Structured Data : Learning, Prediction, Dependency, Testing

Empirical Study : High dimensional mutual information estimation for image registration

Alexandre BRUCKERT
Elliot HOFMAN
Gregoire MAGENDIE

Contents

I - Introduction

Image registration is a common signal processing problem, whose applications can be used to solve a lot of tasks : automatic registration in medical imaging, movement detection in videos, morphing generation with images, ... Intuitively, the problem can be seen as defining and maximizing a similarity metric between two images. The paper focuses on this approach, proposes a new metric and compares with the other existing strategies.

Traditional methods used for image registration consist in the estimation of the mutual information between the two images. Given some images X and Y, the idea is to compute $I(X,Y) = H(X) + H(Y) - H(X,Y)$, where H is the differential entropy $H(X) = -\int p_X(\xi) \log p_X(\xi) d\xi$. Existing algorithms estimate those quantities with histogram methods, thus implying that the data is first set into bins to estimate the frequency probabilities in the above formula. If those methods work well when working in small dimension (typically $d = 1, 2$), they become quite prohibitive as the dimension of the data grows up, because the number of bins needed to separate the samples increases exponentially. Some solutions have been proposed to tackle this issue, but all of them remain very costly when $d > 2$ ($\mathcal{O}(dN^2)$).

As a result, the mutual information estimation is usually made upon 1-dimensional samples (scalars), pixel gray-scale intensity for example. The intuition of the author of the paper Ḧigh dimensional mutual information estimation for image registrationïs that some performance would be gained if one could incorporate more features in the samples (the three RGB intensities for instance). To be able to efficiently compute the mutual information, the author presents a new entropy estimator, based on the nearest neighbor entropy estimator of Kozachenko and Leonenko. He modifies the algorithm so that the new version is numerically robust and computationally efficient. The asymptotic complexity $\mathcal{O}(dN_{\text{pixel}})$ allows for mutual information estimation with high dimensional data.

A remarkable benefit from this approach is that it makes image registration able to be computed with complex feature maps rather than just 1 or 2 attributes per pixel. The results show that the new entropy estimator is effective in determining the correct alignment even in difficult cases in which the classical criteria fails.

Before we dive deeper into the theory, let's sum up the problem. Formally, given two images F and G and a family of geometrical transformations (such as translations, rotations, affine transformations, ...), the purpose is to find the transformation T such that $T(G)$ is as close as possible to F, according to some image similarity criterion.

II - Details on the implementation

In the associated Python notebook, several kinds of tests/applications of the approach proposed in the paper are implemented. The files estimators.py, raw_functions.py contain the source code for the entropy and mutual information estimators introduced in the paper. We self implemented all the algorithms on top of numpy and scikit-learn, and we also provide an implementation built on the package ITE in the file ite_implementation.py (actually only a wrapper around the KL estimators so that the samples can be treated in a batch-fashion).

The notebook starts with some sanity checks, in which we verify that the proposed estimator works fine for some cases in which the theoretical entropy is known. For instance, if X is a 1D Gaussian variable distributed with the parameters (μ, σ^2), the entropy of X is known to be $H = \log(\sigma\sqrt{2\pi e})$. We also verify that the entropy estimator leads to a correct estimation in the case of d i.i.d. Gaussian variables, in that case the entropies will sum up so that the final entropy of the data is $H = d\log(\sigma\sqrt{2\pi e})$. As real world images pixels are probably not drawn independently from each other we finally verify the behavior of the estimator in the case of two 1D correlated Gaussian variables. We simulate this with a multivariate normal distribution, for which the theoretical entropy is supposed to be $H = \frac{1}{2}\log((2\pi e)^2|\Sigma|)$.

Before we finish with the sanity checks, we also simulates $N = 512 \times 512$ 3-dimensional samples and estimate the entropy with the KLD estimator and the KLBD-M estimators for varying size of batches M. The point is to find a value of M which seems to offer a good trade-off between the estimator's bias $\mathcal{O}(\frac{1}{\sqrt{M}})$ and its speed $\mathcal{O}(dNM)$.

