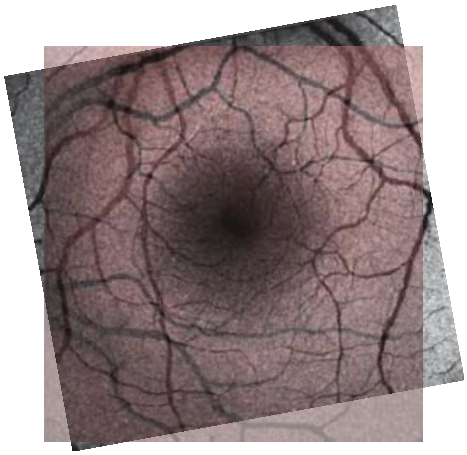
Medical Image Processing 67705
Exercise 4: Rigid Registration

Prof. Leo Joskowicz
TA: Adi Szeskin

Due date: Sunday Dec 27, 2020



In this exercise you will implement rigid registration of 2D retinal scans. Basically, you will get several cases which comprise of a patient's base-line (BL) and follow-up (FU) scans, and you'll need to register the FU to the BL.

For this exercise, you may assume that both images in every case are of the same size and resolution and are isotropic.

You may use functions from the utils.py.

Submission

Write a report in a single pdf file called Ex4_part[#].pdf and your code in a Ex4_part[#].py file. Submit through the Moodle web page a single ZIP archive called [id]_ex4.zip.

Your report should include the answers to all the questions of the exercise as well as images of your results. Make sure to write clear answers and pay especial attention the graphs you crate: all graphs should have a title, a label for each axis, and be of the correct scale. Make sure to write in the report your name, id and CS user name. Note that 10% of the grade is for clear and readable submission. Late submissions will be penalized by 10 points per day.

Background

Two different images of a patient, taken at two different times, usually have significant movement between them. This is because the patient is in different poses, because of internal movements, e.g., breathing, and because of other physical changes that occurred in the time that passed between the scans. Registering the two images is needed to compare the content, e.g., track the differences, or to evaluate the efficacy of a treatment.

Rigid registration consists of computing the translation and rotation that aligns two images assuming that the anatomical structures of interest retain their shape. Rigid registration of 2D images requires computing three parameters: two translations and one rotation, while 3D scans requires computing six parameters: three translations and three rotations.

Rigid registration algorithms can be categorized into two groups: geometry and intensity based. In geometric-based algorithms, the features to be matched, e.g. points are first identified in each image and then paired. The sum of the square distances between the points is then minimized to find the rigid transformation. In intensity-based algorithms, a similarity measure is defined between the images. The transformation that maximizes the similarity is the desired one. The rigid registration algorithms are iterative – in each step, a transformation is generated and tested to see if it reduces the sum of the squared distances between the points or increases the similarity between the images.

Part 1 – Point and image based registration algorithms

Task A (60%): The goal of this task is to find a rigid registration between two ophthalmology 2D images. The images are Fundus Auto Fluorescence images (FAF) of patients that suffer age related macular degeneration (AMD), a condition of retinal atrophy that causes vision loss. For a description of these images, see here.

Background: Point-based registration algorithms compute transformation between two images in four steps:

1. Find feature points in both images
2. Pair the points between the two images
3. Remove outliers
4. Compute the transformation that minimizes the squared sum of distances between the paired points

For this task, we will assume that the feature points have been identified and that the points have been paired. However, the pairing may include outliers. To remove them before computing the transformation, use the RANSAC method.

Download the images and the utils.py code from:

https://drive.google.com/open?id=1qAL93qYRRuR6cFAtFdJVNF_OrCA_WqD0

Follow these steps:

1. (5%) Load the images BL01.tif, FU01.tif and use the function [BLPoints, FUPoints] = getPoints('no_outliers')

The function returns two matrices of size *n*x2 where *n* is the number of points such that each row is a 2D point in the coordinate frane of the image. The points are paired per row between the matrices, e.g. row 5 in 'BLPoints' and row 5 in 'FUPoints' are a pair).

Create a plot showing the point pairs in a plots using a function you will write for this task. Use the annotate function to show the point number next to the point on the image. Report how many wrong matches you find.

2. (20%) Implement the function calcPointBasedReg(BLPoints, FUPoints) – BL = Baseline points; FU = Follow-up points

The function returns a 3x3 rigidReg rigid 2D transformation matrix of the two translations and rotations of the given points and pairings. The matrix is such that applying it to FUPoints yields the points that are closest (least squared distance) to the BLPoints. To compute the matrix, use the Single Value Decomposition (SVD) code provided in this link.

The resulting matris should approximately satisfy the following relation:

[FUPoints ones(N,1)] * rigidReg == [BLPoints ones(N,1)]

Where *N* is the number of points. The matrix rigidReg is formed by a rotation matrix and a translation vector [R(2x2) 0; T(1x2) 1].

3. (5%) Implement the function - calcDist(BLPoints, FUPoints, rigidReg)

   The function computes the distance of each transformed point from its matching point in pixel units. Compute the Root Mean Squared Error (RSME), the error of the registration that results from the computed transformation. Note that the function returns a vector of length *N*. For this task, you need to compute a single scalar value from this vector (in unit of pixels). Compute the error of the transformation for the given points pairing provided on Part A (in pixels).

4. (30%) Implement a function that loads two images – BL and FL – and a set of chosen point, computes the rigid registration transformation between them, and applies it to the FU image.

   In addition, the function will compute a new image consisting of the transformed FU image overlaid on top of the the BL image. (Hint: you can use an edges image of FU in overlay above the BL, or have each in different colors). Add to the report a figure of the two images after registration.

5. (5%) For the following datasets: [BLPoints, FUPoints] = getPoints('with_outliers')

   Repeat parts 1 and 4. Are the images properly registered? What is the registration error? Is this error acceptable? Explain.

6. (30%) Read the documentation and examples of **utils** => **ransac**.

   Use it to write the function [rigidReg, inliers] = calcRobustPointBasedReg(FUPoints, BLPoints)

   This function will compute the transformation with unknown outliers in the pairs list. The **ransac** function inputs two functions and applies them to the data. One function is used to compute the transformation and the other for computing the error. For this task, use the functions you previously wrote.

7. (5%) Repeat item 5, but this time use the robust function to compute the transformation. Compute an image that shows in different colors the distinction between inliers and outliers.

8. To check yourself, create a new set of points and apply a known transformation to them, to get a new set of points. Add small random noises and outliers. Then, recreate the transformation with the functions you wrote. There is no need to submit code or answers for this.

Useful functions for this part:

- from skimage import transform as tf
- AffineTransform
- warp

Task B (40%):

To prepare for the following exercise, implement the following functions:

- SegmentBloodVessel(Image) – perform segmentation of the blood vessels in the retina. You may assume the atrophy in the center (this is the AMD atrophy, marked by the X in the image below) is not part of the segmentation, and the caption at the bottom can be removed.
- FindRetinaFeatures(Image) - The function will find strong features in the image to use for registration. You can use algorithms like SURF, SIFT and ORB.

For this case you will pick at least 2 other cases (BL and FU) from the Google Drive I sent you, to check robustness of your algorithm.