86735: Computer Vision
Image warping and bilinear interpolation

Due date: Wed 14.15, 14.10.15

Prof. F. Odone, Prof. F. Solari, M. Chessa, N. Noceti

Ernest Skrzypczyk, BSc

Problem 1

The goal of the practical laboratory session was to perform backwards warping of an image using bilinear interpolation. Rotation and translation were the only transformation operations required. The formula for forward warping is represented by equation (1):

$$T_{\Theta}(x) = s\begin{bmatrix}\cos(\phi) & -\sin(\phi)\\ \sin(\phi) & \cos(\phi)\end{bmatrix}x + \begin{bmatrix}t_x\\ t_y\end{bmatrix} \tag{1}$$

Contrary to the forward warping transformation matrix, the inverse transformation matrix represented by equation (2) is used on the output image matrix to calculate its values in reference to the original image and its interpolated values at newly created points in the grid. In Matlab the interpolation operation is performed by the command *griddata*.

$$T_{\Theta}^{-1}(y) = \frac{1}{s}\begin{bmatrix}\cos(\phi) & \sin(\phi)\\ -\sin(\phi) & \cos(\phi)\end{bmatrix} + \left(y - \begin{bmatrix}t_x\\ t_y\end{bmatrix}\right) \tag{2}$$

The rotation angle Θ and translation coordinates for the 2D array representing the image t_x and t_y are the only parameters used in the script (SC1) written in Matlab. Several asymmetric values have been used to perform those operations. The results are presented in the processed images below the script.

Script 1: Script written in Matlab.

```
%% Ernest Skrzypczyk - 4268738
%% 07.10.2015
%% Computer Vision - L01 - Image Warping and Bilinear Interpolation
%% Matlab 8.6.0.267246 (R2015b)

clear all; close all;

%Reading images
Image1 = double(rgb2gray(imread('flower.jpg', 'jpg')));
Image2 = double(rgb2gray(imread('boccadasse.jpg', 'jpg')));

%Setting dimensions
[DimY, DimX] = size(Image1);
Ratio = DimY / DimX;
ScaledX = 1024;
ScaledY = ScaledX * Ratio;

CurrentFigure = figure(1); set(CurrentFigure, 'Position', [0, 0, ScaledX, ▼18
    18▲ ScaledY]);
CurrentFigure, imagesc(Image1), colormap gray, title('Original image -- gray ▼19
    19▲ scale colormap -- image 1');
saveas(1, 'CV_L01_101', 'pdf');

%Warping Parameters
ImageRotationAngle = pi / 5;
ImageTranslation = [DimY / 7, DimX / 5];

%Setting up the oversized output image matrix
ImageDiagonal = ceil(sqrt(DimY^2 + DimX^2));
ImageOutput = zeros(2 * ImageDiagonal, 2 * ImageDiagonal);
[DimYN, DimXN] = size(ImageOutput);
% TemporaryImage = zeros(DimYN, DimXN);
TemporaryImage = 255 * ones(DimYN, DimXN);
%Calculating coordinates for the warped image
X0 = floor((2 * ImageDiagonal - DimX) / 2);
Y0 = floor((2 * ImageDiagonal - DimY) / 2);
Xe = floor((2 * ImageDiagonal - DimX) / 2) + DimX;
Ye = floor((2 * ImageDiagonal - DimY) / 2) + DimY;
%Target image centerpoint
Yc = floor(DimYN / 2);
Xc = floor(DimXN / 2);

TemporaryImage(Y0:Ye - 1, X0:Xe - 1) = Image1;
CurrentFigure = figure(2); set(CurrentFigure, 'Position', [0, 0, ScaledX, ▼42
    42▲ ScaledY,]);
CurrentFigure, imagesc(TemporaryImage), colormap gray, title('Image placed in ▼43
    43▲ a oversized frame to prevent image data loss -- image 1');
saveas(2, 'CV_L01_102', 'pdf');

%%Inverse transformation
[X,Y] = meshgrid(1:DimXN, 1:DimYN);
```

```
XN = ( (X - ImageTranslation(1) - Xc) * cos(ImageRotationAngle) + (Y -  ▼49
    49▲ ImageTranslation(2) - Yc) * sin(ImageRotationAngle)) + Xc;
YN = (-(X - ImageTranslation(1) - Xc) * sin(ImageRotationAngle) + (Y -  ▼50
    50▲ ImageTranslation(2) - Yc) * cos(ImageRotationAngle)) + Yc;

%Bilinear interpolation
WarpedImage = griddata(X, Y, double(TemporaryImage), XN, YN, 'linear');
CurrentFigure = figure(3); set(CurrentFigure, 'Position', [0, 0, ScaledX,  ▼54
    54▲ ScaledY]);
CurrentFigure, imagesc(WarpedImage), colormap gray, title('Backward warping -- ▼55
    55▲  rotation and translation -- image 1');
saveas(3, 'CV_L01_103', 'pdf');

%Image 2
% Image2 = double(rgb2gray(imread('boccadasse.jpg', 'jpg')));

%Setting dimensions
[DimY, DimX] = size(Image2);
Ratio = DimY / DimX;
ScaledX = 1024;
ScaledY = ScaledX * Ratio;

CurrentFigure = figure(1); set(CurrentFigure, 'Position', [0, 0, ScaledX,  ▼67
    67▲ ScaledY]);
CurrentFigure, imagesc(Image2), colormap gray, title('Original image -- gray  ▼68
    68▲ scale colormap -- image 2');
saveas(1, 'CV_L01_201', 'pdf');

%Warping Parameters
ImageRotationAngle = - pi / 4;
ImageTranslation = [- DimY / 17, DimX / 3];

%Setting up the oversized output image matrix
ImageDiagonal = ceil(sqrt(DimY^2 + DimX^2));
ImageOutput = zeros(2 * ImageDiagonal, 2 * ImageDiagonal);
[DimYN, DimXN] = size(ImageOutput);
% TemporaryImage = zeros(DimYN, DimXN);
TemporaryImage = 255 * ones(DimYN, DimXN);
%Calculating coordinates for the warped image
X0 = floor((2 * ImageDiagonal - DimX) / 2);
Y0 = floor((2 * ImageDiagonal - DimY) / 2);
Xe = floor((2 * ImageDiagonal - DimX) / 2) + DimX;
Ye = floor((2 * ImageDiagonal - DimY) / 2) + DimY;
%Target image centerpoint
Yc = floor(DimYN / 2);
Xc = floor(DimXN / 2);

TemporaryImage(Y0:Ye - 1, X0:Xe - 1) = Image2;
CurrentFigure = figure(2); set(CurrentFigure, 'Position', [0, 0, ScaledX,  ▼91
    91▲ ScaledY,]);
CurrentFigure, imagesc(TemporaryImage), colormap gray, title('Image placed in  ▼92
    92▲ a oversized frame to prevent image data loss -- image 2');
saveas(2, 'CV_L01_202', 'pdf');
```

```
95  %%Inverse transformation
    [X,Y] = meshgrid(1:DimXN, 1:DimYN);

    XN = ( (X - ImageTranslation(1) - Xc) * cos(ImageRotationAngle) + (Y -  ▼98
        98▲ ImageTranslation(2) - Yc) * sin(ImageRotationAngle)) + Xc;
    YN = (-(X - ImageTranslation(1) - Xc) * sin(ImageRotationAngle) + (Y -  ▼99
        99▲ ImageTranslation(2) - Yc) * cos(ImageRotationAngle)) + Yc;
100
    %Bilinear interpolation
    WarpedImage = griddata(X, Y, double(TemporaryImage), XN, YN, 'linear');
    CurrentFigure = figure(3); set(CurrentFigure, 'Position', [0, 0, ScaledX,  ▼103
        103▲ ScaledY]);
    CurrentFigure, imagesc(WarpedImage), colormap gray, title('Backward warping  ▼104
        104▲ -- rotation and translation -- image 2');
105 saveas(3, 'CV_L01_203', 'pdf');
```

