LAB REPORT - 2

IMAGE PROCESSING

November 19, 2018

Submitted to

Desire SIDIBE

ASRAF ALI Abdul Salam Rasmi

Masters in Computer Vision

Centre Universitaire Condorcet

Universite de Bourgogne

Chapter 1

Geometric Transformation

1.1 Introduction

Geometric Transformations are widely used for image registration and the removal of geometric distortion which is useful in construction of mosaics, geographical mapping, stereo and video [1]. In the previous lab we were dealing with **Image Enhancement** techniques like Histogram modification in which the Intensity level of the pixels were changed but in Geometric Transformation we will change the position of pixels leaving the intensity unchanged (Intensity levels can be modified if needed like in Interpolation).

1.2 Affine Transformation

Affine Transformation is the simplest Geometric Transformation in which the collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation) are preserved [2]. In this lab we performed two Affine Transformation techniques, Translation and Rotation

1.2.1 Tranlation

Translation generally means shifting a pixel vertically (by t_x) or horizontally (by t_y) or both. The translated image will have the same axis orientation with the input image with some pixels removed. Translation will not affect the pixel values and so no further processing necessary. It is given by,

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

This is performed in MATLAB with the following lines of code.

```
for i=1:r
    for j=1:c

        % Shifted vertically by tx and horizontlly by ty
        O(i+tx:end, j+ty:end, :)  = I(i:end-tx,j:end-ty,:);

    end
end
```

Results

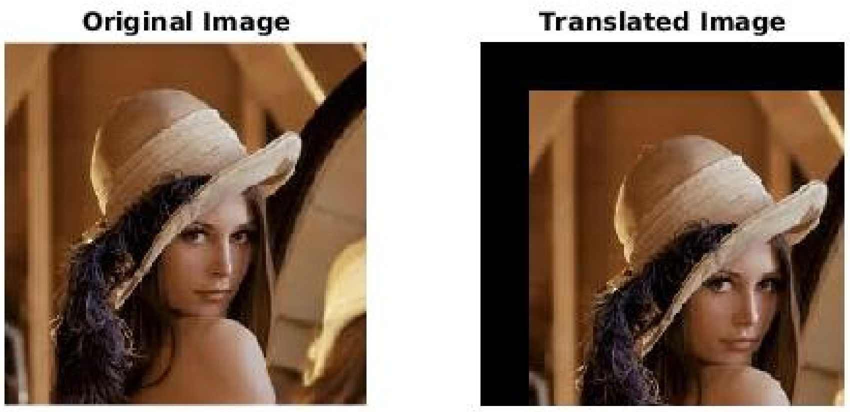

Figure 1.1: Output of Translation

1.2.2 Rotation

In general, Rotation is the shifting of an image by an angle θ usually rotated about the center of the image (x_c, y_c). It is given by,

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}$$

This is performed in MATLAB with the following lines of code.

```
for i=1:r
    for j=1:c

        % Rotation steps (based on the Formula)
        x=(i-midx)*cos(theta)-(j-midy)*sin(theta);
        y=(i-midx)*sin(theta)+(j-midy)*cos(theta);
        x1=round(x)+midx;
        y1=round(y)+midy;

    end
end
```

When rotation is performed the pixels will be aligned properly which will create holes in the rotated image, as shown in figure 1.2. This can be solved by performing Interpolation of the pixels. Here we used two Interpolation techniques.

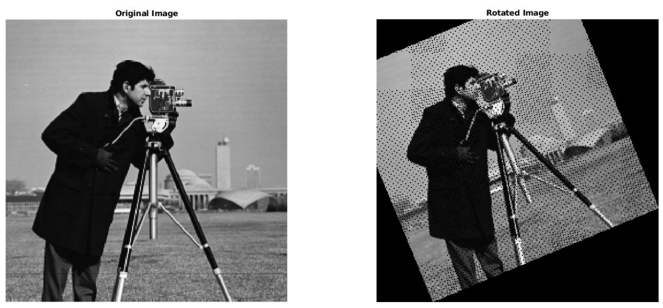

Figure 1.2: Rotation without Interpolation

- **Nearest Neighbor Interpolation** calculates the pixel intensity by averaging the intensity values of the four neighbor pixels in the original image.