Then the same experiments as in the paper are done. We selected the same mandrill image as in the paper, an image of a snake and an image of a tiger. For each of the three 512×512 RGB images, a modified version is obtained with an editing software. As suggested in the paper, we increase the saturation and brightness of the image, invert the colormap and add some gaussian i.i.d. noise. We then rotate the modified image from -10 to 10 degrees and calculate a similarity criterion with the SSD, MI and CoMI estimators introduced in the paper. A good criterion should have its maximum for a rotation angle 0 and decrease smoothly away from it. Those images were chosen because they have several different color areas, so that the CoMI estimator based on a color feature map should be able to extract some interesting common properties between registered images.

We then reproduced the experiment done in the paper with grayscale images. This time, we try to register a blurred version of Lena with a blurred version of its edges. Edges are detected trough a Sobel filter and images are blurred with a Gaussian filter (this being done with an editing software). Once again, we compute the similarity and

computation times with SSD, MI and NbMI from the paper for each angle between -10 and 10 degrees.

We then introduce a first real world application. We looked for a localization in the Recoleta district of Buenos Aires on Google Maps and took a screenshot of the classical view and the satellite view. We then repeat the rotation experiment on those two images, in order to see if the NbMI estimator is a better criterion than SSD and MI for image registration in that case.

Finally, we compare SSD, MI, CoMI and NbMI on a face recognition task. More precisely we chose 9 persons and for each of them we selected 4 images on the web. The two first images are just two different RGB images of the person, the third one is a caricature and the last one is a black and white photo of the person. Each image is centered around the person's face and reshaped to be 128×128. This leads to 36 images and 666 pairs of images. For a given image, we want the criteria to be able to recognize that the 4 closest images are the 4 different images of the same person. We calculate the score as the number of correctly identified images among the 4 first more similar images, and we then average this score for all images to get the final score. This final score is computed for the SSD, MI, CoMI and NbMI metrics.

III - Difficulties encountered

We self implemented our own version of the estimators during this project. The paper mainly proposes a closed-form formula to estimate the entropies, so there were no real need to use an uncommon third-party library. However our code is built on top of numpy for convenient array manipulation and scikit-learn for efficient nearest neighbors computation (the package automatically detects which of the strategies brute-force, ballTree, KdTree is best depending on the data). Some other native-python functions were used to deal with I/O and basic transformations on images (rotation, reshape, grayscale conversion, ...).

We had some troubles however with the dtypes of the images. For instance if two images F and G are encoded as 'uint8' $5 - 10$ in the corresponding entries would output 250 instead of -5 because numbers are encoded modulo 255. This made us loose some time before we spotted the bottleneck.

We also did not know well how to properly rotate the images. When the image is rotated, some background is added and this has to be cropped to ensure fair results. We spent some time finding a good solution to properly crop the largest rotated rectangle in the image, but it led to bad final results. We finally contacted the author of the paper, who told us that he simply cropped the images by a ratio of 0.08 in each size of the image, so that is the method we used by the end.

An other problem we encountered was the computation time issue. Although the proposed estimator is supposed to be fast, it is still quite slow to run with Python. The author told us that he had implemented it in C and run it on a multiple cores processor. So we spent some time trying to understand why we could not reproduce the same order of computation time, before we finally understood that the problem came from the experiment conditions which were absolutely not the same. We then started to go on on Cython tutorials, so that we would run the slowest parts of the code (mainly the nearest neighbors search) in C. Unfortunately, either scipy and scikit learn already make use of Cython in their nearest neighbors implementations, so there would have been no clear improvement doing this.

Finally, we used the ITE package (in its Python version) to check our results. We had to modify the source code slightly so that the KL estimator could be computed in its classical form or by batches. We finally did not need it for the experiments, because this implementation had the same results as our implementation but was running even slower that our version. We let the code in the file implementation_ite.py anyway.

IV - Analysis of the results

The first thing we did was to check that the estimator gave good estimations in the case of simple signals for which the theoretical entropy is known. The graphs obtained represent the bias as a function of the number of samples N.

