CS 419 Homework 1 Report

Melih Kaan Şahinbaş

October 15, 2024

Chapter 1

Introduction

Retinal images are essential for medical assessments, and accurate alignment of these images is critical for effective diagnosis and treatment. This report describes the methodology and results of registering retinal images from the FLORI21 dataset, leveraging control points provided by medical professionals.

This report details the process of registering retinal images acquired across multiple patient visits. The goal is to calculate the affine transformation matrix between pairs of images, apply this transformation to align the images, and evaluate the accuracy of the registration using mean squared error (MSE). The report also includes a discussion of the results obtained.

Chapter 2

Methodology

2.1 Affine Transformation

An affine transformation can be mathematically represented as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & tx \\ c & d & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

where (x, y) are the original coordinates, (x', y') are the transformed coordinates, and a, b, c, d, tx, ty are the parameters of the affine transformation.

2.2 Matrix Representation

To calculate the affine transformation matrix T from the control points, we express the relationship between the control points as (

$$x_n, y_n$$

denotes for coordinates in the reference image

$$x'_n, y'_n$$

denotes for coordinates in the test image:

$$A \cdot T = B,$$

where:

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_N & y_N & 1 \end{bmatrix},$$

and

$$B = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_N \\ y'_N \end{bmatrix}.$$

The transformation matrix T can be computed using the least squares method:

$$T = A^{-1} \cdot B.$$

2.3 Affine Transformation Matrix

The calculated affine transformation matrices for the two pairs of images are presented below:

$$\text{Matrix}_1 = \begin{bmatrix} 0.994820069 & -0.0672158598 & 115.212168 \\ 0.0680550151 & 0.995776104 & -67.5559369 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$\text{Matrix}_2 = \begin{bmatrix} 1.62683291 & 0.0840784864 & -1505.87566 \\ 0.00992188434 & 1.27168356 & -728.344685 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Last row is added for calculation

2.3.1 Applying the Affine Transform

After calculating and modifying the affine transformation matrix, the subsequent step is to apply this transformation to the image.

```
transformed_image = cv2.warpAffine(test_image, extended_affine_matrix[:2],
                        (test_image.shape[1], test_image.shape[0]),
                        flags=cv2.INTER_CUBIC,
                        borderMode=cv2.BORDER_CONSTANT,
                        borderValue=(0, 0, 0))
```

The transformation matrix is applied to the image, resulting in a newly transformed image. The cv2.warpAffine() function executes the transformation by using the affine matrix on every pixel in the image, employing cubic interpolation to estimate values for any pixels that may not have a corresponding output.

2.3.2 Padding Calculation

Before applying the transformation, padding is calculated to ensure that the transformed image fits within the image boundaries. The maximum height and width are computed as follows:

$$\text{max\_height} = \max(h, h \cdot |m_{01}| + w \cdot |m_{00}|) \tag{2.4}$$

$$\text{max\_width} = \max(w, h \cdot |m_{11}| + w \cdot |m_{10}|) \tag{2.5}$$

where h and w are the height and width of the original image, and m_{ij} are the components of the transformation matrix T.

Components of transformation matrix is used to calculate possible edges of the image after the translation and if it exceeds the boundaries, padding is added to reduce the error and prevent significant losses.

2.3.3 Inverse Transformation

To revert the transformed image I' back to its original state, we utilize the inverse transformation matrix. The following line of code applies this transformation using the `cv2.warpAffine` function:

```
I_double_prime = cv2.warpAffine(I_prime, inverted_transformation_matrix[:2],
                        flags=cv2.INTER_CUBIC,
                        borderMode=cv2.BORDER_CONSTANT,
                        borderValue=(0, 0, 0))
```

This line effectively recovers the original image by applying the inverse transformation to the transformed image, accounting for any padding applied previously.

2.3.4 Mean Squared Error Calculation

The difference between the original and recovered images is assessed using the Mean Squared Error (MSE) formula:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (I_{\text{original}}(i) - I_{\text{recovered}}(i))^2 \tag{2.6}$$

where $I_{\text{recovered}}$ is the picture that is obtained after applying and then reversing the affine transformation, and I_{original} is the original image. The MSE provides a numerical indicator of the amount of error introduced by the transformation. A lower MSE suggests that the transformation was near the target because the recovered image is closer to the original image.

In the code, the MSE is calculated with the following expression:

```
mse = np.mean((img - recovered_image)**2)
```

2.3.5 Comments on the Error

An important metric for assessing the quality of image translation is the Mean Squared Error (MSE), which gives a numerical representation of the difference between the original and modified images. In this instance, Test Image 1's MSE was estimated to be around 0.39, whereas Test Image 2's MSE was noticeably higher at around 2.19. This discrepancy can be explained by the fact that Test Image 2 required padding because MSE was first calculated by 40 for second image before any padding was added. This was because lack of padding in the image and translated imaged exceeded the borders. In order to keep the translated image from exceeding its bounds, padding had to be added, which declined the error value. Even though the present findings show a noticable variation in MSE, there are a number of tactics that might be able to further lower these errors. Improved alignment of translated pictures may result from investigating new methods, modifying parameters, or applying various functions to improve image registration accuracy and reduce the MSE.