Class Project 2: Retinal image registration

Romain Wiorowski, Taha Zakariya, Camille Arruat
Dec. 23rd, 2021

***Abstract*—Image registration is key in many medical areas, and especially in retinal images analysis as it deeply helps diagnostic from multimodal and multitemporal sources. This project explores some of the registration techniques applied to retinal images of a large private dataset: cross-correlation, feature-based and deep-learning algorithm. Vessel extraction is also key for some of these methods and take an important part of this study. Preliminary results of most techniques give a glimpse of what could be done for improvement, while we couldn't obtain a satisfying deep-learning algorithm.**

I. Introduction

Image registration's goal is to align different images into one single coordinate system. In our study, retinal images need to be aligned together in order to help analysis and diagnostic, also through time (images of the same eye can be taken years later, with a different device).

More precisely, multiple retinal images are taken for each patient. Those images can be left or right eye, and centered on Optic Nerve Head or macula. The pictures who need to be registered together are the ones that share the same three parameters.

We can distinguish two types of registration: *single-modal registration* where the images that need to be aligned come from a same device, and *multi-modal registration* where multiple devices are used, involving variations in resolutions, contrasts, colors, ... Among the datasets we had to register, some of them were single-modal while others were multi-modal.

Many types of transformations can be applied in order to achieve such an alignment: rotation, scaling, change of resolution, shifting, some stretching... but flipping is not allowed.

When registering, two images are considered: one is referred as the *target* image and the other as the *source* or *moving* image. In our case, for a set of images we want to register together (same patient, same eye and same centerness), one will be chosen as the target and all other ones will be sources of this target.

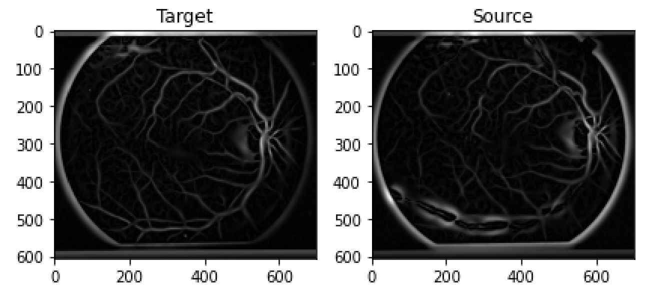


Figure 1. Two retinal images we want to be registered together. The source image (right) needs to moved to be aligned with the target image (left).

Among classic registration techniques [1], we can classify them into two main categories:

- *Intensity-based algorithms* work on intensity patterns in images via correlation metrics.
- *Feature-based algorithms* find relevant keypoints (points, lines...) as features and try to match them together.

Combination of these two [2] are often done for better results.

Image preprocessing also play a big role in the success of most of the registration algorithm, especially in retinal field. Lot of papers focus on this topic [3]. Vessels extraction is notably important for feature-based and matching algorithms, as it deeply helps feature detection and descriptor extraction.

II. Methods

We tried different approaches to this problem: the classic ones first before *exploring* more advanced ones.

First of all, each dataset takes the form of a folder containing all images that need registration, following a specific file name convention, which includes:

- patient identifier
- date
- left or right eye
- image type
- centerness: OHN (Optic Nerve Head) or macula
- image number

We build a pandas dataframe from these file names, and a script creates another one containing a list of couple images (a target and a source), being consistent with our registering sets (we only register images from same patient, same eye and same centerness). Within such a set, only one image is taken as the target, and all other ones are source images.

A. *Data preprocessing*

One of the first data preprocessing steps we tried is vessels extraction using different filters, also called ridge operators as they are adapted to detect lines:

- Meijering
- Sato
- Frangi

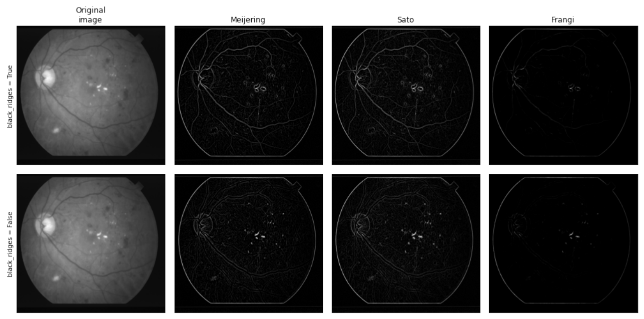


Figure 2. Different filters applied to a retinal image using Skimage implementations, with black_ridge being set to False, then to True. Note: this image comes from a different dataset.