  This is performed in MATLAB with the following lines of code.

```matlab
if (down ≤ r && right ≤ c)
    intensity_1 =  I(up, left);
    intensity_2 =  I(down, left);
    intensity_3 =  I(up, right);
    intensity_4 =  I(down, right);
    intensity = (intensity_1 + intensity_2 + intensity_3 + ...
        intensity_4)/4;
end
```

- **Bilinear Interpolation** uses points from a bounding rectangle. Useful when the known points are on a regular grid. It can be performed simply as multiple linear interpolation and is given as follows.

$$O(x,y) = (1-a)(1-b)I(i,j) + a(1-b)I(i+1,j) + b(1-a)I(i,j+1) + abI(i+1,j+1)$$

  This is performed in MATLAB with the following lines of code.

```matlab
if (down ≤ r && right ≤ c)
    intensity_1 =  I(up, left);
    intensity_2 =  I(down, left);
    leftIntensity = (x1-up) * (intensity_2 - intensity_1) + ...
        intensity_1;
    intensity_3 =  I(up, right);
    intensity_4 =  I(down, right);
    rightIntensity = (x1-up) * (intensity_4 - intensity_3) + ...
        intensity_3;
    intensity = (y1 - left) * (rightIntensity - leftIntensity) ...
        + leftIntensity;
end
```

Results

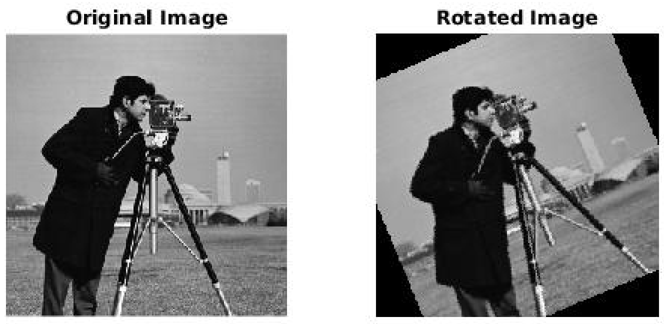

Figure 1.3: Rotation (Nearest Neighbor Interpolation)

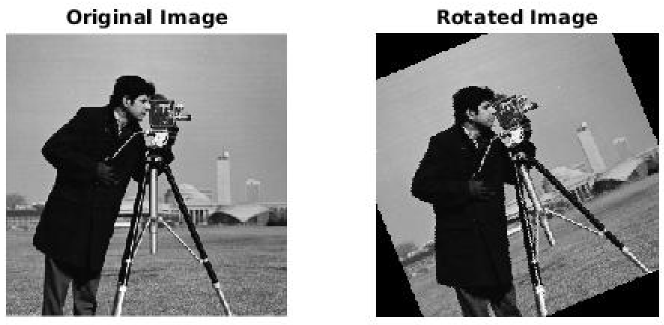

Figure 1.4: Rotation (Bilinear Interpolation)

1.3 Projective Transformation

Projective Transformation is a combination of Affine Transformation and Projective warping. The idea is to project an image taken at angle onto another image of the same scene but at different angle. In the Projected Image, the lines are mapped as lines but the parallel lines are not necessary to be parallel. Here we used MATLAB built-in function 'fitgeotrans' (previously 'cp2tform') for performing Projective transformation and 'imwarp' (previously 'imtransform') to fit the transformed image into the reference image.