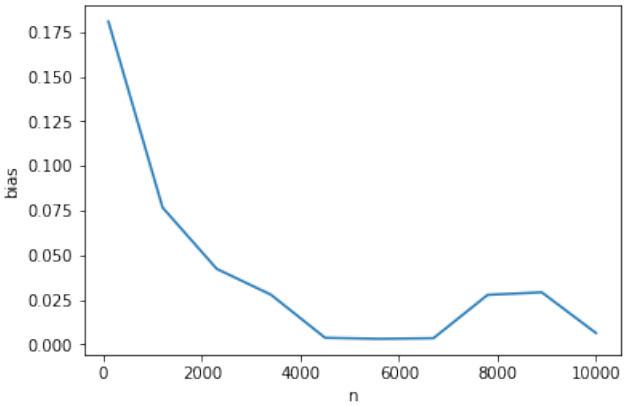


Figure 1: Bias for 1 1D Gaussian variable

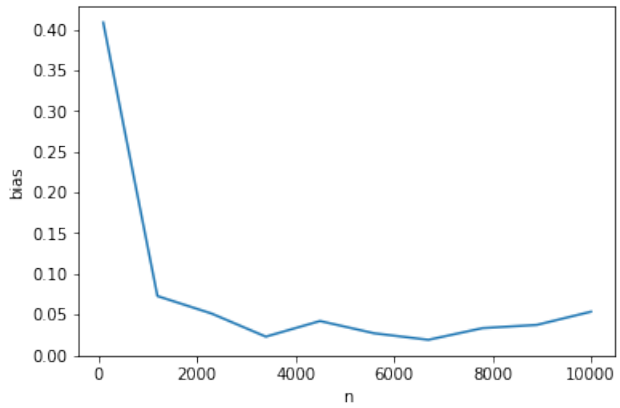


Figure 2: Bias for 10 1D i.i.d. Gaussian variables

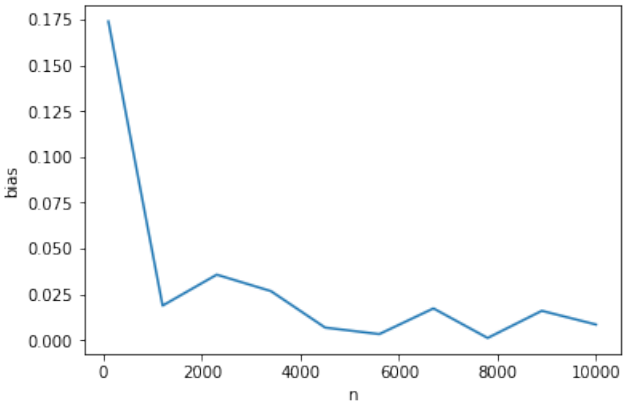


Figure 3: Bias for 2 1D correlated Gaussian variables

In all cases, the estimator proposed in the paper is able to estimate the theoretical entropy with a bias that goes as $\mathcal{O}(\frac{1}{\sqrt{N}})$. The curves may not be strictly decreasing because different samples are generated for each different value of N. It would be more exact to reproduce several times the estimation for each value of N and report the bias/variance, however we simply wanted to make sure the estimator was correctly implemented and able to estimate the entropies as stated in the paper.

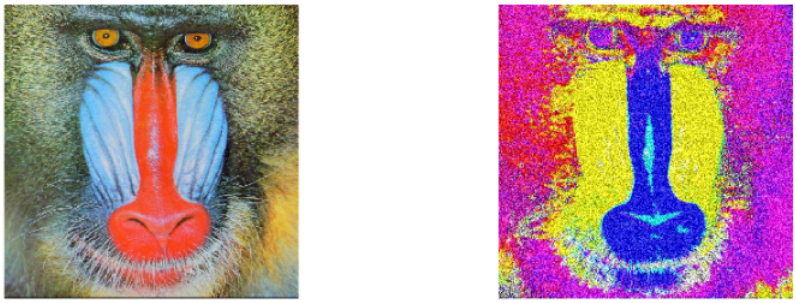
As the main contribution of the paper is to propose a faster version of the KL estimator based on a per-batch computation, we also wanted to verify that the batch-estimator could estimate the entropy with the theoretical bias of $\frac{1}{\sqrt{M}}$, M being the size of the batches. We also wanted to find a value of M that could offer a fair trade-off between the bias and the speed of the batch KL estimator, given that we are going to test the algorithm on 512×512 RGB images. The following results were obtained for $N = 512 \times 512$ and $d = 3$ random Gaussian variables.

```
KLD : bias :  5.7270824167888066e-06  timing :  1.863158941268921
KLBD-50 : bias :  0.0754319396627352  timing :  2.180079221725464
KLBD-100 : bias :  0.06365579783899733  timing :  1.4480175971984863
KLBD-500 : bias :  0.03543397182278696  timing :  1.016334056854248
KLBD-1000 : bias :  0.023172491842511356  timing :  1.0608181953430176
```

Figure 4: Bias obtained for the batch KLBD estimator with different values of M

Assuming that they way Python shuffles, slices, performs nearest neighbors search, etc ... leads to differences with the original paper that has been implemented in C on a more powerful machine, we do not care too much about the execution times. It seems that the bias of the chunk estimator indeed goes as $\frac{1}{\sqrt{M}}$. Based on those results, we fixed the batch size to $M = 500$.

