Lab 4 - Image Registration

Songyou Peng
psy920710@gmail.com

I. INTRODUCTION

In this lab, we implement a typical image registration framework based on simple rigid transformation as well as affine transformation. The sum of squared distance and mutual information are two similarity measures that we use for minimising the differences between fixed and moving images.

Also, a multi-resolution framework has been implemented and we will discuss the advantages of it. Throughout the report, we will talk about some observations based on different images, parameters or others.

II. REGISTRATION FRAMEWORK QUESTIONS ASWERING

In this part, we will answer the questions about the registration framework in the hangout.

- Identify each of the components of an image registration framework.
  A: First the "Transform" applied to moving images in our case can be either simple rigid transform (rotation and translation) or affine transform. The difference is just the matrix forms.
  After transformation, there may exist some blank areas in the new images, so "Interpolator" should be applied to fill the pixels without values. This can be found in `image_interpolation.m`.
  Then we can compare the difference between the fixed images and the transformed images based on a certain similarity measure (metric). Here we can use SSD or Mutual information (MI). The metric part can be found in the `affine_function.m` file as well as our own `mutual_info.m` function.
  Because we don't know if the current transformation is correct or not, we use an "optimizer" to iteratively minimise the similarity metric. Here we apply Matlab function `fminsearch` as the optimizer.

- What is the function of the scale vector?
  A: The scale of different measures varies. For example, the ranges of both x and y direction are from 1 to 256 pixels. If we move 1 pixel, not many things happen. However, the range for the angle is from 0 to 2π. Moving 1, in this case, would be a big deal. It may take a long time to get the result because it may always pass the correct position instead of approaching it. Therefore, we need to scale different measures. Here we should give translations 100 times larger scales than angels.

- Why is Gaussian smoothing used?
  A: As we know, after interpolation, many pixel values may be a bit different from the real values. If we don't smooth the image first, it may be difficult to register two images. After smoothing, the neighbour values becomes more or less similar, which increase chances for successful registrations.

- Where is the center of rotation of the transformation?
  A: Center of images. As we know, the rotation matrix rotates images with respect to (0,0). In the code `affine_transform_2d_double`, the center of images are set to (0,0) and then calculate the transformed coordinates.

III. MODIFICATION OF FRAMEWORK

Though the registration framework has been given, we still try to modify by adding mutual information, affine transformation, and multi-

resolution. Here we will go into the implementation details along with some discussions.

A. Mutual information (MI)

In order to acquire mutual information, we first need to get the entropy of fixed and moving images. We first get the histogram of both images and remove the bins without any elements, and then compute the sum of the multiplication of probability p_i of all the values and their corresponding $log_2 p_i$. The negative sign should also be added.

The second step is to calculate the joint histogram of two images and its entropy. Finally, the mutual information is the sum of the entropy of two images minus the joint histogram entropy. When two images are similar, the mutual information becomes high. Since our optimizer needs to minimize the metric, we need to give a negative sign for mutual information.

B. Affine transformation

Since simple rigid translation + rotation matrix can not deal with most of the complicated registration cases, we need to introduce the affine transformation, which contains also scaling and shearing [1]:

$$scale: \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, shear: \begin{bmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

After combining the rotation and translation matrices, the affine matrix can be written as:

$$M = \begin{bmatrix} s_x cos\theta + s_x h_x sin\theta & -s_x sin\theta + s_x h_x cos\theta & T_x \\ s_y sin\theta + s_x h_y cos\theta & s_y cos\theta - s_x h_y sin\theta & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

We can estimate 2 scaling + 2 shearing + 2 translation + 1 rotation = 7 parameters but for the sake of simplification, we model the 6 elements of the first two rows like:

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

the x can be set as: $[a,b,c,d,e,f]$. We will discuss about some observations about the affine transformation in section IV. When applying affine, remember the initial x should not be $[0,0,0,0,0,0]$ but $[1,0,0,0,1,0]$ because the former one is not even a valid configuration so really far away from the correct answer. The optimizer may get stuck in the local extreme (Fig. 1).

(a) $x = (0,0,0,0,0,0)$ (b) $x = (1,0,0,0,1,0)$

Fig. 1. Comparison between two initial x using affine transformation

C. Multi-resolution (MR)

Once mutual information and affine transformation have been implemented, we tried to include multi-resolution scheme into the registration framework.

After smoothing the fixed and moving images, we respectively build image pyramids for both images with scaling factor 2 (Fig. 2). Once we have the pyramids, we use the optimizer to find the best x from the smallest scale, and then use it as the input for the next scale. Finally, we can get the good results.

Fig. 2. Image Pyramid

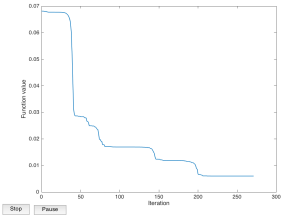
We know the scaling factor of the pyramid is 2. It should be noted that once acquired a x for the current image, we need to multiply 2 times for the translation terms inside x before inputting to the next iteration. If not, it takes longer time for the optimizer to find the correct x or even get stuck

(a) Fixed image

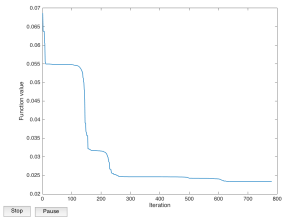
(d) Moving image

(b) Rigid

(e) Affine



(c) metric against iterations of rigid



(f) metric against iterations of affine

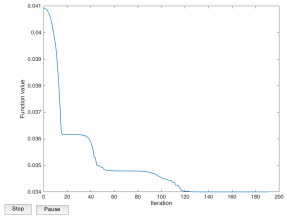
Fig. 3. Comparison between rigid and affine transform in the registration of `lena1` with SSD

(a) Fixed image

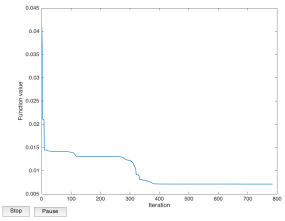
(d) Moving image

(b) Rigid

(e) Affine



(c) metric against iterations of rigid



(f) metric against iterations of affine

Fig. 4. Comparison between rigid and affine transform in the registration of `lena3` with SSD

in the wrong place. For example, when we use SSD and rigid transformation and set `lena2` as fixed image and `lena1` as moving one, it takes 7 seconds for scaling translation and 10 seconds for no scaling.

