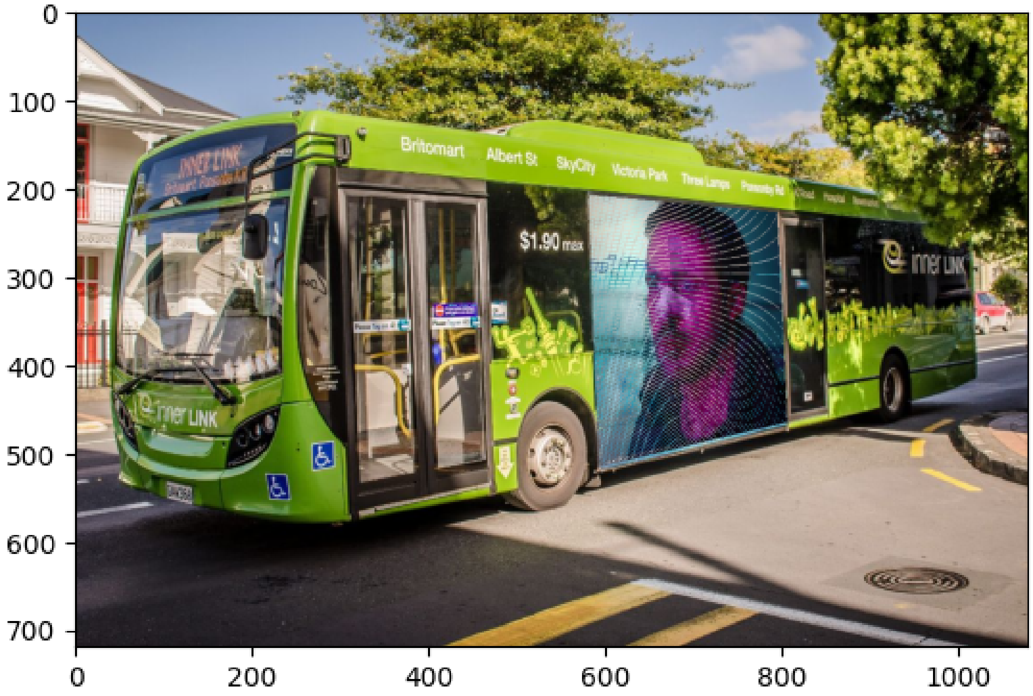
- This matrix consist of finding the transformation applied to the src points to become dst points

- We define the probleme as a linear algebra system based on the matrix equation $Ax = b$
  - We have x, which is the position in the source image (the simpson poster) and b (the position in the bus)
  - The objective is to find A, the matrix to multiply by when we will iterate from pixel to pixel.
    - We build the system with two rows for each point, like
      - $r_i = [x_{si}, y_{si}, 1, 0, 0, 0, -x_{si} \times x_{di}, -y_{si} \times x_{di}]$
      - $r_{i+1} = [0, 0, 0, x_{si}, y_{si}, 1, -x_{si} \times y_{di}, -y_{si} \times y_{di}]$
    - s stands for source and d for destination.

4.2 Warping

There is two types of warping, the first one is iterating on the pixel of the source image, and put them into the destination image (from simpson to bus) It is faster in this case, but depending on which image is bigger, it can be faster to iterate on the source, or on the destination.

It is important to note that pixel does not always have a corresponding 1 to 1 pixel, so we have to interpolate the value of the pixel, which is not done in this case and lead to holes on the image, it should be the next step if we dig further into image transformation.



The second one, is to iterate on the destination image, and find the corresponding pixel in the source image, and put it into the destination image. It requiere to compute the inverse of the homography matrix, and it is slower in this case, but there is no hole in the image.