During our following experiments, we figured out that the Meijering filter suits the most for our algorithms.

As we can see in Figure 2, such filters also recognize the circular edge of the picture, which can deteriorate the efficiency of some of the algorithms. Since images are taken from a specific set of devices, we can remove this white circle using its absolute position.

Such filters are also quite time consuming, and that's why we computed it on the whole dataset first, before experimenting on it.

B. *Using cross-correlation to compute shift and rotation*

After having filtered the images, we aimed to register one image, referred as the "source", with respect to another, referred as the "target". The first approach we tried is a rigid method based on cross-correlation. Indeed, cross-correlation is a measure of the displacement between two objects. The algorithm we implemented consists of two steps:

1) We started with the computation of the **angle of rotation** between the two images. This is done by transforming the images into polar coordinates and computing the cross-correlation. This returns the angle of rotation. We then rotate the source image by this angle.
2) The second step corresponds to the computation of the **shift** between the now rotated source image and the target image. For this, we compute the cross-correlation between the two images. We then shift the source image with respect to it.

To calculate the cross-correlation, we applied the *phase_cross_correlation* function from the scikit-image library.

C. *Feature matching*

Feature matching algorithms [1] consist of the following steps.

1) Feature detection: find relevant keypoints from the image
2) Descriptor extraction: extract the features into an adapted representation
3) Descriptors matching: with a proper metric, match both images descriptors
4) Outliers rejection: previous step often outputs lots of outliers
5) Warping: can be done with many transformation types (euclidean transformation, affine transformation, polynomial transformation, ...).

We tried several pipelines, adjusting every of these steps.

1) Feature detection: Among all feature detection algorithms, we can mention ORB, BRIEF, SIFT.

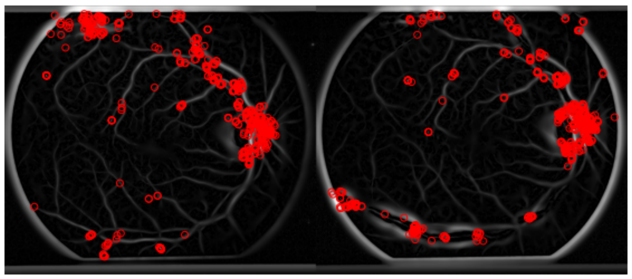


Figure 3. Keypoints obtained for target (left) and source (right) with ORB (n=500).

2) Descriptor extraction: Most of feature detection algorithms also come up with descriptor extractors (it's the case for ORB and BRIEF for example).

3) Descriptors matching: We choose to use a brute-force matching algorithm and its skimage's implementation (skimage.feature.match_descriptors).

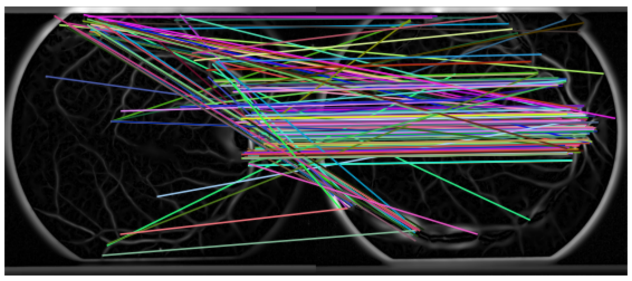


Figure 4. Matching of descriptors between target (left) and source (right) with brute-force algorithm.

4) Outliers rejection: We use RANSAC to reject outliers, with parameters min_samples=10 and max_trials=300. The residual threshold deeply depends on the previous choices.

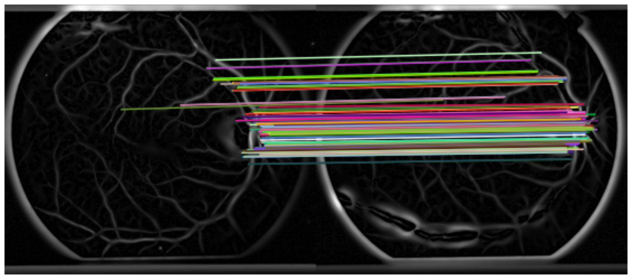


Figure 5. Matching of descriptors between target (left) and source (right) after outliers rejection (RANSAC).