Results

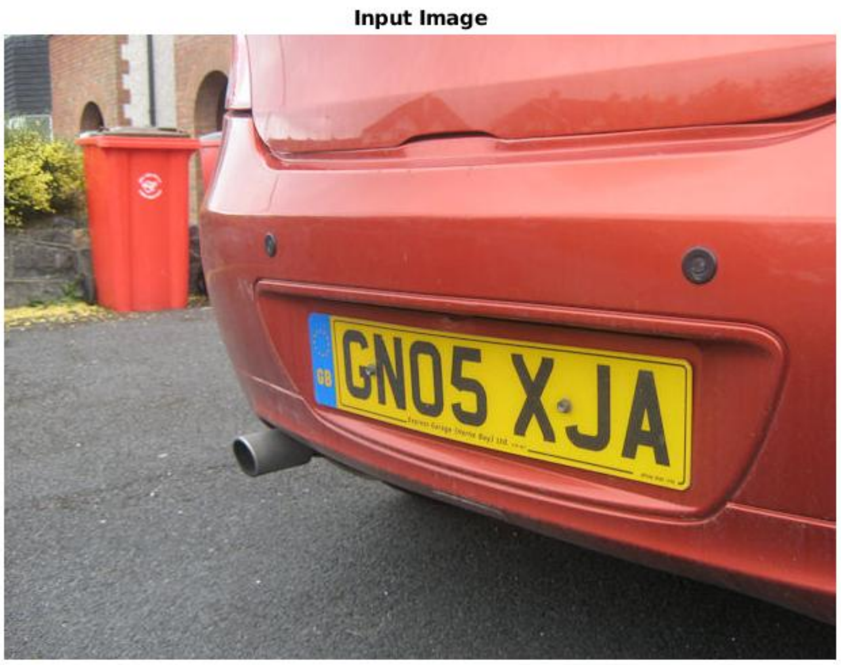

Figure 1.5: Input Image

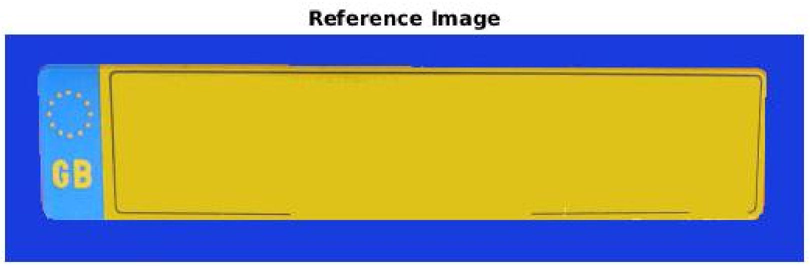

Figure 1.6: Reference Image

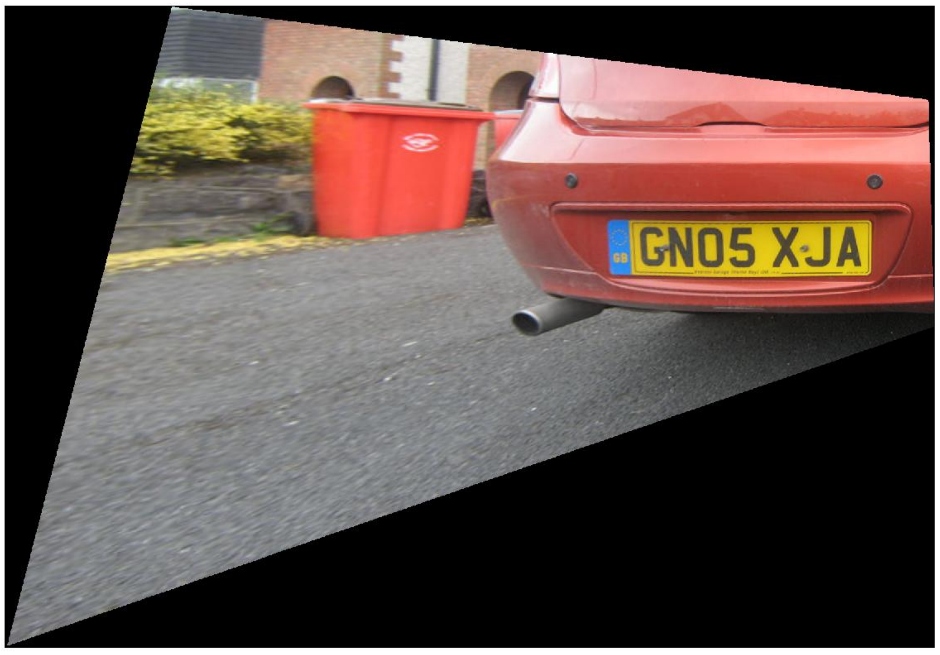

Figure 1.7: After Projective Transformation

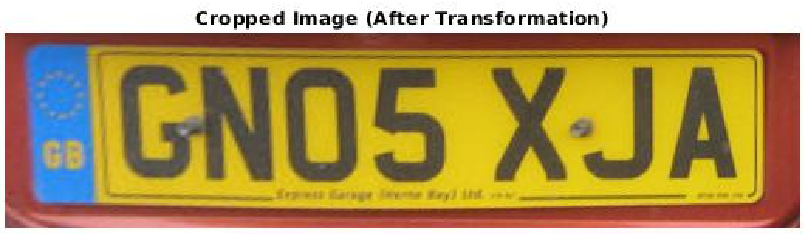

Figure 1.8: Cropped Image (After Transformation)

1.4 PROCRUSTES ANALYSIS

The aim of Procrustes method is to align the input coordinates with the reference in order to minimize the Euclidean distances between points without changing the shape. It is performed with three independent tasks.