IV. Results & Observations

In this part, we will show different comparison results, based on which we will discuss our observations.

A. *Affine v.s Simple Rigid*

For the simple rigid transformation, dealing with simple registration case is suitable, e.g. `lena1`. However, if the registrations are more than rotation and translation (`lena3` also needs scaling), of course, the results are not decent while the affine transform can give a better result.

For the affine transformation, it can cope with most of the registration cases, but it works well only if the multi-resolution framework is used. For example in Fig. 3, the result acquired from single image affine transform is bad, but applying MR is able to give a decent registration. The comparison is shown in Fig. 5.

In a nutshell, the rigid transform can bring good results if the registration cases are simple. Affine transform works for a large number of the cases, but it is not quite stable when only one single image are used. So the affine transformation should be used along with multi-resolution.

B. *Multi-resolution v.s Single Image*

As we have already discussed in the last two parts, the affine transform cannot always get good results if only one single image is used as input.

In our opinion, the reasons why the single image affine results are bad are: Firstly, it is harder for the optimizer to get the correct extreme of affine transformation than rigid transformation due to the

(a) Affine without MR

(b) Affine with MR

Fig. 5. Illustrations for multi-resolution affine transform in the registration of `lena1` with SSD

(a) Affine without MR. SSD: 0.0071

(b) Affine with MR. SSD: 0.00041

Fig. 6. Illustrations for multi-resolution affine transform in the registration of `lena3` with SSD

number of elements that needed to be estimated. Also, the optimizer we are using is not quite strong and easily to get stuck in the local minimum. Multi-resolution framework can solve the problem because even though the optimizer starts with a wrong direction and gets stuck in the local minimum in lower resolution images, it will start searching again for the next higher resolution and it is not very difficult to find a correct direction because the local minimum is not far from the starting point.

In Fig. 6, we can see that applying multi-resolution gives a much better result when the initial guess is the same. Moreover, MR only takes 13 seconds to get the result while single image affine needs 57 seconds.

Overall, the multi-resolution framework can not only give better registration results but also accelerate the process.

C. MI v.s SSD

After dense experiments, we found out mutual information need to be given really good initialization, otherwise, the register results are really bad. As seen in Fig. 7, (b) initial x is close to the

(a) $(1,0,0.2,0,1,-0.5)$

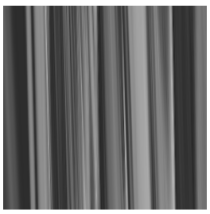
(b) $(1,0.1,0.2,-0.1,1,-0.5)$

Fig. 7. Illustrations for multi-resolution affine transform in the registration of `lena3` with MI. (a) MI: -1.209, (b) MI = -6.347. Ground truth is $x = (0.891, 0.0783, 1.370, -0.076, 0.933, -2.121)$

(a) Fixed image

(c) Moving image

(b) MI



(d) SSD

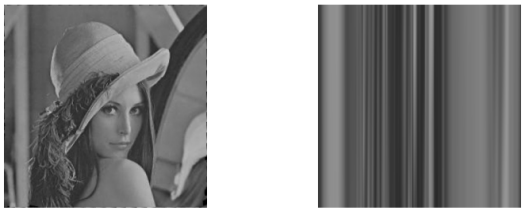
Fig. 8. Illustrations for multi-resolution affine transform in the registration of `lena3` with MI. (a) MI: -1.209, (b) MI = -6.347. Ground truth is $x = (0.891, 0.0783, 1.370, -0.076, 0.933, -2.121)$

ground truth and the result is decent, while when we change a bit in (a), the result becomes bad.

Although compared with SSD, most of the time MI cannot provide satisfied results, there is one case that MI outperforms SSD, which is the inverse version of the image. As we can see in Fig. 8, we can figure that MI at least try to fit the boundary of the image, while SSD totally gives a really wrong registration result.

D. Scaling Factor

Finally, we will talk about the influence of the scaling factor on the registration result. When the scaling factor is set properly, the system can reach results faster. Moreover, because the scale



(a) Proper scale (b) Equal scale

Fig. 9. Illustrations for multi-resolution affine transform in the registration of `lena1` using different scale

is too small, which lead to the moving step in the optimizer is also small, the result may end at the local minimum. In the Fig. 9, the result that the translation parameters are set 100 times to other parameters is much better than setting all parameters equally.

V. CONCLUSION

In this lab, we implement some parts of a complete image registration system. We compare the performance between single image and multi-resolution framework, as well as the advantage and disadvantage of two similarity metrics: SSD and MI. What's more, some parameters like the initial input and scaling factor are tuned in order to analyse the influence.

REFERENCES

[1] House, Donald H. "Affine Transformations." Clemson.edu. Accessed April 29, 2016. https://people.cs.clemson.edu/ dhouse/courses/401/notes/affines-matrices.pdf.