The results we got with the three RGB images (Mandrill, Snake, Tiger) and the grayscale image (Lena) are reported bellow :



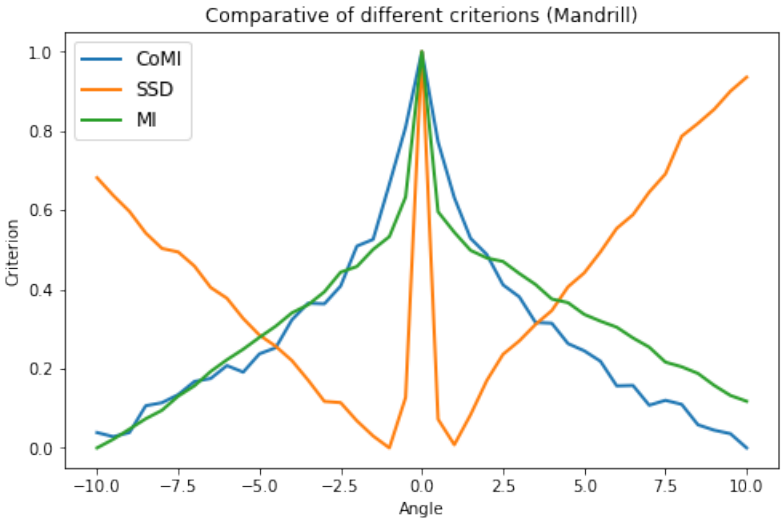


Figure 5: Results on Mandrill images

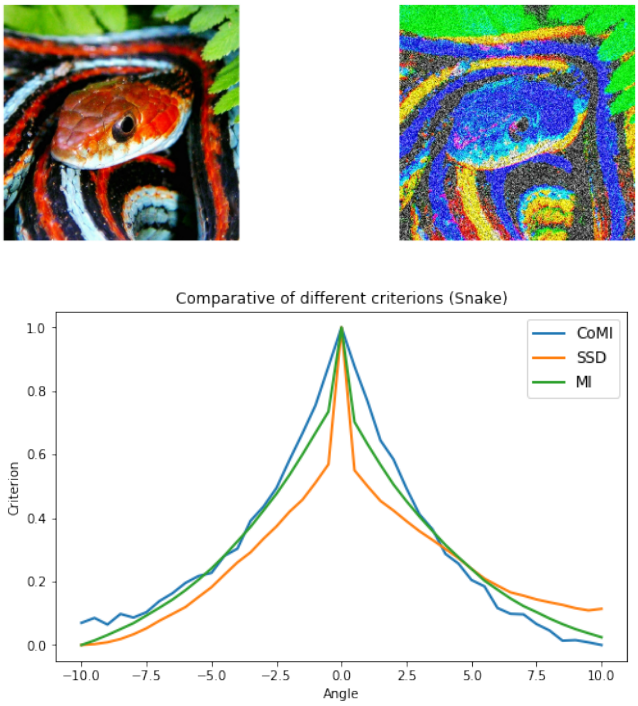


Figure 6: Results on Snake images

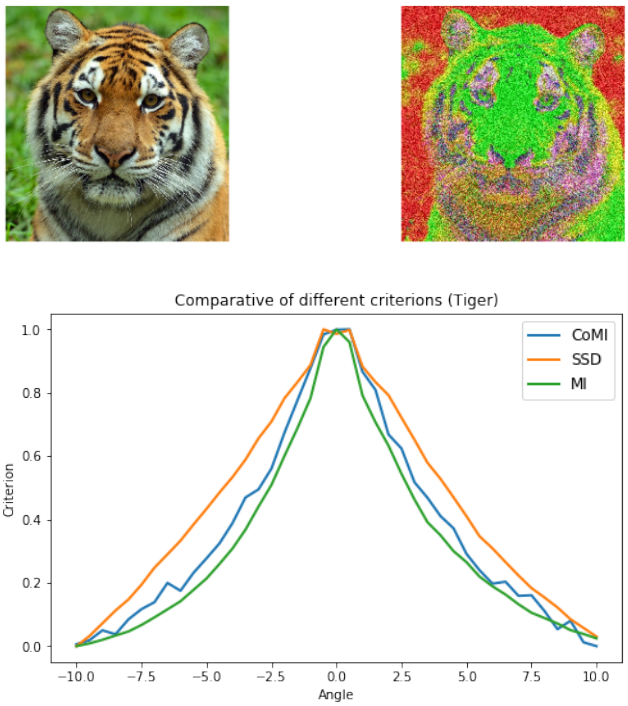


Figure 7: Results on Tiger images

The results we got were quite disappointing. Indeed, in all of the three cases the MI criterion seems more accurate than the the CoMI criterion, and except in the first case (the Mandrill) the SSD criterion does not get confused so much. If the CoMI criterion always recover the good rotation between the two images that are registered, it must be pointed out that it needs on average 2.41 seconds to be computed whereas the two other criterions are computed in 0.03 and 0.05 seconds (SSD and MI). Many parameters and various image processings were tried but the conclusions seem to be the same in all cases. We also know that the entropy estimator is correct (from the sanity checks) and that the experiments are done in the same conditions than in the original paper (because we had some email discussions with the author). The CoMI