5) Warping: We use skimage.transform.warp with the model we got through RANSAC. The moved image needs to have the same dimensions as the target, and it's filled with black pixels if needed. Results of an example can be found in the notebooks.

D. Deep-learning

Finally, we try the deep-learning approach to see how it compares with the standard unsupervised one. The only deep learning model we had had access to is VoxelMorph, so we based our approach on it.
Our reasoning consisted on the following steps:

- **Preprocessing**: we tried several approaches during preprocessing. We first tried applying the previously cited filters (meijering, sato ...) with different features (like ORB), but after many attempts, we noticed that loss is higher and we get better results without this kind of preprocessing. Thus our second approach consisted of only using the minimum necessary preprocessing, so we only transformed out image to grayscale and made it a square because VoxelMorph doesn't work otherwise. We also resized the image due to the lack of memory in the server as it couldn't process the model. Also, to make data homogeneous during the training, we divided data into categories depending on the eye position and centeredness. Finally, as deep-learning approach needs some ground-truth labels and a lot of data, we augmented our images by shifting and rotating them. This way the algorithm is supposed to converge better.
- Loss: one of the main concerns about this approach is the loss. After a lot of research [4], we concluded that for non rigid image registration, there exists two main losses to better convergence: normalized mutual information and structural similarity. We actually to implement using with VoxelMorph, but not having access to the model itself it failed: VoxelMorph's implementation uses tensors and transformations that make the traditional skimage library losses unusable. Thus, we unfortunately only used these metrics to test the output of the model, which would've been different we could have access to the model. Consequently we kept the available losses in VoxelMorph: MSE and MAE.

Working on the public image gave this output of 6, and gave a structural similarity of 0.94. As VoxelMorph wasn't flexible, we didn't have time to dig further and test the model with all the dataset, but with some more time and access to the internet, we can expect to make a good accuracy overall.

III. Results

Evaluating results imply finding an appropriate metrics. The considered ones are:

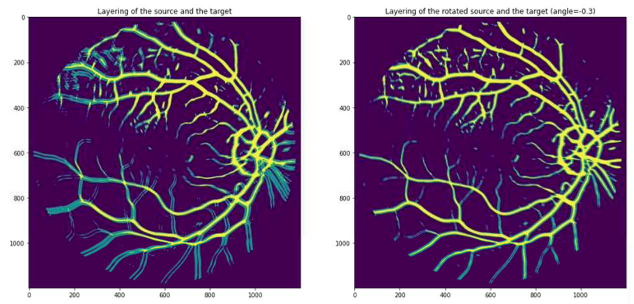
- *(Normalized) mutual information*, the one we will choose
- *Mean squared error*, but it seems that, even if it's good to measure perfectly registered images, it won't be able to recognize fine registrations.
- *Structural similarity*

We will for each pair of images to be registered together compare the similarity between:

- the target and the source image
- the target and the moved image

A. Cross-correlation method

This technique gives very good results: for example, on the public image given in the notebook, we can improve the mutual information from 1.33 (target-source) to 1.48 (target-moved). Applied on the overall dataset, the alignment are correct for most of the registration sets.


Layering of the source and the target
Layering of the rotated source and the target (angle=-0.3)
Figure 7. Computation of the rotation angle, and rotation of the source image. Layering of the source and the target (left) and layering of the rotated source and the target (right).