5 Image Stitching

5.1 Steps

Given several images that were taken with a slightly different point of view (rotation to the left or to the right, in a panorama fashion), the goal is to stich the images together to reconstruct the panorama. We're going to process images two by two (and not all at the same time), stitching the second image to the first, and then the third to the result of the previous step, and then fourth to the result, ... To do that, several steps are needed:

- Firstly we need to find features in the two images that we want to stitch.
- Secondly, we need to match the features of the two images i.e. identity what are the common features of both images and find where these common features are on each of the images
- Thirdly we need to evaluate the transformation needed to go from the set of features of the first image to the set of features of the second image
- Lastly, we need to stitch the two images using the previously found transformation

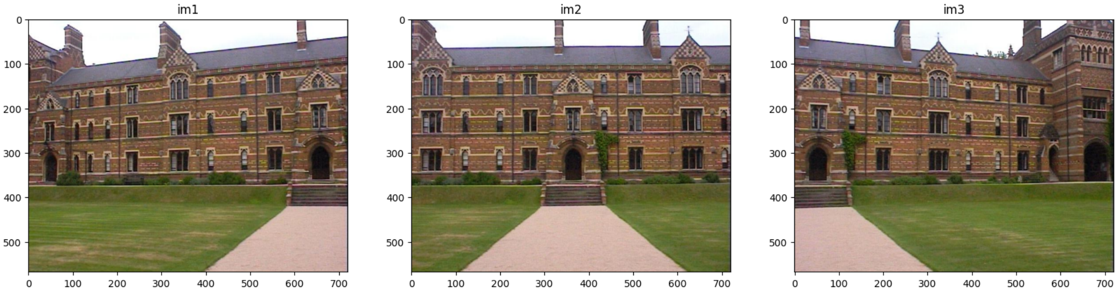
5.2 Feature searching

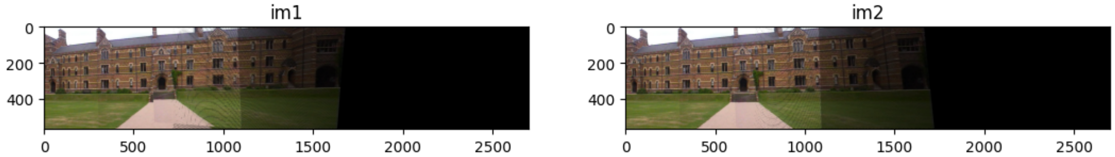
To find features in a given image, multiple methods are available. We've tried both the SIFT and ORB algorithms which are both feature detectors and descriptors.

5.3 Feature matching

Once features have been found on both the images that we want to stitch, we need to find which feature of the first image corresponds to which features on the second image. This is done by computing a distance between the descriptors of the features. The distance function depends on the format of the descriptors so the L2 norm is used to compute the distance between SIFT descriptors whereas the Hamming distance is used to compute the distance between ORB (BRIEF) descriptors.

Doing so, we can find, for each feature descriptor of the first image, the feature descriptor (and hence the feature point) on the second image that best matches it. This gives us pairs of points that will be used to estimate the transformation needed between the two images.





5.4 Transformation (warping)

Once the matching feature points have been found between the two images, we can estimate an homography trasnformation matrix to go from the second image to the first image, ``pasting'' the second image onto the first image, effectively stitching the two images together.

This transformation can be estimated either by choosing 4 points on the image and solving the regular linear system of equation to find the homography matrix or by using an algorithm such as RANSAC. Using RANSAC is the approach that we have chosen. RANSAC functions by selecting, at random several feature points between the two images. It then constructs an homography matrix using only the points selected at the previous step. The transformation matrix thus constructed is then tested against all the other feature points of the images. If points from image 2 are correctly transformed to their equivalent (found during the feature matching step) on the image 1, then this means that the transformation matrix is correct. RANSAC thus proceeds iteratively to find the best transformation matrix between the two set of points, putting aside outliers.

The transformation matrix estimated by RANSAC is then used to warp all the points of the second image onto the first image. This has for effect to make corresponding features meet and this effectively stitches the images.

5.5 Limitations

There are one main limitation to our current implementation: the difference of image sizes after they have been warped. This poses a problem when warping an image results in an image that is larger than it originally was. Indeed, when warping the image enlarges it, the destination space is bigger than the starting space. This results in empty spaces between the pixels of the warped image. Effectively, this creates black(or whatever the background color is) lines in the warped image as can be seen on the figure of the above cell. One solution to this problem could be interpolating the values of adjacent pixels to fill in the blanks but we haven't implemented this solution.