The above script is commented in a self explanatory way. The only aspect worth mentioning is the use of an oversized image for the original image within. The reason for this operation is the will to save all data of the processed image, so that no cropping can occur. Also to show the effect of the command *griddata* the background of the oversized image was chosen to be white, so has a value of 255 or FF in hex.

Processed images according to the algorithm presented in the script are presented below:

Figure 1: Original image without modifications.

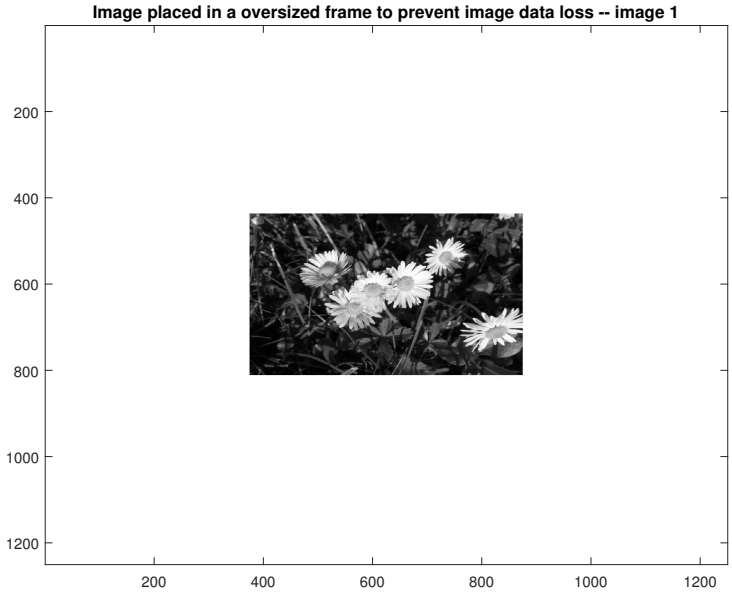

Figure 2: Original image placed in an oversized frame to prevent any data loss through transformations.