estimator seem to work well anyway, our problem was that we could not trick the SSD and MI estimators so that they get confused. We finally could not figure out why our results differed from the paper's results.

We then tried the NbMI estimators on grayscale images, for which we got the following results :

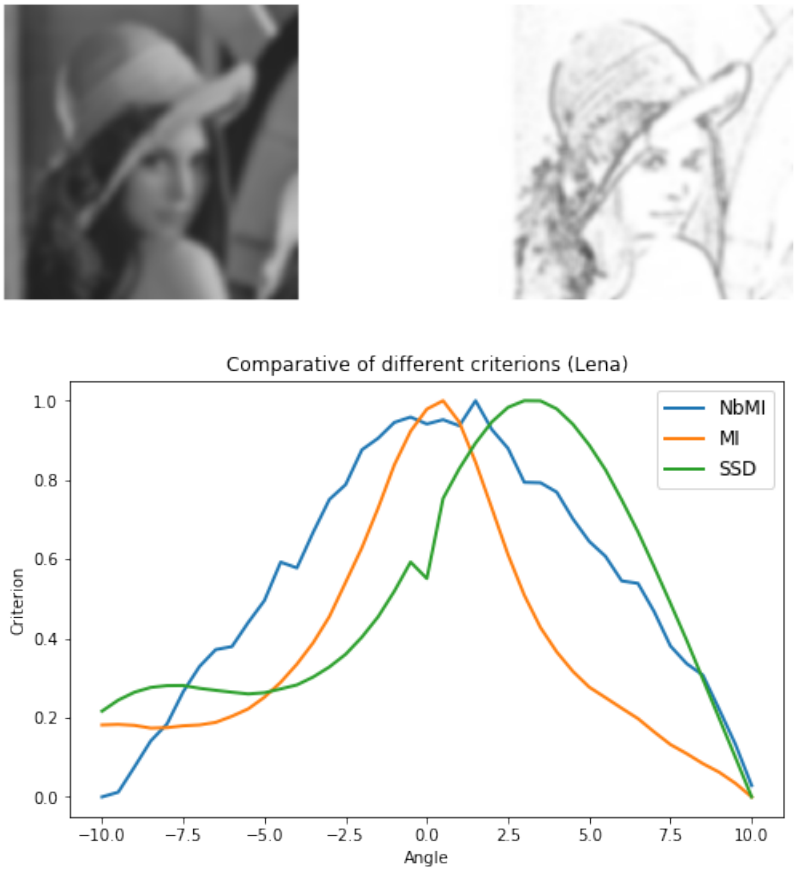


Figure 8: Results on Lena images

We used a window of $h = 2$ leading to a 25D feature map ($(2h + 1)^2$), and we padded the borders with -1's so that we could slide the neighborhood box on the whole image. Once again, our results differ from the results of the paper. This time the SSD criterion gets confused, but the MI is still quite relevant. It must be said however that as the NbMI criterion is stochastic, it would have been more fair to estimate the mutual information several times and report the average with one standard deviation. From the curves, it seems that the NbMI criterion could induce a good criterion for this image registration task, however the computational time (7.49 seconds per angle) does not justify to use it, because the MI criterion seems just as good and needs only 0.04 seconds to be computed at each angle.

