CSC320

Introduction to Visual Computing

Sinan Li

2024

CONTENTS

Part I

Foundations

CHAPTER

1 IMAGE GEOMETRY

1.1 Linear 2D Transforms

1.1.1 Homogeneous 2D Point Coordinate

Basic Notational Conventions

- Column-Vector Representation

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

- Row-Vector Representation

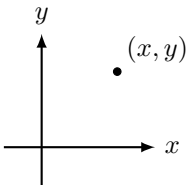
$$\begin{bmatrix} x & y \end{bmatrix}$$

- Matrix Transpose Operation

$$\begin{bmatrix} x \\ y \end{bmatrix}^\top = \begin{bmatrix} x & y \end{bmatrix}$$

Homogeneous Coordinates

In the standard Euclidean coordinate system, a point is represented by a pair of coordinates (x, y), where x and y are the coordinates of the point along the x and y axes, respectively.



In homogeneous coordinates, a point is represented by a triple of coordinates (x, y, w), where x and y are the coordinates of the point along the x and y axes, respectively, and w is a scaling factor that is normally set to 1. The homogeneous coordinates of a point are not unique, since any multiple of the coordinates (x, y, w) represents the same point. For example, $(2, 3, 1)$ and $(4, 6, 2)$ represent the same point. The homogeneous coordinates of a point are unique only up to a scaling factor.

Remark

Two vector of homogeneous coordinates (x, y, w) and (x', y', w') represent the same point if and only if there exists a non-zero scalar λ such that

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \lambda \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}$$

Example. The homogeneous coordinates of the point $(2, 3)$ are $(2, 3, 1)$, $(4, 6, 2)$, $(6, 9, 3)$, etc. ♢

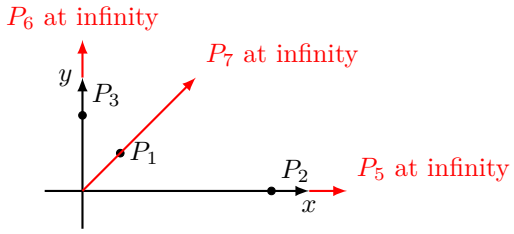
Homogeneous to Euclidean Coordinat

To convert a homogeneous coordinate (x, y, w) to a Euclidean coordinate (x', y'), we divide the first two coordinates of the homogeneous coordinate by the third coordinate, i.e.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

Example. Plot the following points in the Euclidean plane:

$$\begin{array}{lll} P_1 = (2, 2, 2) & P_2 = (10, 0, 2) & P_3 = (0, 8, 4) \\ P_4 = (1, 0, 0.01) & P_5 = (1, 0, 0) & P_6 = (0, 1, 0) \\ P_7 = (1, 1, 0) & & \end{array}$$



Note that P_5, P_6, and P_7 are at infinity. This is a very important property of homogeneous coordinates. ♢

Remark

Points infinitely far away from the origin in the Euclidean plane have a finite representation in homogeneous coordinates (i.e. $(x, y, 0)$). These points are sometimes called **ideal points**.

1.1.2 Homogeneous 2D Line Coordinates

Homogeneous 2D Line Coordinates

Remark

The general equation of a line in the Euclidean plane is

$$ax + by + c = 0$$

where a, b, and c are real numbers and a and b are not both zero.
In homogeneous coordinates, using the matrix notation, the general equation of a line is

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0$$

Or, equivalently,

$$\ell^\top p = 0$$

where ℓ is the vector holding line coefficients and p is the vector holding point coordinates.

Definition 1.1.1

The **homogeneous coordinates of a line** with the equation

$$ax + by + c = 0$$

is the vector

$$\ell = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Example. The homogeneous coordinates of the line $y = x$ is

$$\ell = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

Line Passing Through Two Points

In this case, ℓ must satisfy the following equations:

$$\ell^\top P_1 = 0 \qquad \ell^\top P_2 = 0$$

where p_1 and p_2 are the homogeneous coordinates of the two points.