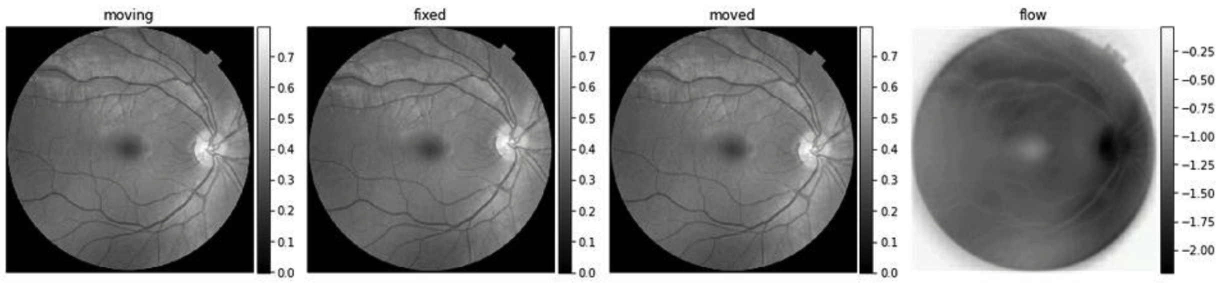


Figure 6. VoxelMorph public image registration

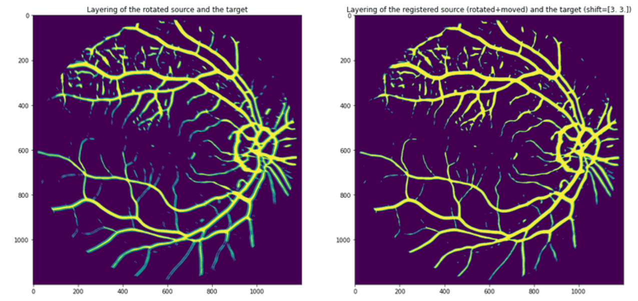

Layering of the rotated source and the target
Layering of the registered source (rotated+moved) and the target (shift=[3. 3.])

Figure 8. Computation of the shift and alignment. Layering of the rotated source and the target (left) and layering of the registered source and the target (right).

B. *Feature-matching method*

Applied to the whole private dataset, this technique improves the mutual information from 1.043 (target-source) to 1.066 (target-moved).

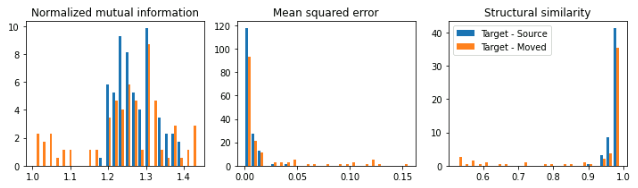


Figure 9. Distribution of target-source similarities (blue) and target-moved similarities (orange).

If we look closer to the registering sets, we can see that it mostly align in a good way. But in other case, it does very inappropriate stretching, mainly when contrast is low and vessel extraction struggles.

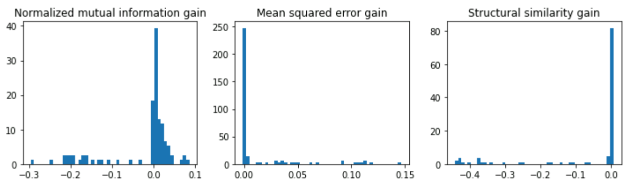


Figure 10. Distribution of gain in similarities

IV. Improvements and next steps

We faced several challenges during this project:

- The medical aspect of this project implied us to work on a secure and offline environment. Usage of pre-trained models for example wasn't that possible.
- We needed time to feel comfortable with the topic of image registration, its techniques, and what we could do. With more time, we would look deeper into deep-learning techniques. Also, (semi-)supervised techniques would be possible in other given datasets (some devices do themselves the alignment).

V. Conclusion

Through this project, we aimed at registering retinal images. We worked on images from the same patient, of the same eye and centered the same way.
We first preprocessed the images in order to enhance the vessels. We tried different filters, namely Meijering, Sato and Frangi. We decided to use Meijering or Sato for the continuation of the project as we observed that they performed the best on our images.
Then we applied different methods to register our images, trying approaches based on rigid and non-rigid transformations:

1) computation of rigid transformations using cross-correlation
2) computation of non-rigid transformations through features extraction and matching
3) computation of non-rigid transformations based on a deep-learning approach

Comparing the outputs of the algorithms, we conclude that the cross-correlation method produces good results and can be enhanced by computing more affine transformation parameters.

References

[1] E. G. Solares, "Image registration using scikit image," 2018. [Online]. Available: http://www.gilgalad.co.uk/post/image-registration-skimage/

[2] Z. Li, F. Huang, J. Zhang, B. Dashtbozorg, S. Abbasi-Sureshjani, Y. Sun, X. Long, Q. Yu, B. ter Haar Romeny, and T. Tan, "Multi-modal and multi-vendor retina image registration," *Biomed. Opt. Express*, vol. 9, no. 2, pp. 410–422, Feb 2018. [Online]. Available: http://www.osapublishing.org/boe/abstract.cfm?URI=boe-9-2-410

[3] Z. Ghassabi, J. Shanbehzadeh, and A. Mohammadzadeh, "A structure-based region detector for high-resolution retinal fundus image registration," *Biomedical Signal Processing and Control*, vol. 23, pp. 52–61, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1746809415001391

[4] DeepReg, "Image registration with deep learning." [Online]. Available: https://deepreg.readthedocs.io/en/latest/tutorial/registration.html