In the geo-tracking example we tried to register a Google map classic view with its satellite-rendered counterpart. As real acquisitions are made by satellites, this could be used to perform tracking on classic views which are more memory efficient. However we had the same observations : SSD gets absolutely confused, NbMI gets the right result but does not compete with MI which is able to perform the registration just as well in about 120 times less time.

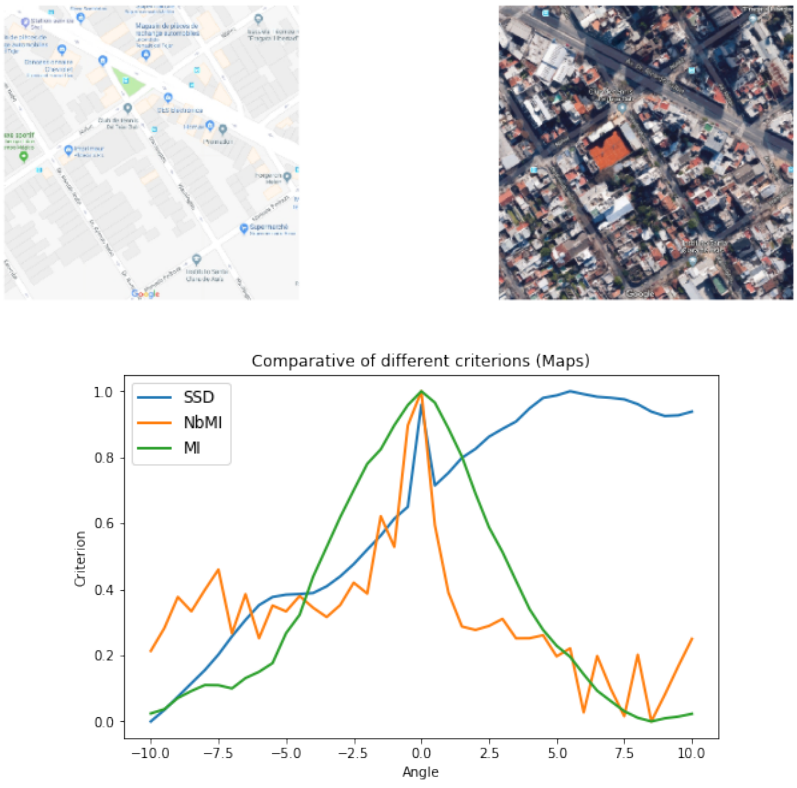


Figure 9: Results on Maps images

As for the face recognition task, surprisingly the SSD criterion is the best at detecting similar faces. However it must be pointed out that the CoMI and NbMI criterions are not very suited for this task, as a lot of factors can participate to confuse them : background, orientation of the face, variations of the faces among different images, ... The ability to detect edges and color similarities are not the only factors that have impact for this task. We noticed that the CoMI criterion seems good at clustering women faces together, maybe better than the SSD. We show bellow some examples, with the anchor image and the four most similars images found by the different criterions (see the notebook for more examples).

