Affine Transformation Estimation and Interpolation

Mudit Sethia, Disha Pandey, Yash Rampuria

August 25, 2023

1 Introduction

In this assignment, we explore the estimation of an affine transformation between two images of the Gateway of India, acquired from different viewpoints. We will use manually selected salient feature points to estimate the transformation and then apply nearest neighbor and bilinear interpolation for alignment.

2 MATLAB Code

2.1 Reading and Displaying Images

We start by reading and displaying the two images, 'goi1.jpg' and 'goi2.jpg', in MATLAB.

```
% Read the images
im1 = double(imread("goi1.jpg"));
im2 = double(imread("goi2_downsampled.jpg"));

% Display the images
figure;
imshow(im1/255);
title('Image 1');
figure;
imshow(im2/255);
title('Image 2');
```

2.2 Collecting Corresponding Points

We manually select and store 12 pairs of corresponding points from both images using the `ginput` function.

```
% Collect corresponding points
for i = 1 : 3
    figure(1);
    imshow(im1/255);
    [x1(i), y1(i)] = ginput(1);
    figure(2);
    imshow(im2/255);
    [x2(i), y2(i)] = ginput(1);
end
```

2.3 Affine Transformation Estimation

We estimate the affine transformation matrix using a least-squares framework.

```
len = size(x1);
init = [x1;y1;ones(len)];
fin = [x2;y2;ones(len)];
```

```
% Calculating the Transformation using the least-sqaure framework.

tform = (fin*(init'))/(init*(init'));


% Display the affine transformation matrix
disp('Affine Transformation Matrix:');
disp(tform);
```

2.4 Warping Images

We apply the estimated affine transformation to warp the first image to align it with the second image.

```
% For nearest neighbour interpolation
dim_og = size(im1);
new_im = zeros(dim_og);
inv_tform = inv(tform);
for x = 1:dim_og(1)
    for y = 1:dim_og(2)
        corr_im = round(tform\[x,y,1]');
        n_x = corr_im(1);
        n_y = corr_im(2);
        if(n_x > dim_og(1) || n_x <1)
            continue
        end
        if(n_y > dim_og(2) || n_y <1)
            continue
        end
        new_im(x, y) = im1(n_x,n_y)/255;
    end
end

% Bilinear interpolation
for x = 1:dim_og(1)
    for y = 1:dim_og(2)
        % Apply the inverse transformation to find the corresponding point
        corr_im = inv_tform * [x; y; 1];
        n_x = corr_im(1);
        n_y = corr_im(2);

        % Check if the transformed point is within bounds
        if (n_x > dim_og(1) || n_x < 1 || n_y > dim_og(2) || n_y < 1)
            continue;
        end

        % Calculate the four surrounding pixel coordinates
        x1 = floor(n_x);
        x2 = ceil(n_x);
        y1 = floor(n_y);
        y2 = ceil(n_y);

        % Compute the fractional parts for interpolation
        alpha = n_x - x1;
        beta = n_y - y1;

        % Perform bilinear interpolation
        new_im_bilinear(x, y) = (1 - alpha) * ((1 - beta) * im1(x1, y1) + beta * im1(x1,
```

```
            alpha * ((1 - beta) * im1(x2, y1) + beta * im1(x2, y2));
    end
end
```

2.5 Interpolation

We implement both nearest neighbor and bilinear interpolation to visualize the aligned images.

```
% Display the aligned images
figure;
imshow(new_im);
title('Nearest Neighbour interpolation');
figure;
imshow(new_im_bilinear/255);
title('Bilinear interpolation');

figure;
montage({im1/255, im2/255, new_im, new_im_bilinear/255});
title('Iamge 1, Image 2, Nearest Neighbour interpolation, Bilinear interpolation');
```

3 Results

3.1 Affine Transformation Matrix

The estimated affine transformation matrix is:

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

3.2 Interpolated Images

(a) Nearest Neighbor Interpolation (b) Bilinear Interpolation

Figure 1: Aligned Images with Different Interpolation Methods

4 Discussion

Let's imagine a scenario where, in our initial step, we happened to select 12 points in the first image, and coincidentally, these points lie in a straight line, making them collinear. Collinear points are those that can be connected to form a single straight line. When it comes to estimating an affine transformation matrix, this can have some interesting effects.

4.1 Effect on Affine Transformation Estimation (part e)

Our method for estimating the affine transformation matrix assumes that the selected points are spread out across the image, providing diverse information about how the two images relate to each other. However, when all the points are collinear, they essentially carry the same information, which can lead to a few problems:

1. **No Unique Solution**: When the selected points all fall on a straight line, it's like having too many cooks in the kitchen. There are more unknowns (the elements of the transformation matrix) than there are pieces of unique information to help us figure out those unknowns. This can result in multiple possible solutions or no solution at all.

2. **Tricky Situations**: Collinear points can lead to tricky situations where the matrix we're trying to estimate becomes unstable or unworkable. It's like trying to solve a puzzle with missing pieces. Sometimes, the puzzle just can't be completed.

3. **Questionable Results**: Even if we manage to get a solution in cases like these, it might not be very trustworthy. Collinear points can introduce errors and distortions into our estimated matrix, which means our final aligned images might not look quite right.

4.2 Recommendation

So, what's the takeaway here? Well, if we want a dependable and accurate affine transformation, it's crucial to choose points that are spread out and provide a variety of spatial information. Avoiding collinear points is like ensuring we have a balanced mix of ingredients in a recipe – it makes the cooking process smoother.

In practical terms, it's a good idea to pick feature points that cover different parts of the image, pointing in different directions. This way, we can get a more robust and precise estimation of the affine transformation.

5 Conclusion

In this assignment, we delved into the fascinating world of image transformation, with a particular focus on estimating an affine transformation between two images of the iconic Gateway of India. Our journey involved selecting significant feature points in both images and leveraging them to calculate the affine transformation matrix, a critical step in aligning the images.

We didn't stop there; we also implemented two different interpolation methods: nearest neighbor and bilinear. These methods play a crucial role in ensuring the final aligned images look smooth and visually appealing.

One intriguing aspect we explored was the effect of selecting collinear points. Imagine if all the chosen points aligned in a straight line – collinear points. This scenario introduced complexities in our transformation estimation process, challenging the uniqueness and reliability of the results. It underscored the importance of diversity in point selection, ensuring a robust transformation estimation.

In wrapping up our journey, we've not only gained a deeper understanding of the power of affine transformations in image processing but also appreciated the nuances of point selection and interpolation techniques. These insights will undoubtedly prove valuable as we continue to explore the exciting field of computer vision and image processing.

In conclusion, this assignment has been a captivating exploration into the world of image manipulation, offering us a glimpse into the intricate dance between mathematics and visual artistry.