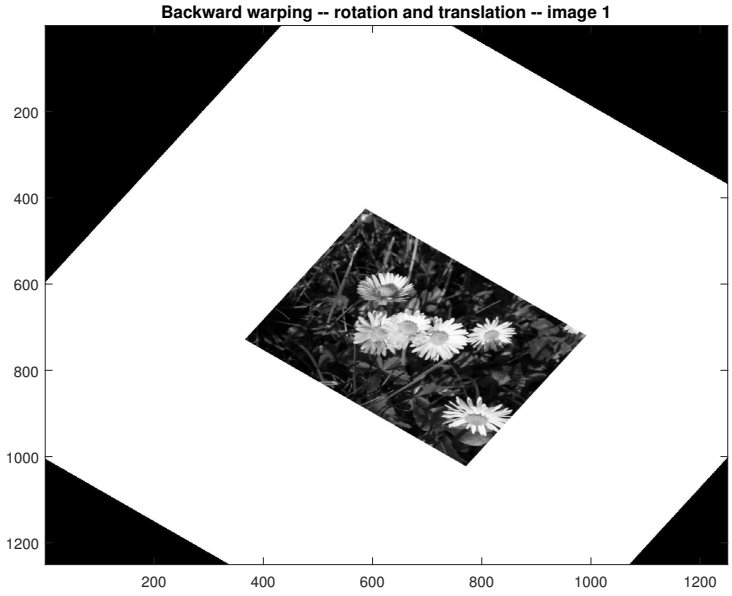

Figure 3: Backwards warping completed according to the presented algorithm on the oversized image.

Figure 4: Original image without modifications.

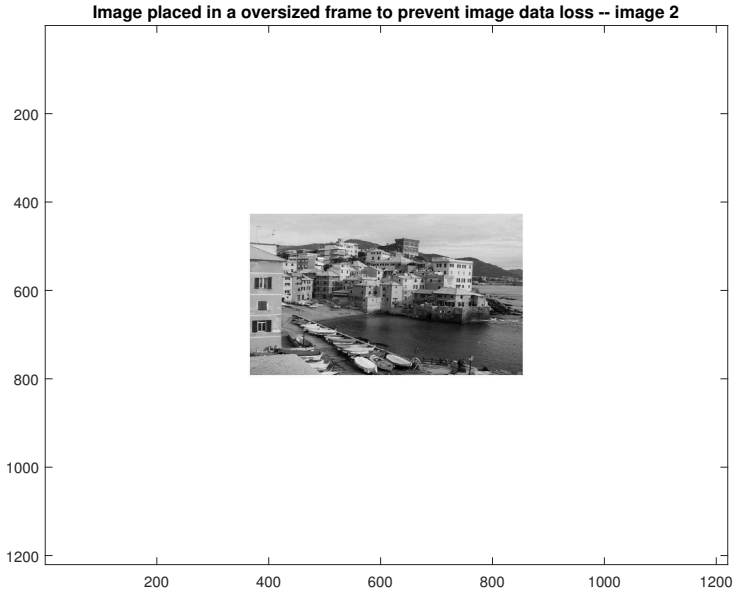

Figure 5: Original image placed in an oversized frame to prevent any data loss through transformations.

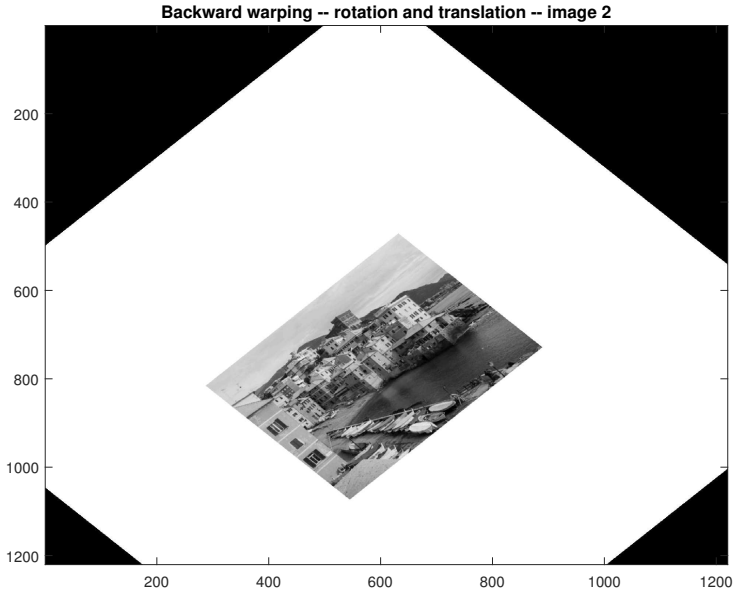

Figure 6: Backwards warping completed according to the presented algorithm on the oversized image.

Conclusions

The algorithm presented in the script and the process of backwards warping in accordance to the equation (2), show how easily transformations of rotation, translation and the not present scaling can be performed. Additionally the aspect of data loss by usage of transformations has been avoided.