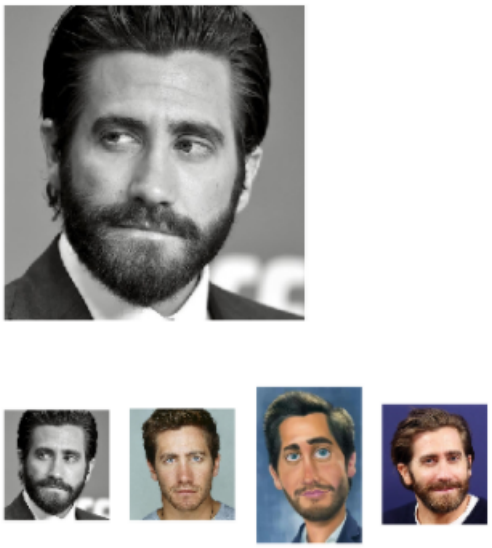


Figure 10: An example for which SSD performed perfectly

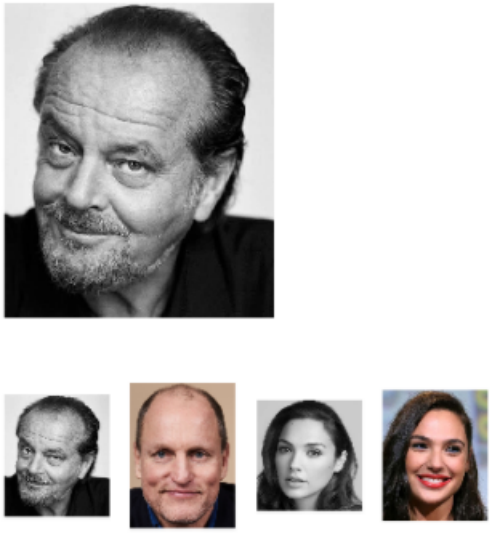


Figure 11: SSD got absolutely confused

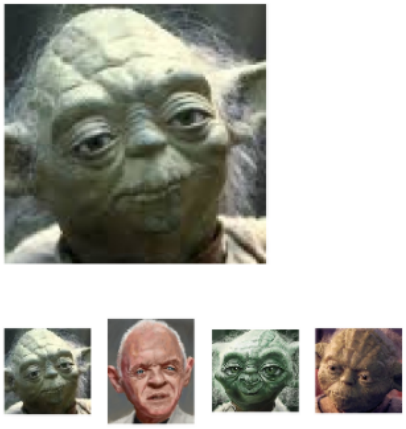


Figure 12: Master Yoda clustered with Anthony Hopkins !

Figure 13: Unsuprisingly, CoMI is good at clustering BN images together

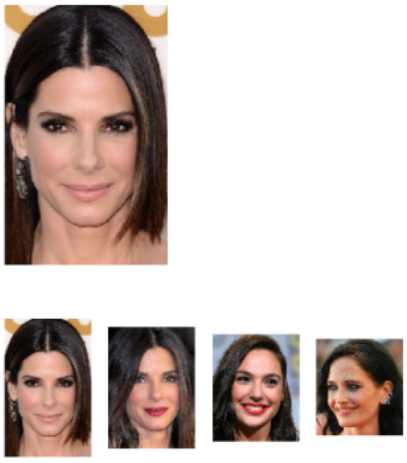


Figure 14: NbMI is good at clustering women faces together

V - Conclusion

To conclude this empirical study, it is clear that we got a bit disappointed and confused by our results. In most cases, the mutual information estimator indeed allows to perform image registration, but its use is not justified by its computational cost and most of all by the fact that simple mutual information based on histograms in lower dimension already allows to align the images correctly. We spent a lot of time trying to figure out why we could not reproduce the same results as in the paper, but we could not. Our implementation seems correct since we can recover perfectly the theoretical entropies for simple Gaussian signals and the conditions of the experiment have been discussed with the author of the paper in order to ensure there were no bottleneck on this side either.

According to our results, it seems that the method indeed allows to calculate entropies based on some more complex feature maps rather than just the pixel intensities, however it did not yield better results than simple estimators in our experiments.

Also, we implemented it in Python rather than in C as the author originally did. The way Python estimates the nearest distances is still much slower than the execution times reported in the paper, so we were disappointed about this aspect aswell. We could verify however that splitting the samples in several batches indeed allows to get a faster estimation without increasing the bias too much.

Typical applications of image registration include medical imaging, military automatic target recognition, images morphing, ... A huge drawback of the method is that it has to test all of the possible transformations between image F and image G so as to find the one transformation T such that F is close to $T(G)$. In the case of morphing for instance, some control points would be chosen in the source image F and the destination image G and for each intermediary image some registration would be applied so as to know how to move those control points. But the transformation to go from one step to the next step could include translations, rotations, affine mappings, warpings, or a composition of the four ... One can always parametrize and reduce the space of searches, but the proposed approach would require to compute and save the mutual information for every possible transformations, which would quickly become computationally costly.

Although we could not reproduce the results, we understand that it might possibly comes from some subtlety we did not spot. The lighten version of the binless KL estimator proposed by the author seems like a promising idea, as it allows to estimate entropy based on some complex features. The closed-form estimator is easy to compute and does not require any uncommon third-party library (we only needed numpy and scikit-learn for that purpose).