Programming Assignment 1

Computer vision

1)

To demonstrate that the given affine map is a composition of a shear, followed by a rotation, then by a magnification (scaling), and finally a translation, let's break down the transformation step by step and calculate the corresponding matrices.

To demonstrate that the given affine map is a composition of a shear, followed by a rotation, then by a magnification (scaling), and finally a translation, let's break down the transformation step by step and calculate the corresponding matrices.

The given affine map is represented as follows:

| x1 | | 1 1 | | x1 |

| x2 | + | 1 3 | \* | x2 |

+ | 4 0 |

Shear Transformation:

The shear transformation can be extracted from the 2x2 matrix on the top right:

Shear Matrix (S) = | 1 1 |

| 1 3 |

Rotation Transformation:

To isolate the rotation component, we need to remove the shear part. To do this, we can find the inverse of the shear matrix (S^(-1)):

S^(-1) = | 3 -1 |

| -1 1 |

Now, we can find the rotation matrix (R) by multiplying S^(-1) with the original matrix:

| 1 1 | | 3 -1 | | 3 2 |

| 1 3 | \* | -1 1 | = | 2 4 |

To find the rotation angle (theta), we can use the formula:

theta = atan2(2, 3) ≈ 33.69 degrees

So, the rotation matrix (R) is a rotation of approximately 33.69 degrees.

Magnification (Scaling) Transformation:

We have already found the rotation component. Now, to isolate the magnification part, we need to remove the rotation part. To do this, we can find the inverse of the rotation matrix (R^(-1)):

R^(-1) = | cos(-33.69) -sin(-33.69) |

| sin (-33.69) cos(-33.69) |

Multiplying R^(-1) with the matrix containing the shear component and translation gives us the magnification matrix (M):

| 3 2 | | cos (-33.69) -sin (-33.69) | | 1.5 0 |

| 2 4 | \* | sin (-33.69) cos(-33.69) | = | 0 3 |

So, the magnification matrix (M) scales by a factor of 1.5 in the x-direction and 3 in the y-direction.

Translation Transformation:

The translation part is already given:

Translation Matrix (T) = | 4 |

| 0 |

Now that we've found the individual transformation matrices, let's compose them:

Affine Matrix (A) = T \* M \* R \* S

To apply this transformation to an image using Python, you can use the code and explanations provided in the previous response, adapting the transformation matrices accordingly.

Input image

A black square with white border

Description automatically generated A screenshot of a phone

Description automatically generated

Rescaled images

A person smiling with a beard

Description automatically generated with medium confidence

Output deformed images

A black and white background

Description automatically generated A person standing on a white surface

Description automatically generated A white rectangular object with a black background

Description automatically generated A black and white surface

Description automatically generated

Interpolated images

A person with dark hair

Description automatically generated A person's face with long hair

Description automatically generated A person's face in a mirror

Description automatically generated

A person's face in a mirror

Description automatically generated

2

a)

In a finite projective camera, the camera center (also known as the optical center or the pinhole) is located at a finite point in the world coordinate system. To find the coordinates of the camera center in the world coordinate system, we can use the concept of the camera matrix and the equation for a finite projective camera.

The camera matrix, denoted as P, can be expressed as:

P = K [R | t]

Where:

K is the camera intrinsic matrix.

[R | t] is the extrinsic matrix, combining rotation (R) and translation (t).

In this case, we can use the information provided for the four pixels to estimate the camera matrix and subsequently extract the camera center.

Let's denote the four pixel coordinates as follows:

Pixel 1: (5, 100, 1)T for a world point at infinity along the world X direction.

Pixel 2: (400, 300, 1)T for a world point at infinity along the world Y direction.

Pixel 3: (500, 490, 1)T for a world point at infinity along the world Z direction.

Pixel 4: (20, 20, 5)T for the world origin.

Now, we can formulate the equations for these pixel coordinates in terms of the camera matrix P:

For Pixel 1:

Pixel 1 = P \* World Point at Infinity along X direction

(5, 100, 1)T = P \* (X\_infinity, Y\_infinity, Z\_infinity, 1)T

For Pixel 2:

Pixel 2 = P \* World Point at Infinity along Y direction

(400, 300, 1)T = P \* (X\_infinity, Y\_infinity, Z\_infinity, 1)T

For Pixel 3:

Pixel 3 = P \* World Point at Infinity along Z direction

(500, 490, 1)T = P \* (X\_infinity, Y\_infinity, Z\_infinity, 1)T

For Pixel 4:

Pixel 4 = P \* World Origin

(20, 20, 5)T = P \* (0, 0, 0, 1)T

These equations represent a system of linear equations that can be solved for the elements of the camera matrix P. Once we have the camera matrix P, we can extract the camera center from the last column of the extrinsic matrix [R | t]. The camera center will be a finite point in the world coordinate system.

b)

To reproject an old 2D image through the given software camera, you can follow these steps:

Calculate the Camera Matrix (P):

Use the information provided to calculate the camera matrix P. You have the pixel coordinates (in homogeneous representation) for the image points corresponding to world points at infinity along the X, Y, and Z directions, as well as the pixel coordinates of the image point corresponding to the world origin. These points will help you determine the elements of the camera matrix.

Place the Old 2D Image in Front of the Camera:

Assume that the old 2D image is placed at a distance of two focal lengths from the camera center, with the principal axis passing through the center of the old image. This means that the old image is placed parallel to the sensor plane and is positioned such that it intersects the optical axis at a distance of two focal lengths from the camera center. This placement ensures that the old image is in focus when projected onto the sensor.

Create a 3D Model of the Old Image:

Represent the old 2D image as a 3D planar surface in 3D space. You can choose the dimensions and orientation of this planar surface as needed, but make sure it aligns with the placement described in step 2.

Transform the 3D Model to Homogeneous Coordinates:

Transform the 3D coordinates of the points on the planar surface to homogeneous coordinates. The homogeneous coordinates should have a 1 in the fourth coordinate to represent points in 3D space.

Project 3D Points to 2D Image Space:

Use the camera matrix P to project the 3D points of the old image onto the 2D image plane. This can be done by multiplying each 3D point in homogeneous coordinates by the camera matrix P. The resulting 2D points on the image plane represent the reprojected image.

Display the Reprojected Image:

After obtaining the reprojected 2D points, you can display the reprojected image by mapping the pixel coordinates of the reprojected points to the corresponding pixel values in the old 2D image. This will create a visualization of what the old image would look like when viewed through the software camera.

Finalize and Visualize:

Display the reprojected image alongside the original old 2D image to compare the two. Ensure that the reprojected image is correctly aligned and scaled according to your placement in step 2.

Remember to adjust the parameters and dimensions of the planar surface representing the old image to achieve the desired reprojection. The key is to use the camera matrix P to project the 3D points onto the image plane accurately.

3)

Observations on Image Reconstructions:

Magnitude-only Reconstructions:

1. Magnitude-only reconstructions of images A and B exhibit a representation with texture preserved.
2. Structural details are not well-preserved in these reconstructions which encode information about the intensity variations in the original images.

Phase-only Reconstructions:

* Phase-only reconstructions of images A and B lack structural details and appear as uniform gray or noise-like patterns.

A screenshot of a computer screen

Description automatically generated