- **Translation** - Place the centroid of both input points set and reference points set at the origin.

$$X_i = X_i - \bar{X}$$

- **Scaling** - It is done to transform the input coordinates to same size of the reference coordinates (either by stretching or compressing). Scaling is given by,

$$S = \frac{X_i}{\sqrt{X_i^2 + Y_i^2}}, \frac{Y_i}{\sqrt{X_i^2 + Y_i^2}}$$

- **Rotation** - Rotation matrix is computed by performing 'svd' to XY^T. After computing Rotation matrix, R, we multiply R with the Scaled X to fit the input coordinates with reference coordinates.

Results

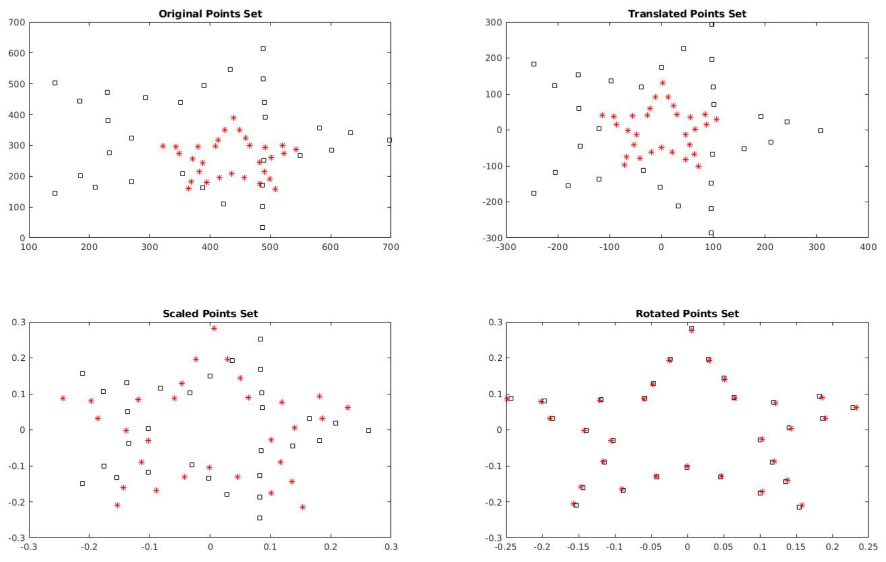

Figure 1.9: Procrustes Analysis for Star Points

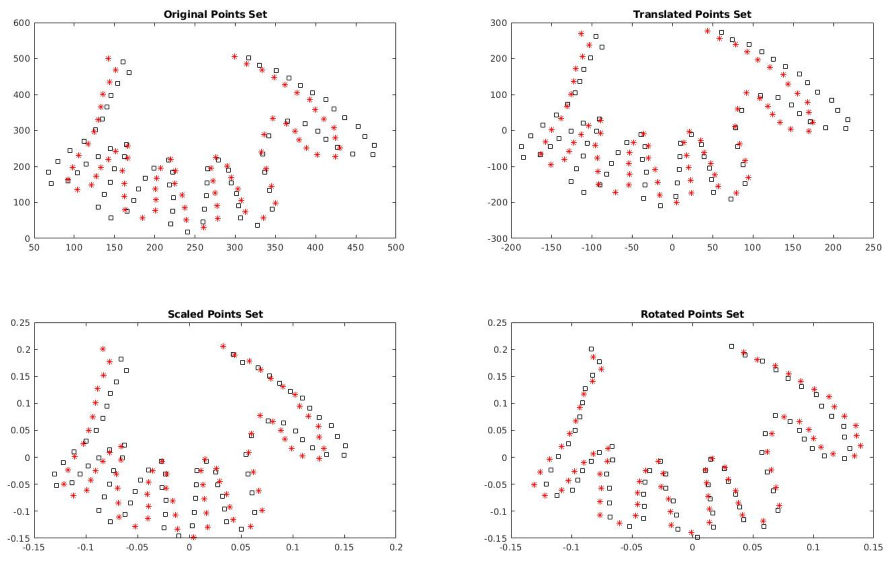

Figure 1.10: Procrustes Analysis for Hand Points

Bibliography

[1] H. Rhody, “Geometric image transformations,” in *Lecture Notes : Rochester Institute of Technology*, 2005.

[2] Y. Wang, “Geometric transformations: Warping, registration, morphing,” in *Lecture Note : Polytechnic University, Brooklyn, NY*, 2009.