We can take the cross product of p_1 and p_2 to obtain ℓ:

$$\ell = P_1 \times P_2$$

> **Remark**
>
> To compute the cross product of two vectors, we can use the following two methods.
>
> $$\ell = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \times \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$
>
> As matrix multiplication:
>
> $$P_1 \times P_2 = \begin{bmatrix} 0 & -z_1 & y_1 \\ z_1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} P_2$$
>
> As determinant:
>
> $$P_1 \times P_2 = \begin{vmatrix} i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}$$

Intersection of Two Lines

In this case, p must satisfy the following equations:

$$\ell_1^\top p = 0 \qquad \ell_2^\top p = 0$$

where ℓ_1 and ℓ_2 are the homogeneous coordinates of the two lines.

We can take the cross product of ℓ_1 and ℓ_2 to obtain p:

$$p = \ell_1 \times \ell_2$$

> **Remark**
>
> For parallel lines, we cannot compute its standard Euclidean intersection point. Instead, we can compute the homogeneous intersection point, which will be at infinity.

1.1.3 Affine 2D Transformations

- **Identity**

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \cong \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Stretching** (along x)

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \cong \begin{bmatrix} sx \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  Stretching (along y)

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \cong \begin{bmatrix} x \\ sy \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Shearing** (along y)

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \cong \begin{bmatrix} x \\ hx + y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Rotation** (about the origin bt angle ϕ)

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \cong \begin{bmatrix} (\cos\phi)x - (\sin\phi)y \\ (\sin\phi)x + (\cos\phi)y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Translation** (along x)

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \cong \begin{bmatrix} x + d \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  Translation (along y)

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \cong \begin{bmatrix} x \\ y + d \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & d \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

We can combine these transformations to form more complex transformations. For example, we can combine a rotation and a translation to form a **rotation about a point** transformation.

Definition 1.1.2 Affine Transformation (Intuitive)

An **affine transformation** is any combination of scaling, shearing, rotation, and translation.

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

where $g \neq 0$ is a scaling factor that does not affect the transformation.

Remark

An affine transformation is a transformation that preserves parallelism and ratios of distances along parallel lines.

In the original image, two parallel lines would intersect at some infinity $(x, y, 0)$. After an affine transformation, the two lines would still intersect at some infinity $(x', y', 0)$.

$$\begin{bmatrix} x' \\ y' \\ 0 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & g \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

Definition 1.1.3 Affine Transformation

An affine transformation is any invertible 3×3 matrix that preserves the points at infinity.

But what if the last row of the matrix is not $(0, 0, 1)$?

1.1.4 Homographies

Homographies are also called **projective transformations** or **perspective transformations**. They still preserve linearity, but they do not preserve parallelism.

Definition 1.1.4

A **homography** is any 2D transformation of homogeneous coordinates that is represented by an invertible 3×3 matrix.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ l & m & g \end{bmatrix}$$

All homographies are invertible. This is because Homographies map unique points to unique points.

Remark

Lines remain straight after a homography.
Let $\ell^T p = 0$ be a line. It suffices to show that all points on this line are mapped to points on a line.

Example. A naive image warping algorithm is forward mapping.

```
input: src_image, H
output: dest_image

for c=1 to num_columns                // cycle over all source pixels
   for r=1 to num_rows
        x, y = pixel_xy(r,c)          // get the source pixel's (x,y) coordinates
        p = homogeneous_coords(x, y) // convert to vector homogeneous coords
        p_prime = H*p                 // apply homography
        x_prime, y_prime = euclidean_coords(p_prime)  // convert to Euclidean coords
        r_prime, c_prime = pixel_rc(x_prime, y_prime) // calculate the dest pixel
```

```
dest_image(r_prime, c_prime) = src_image(r,c) // copy pixel color
```

This algorithm has some problems:

- Magnification causes "holes" in destination image
- Minification causes overwriting of pixel contents

♢

Backward-Mapping Algorithm

This is also known as the **inverse mapping algorithm**. This algorithm does not create "holes" in the destination image.

```
input: src_image, H
output: dest_image

for c_prime=1 to num_columns                          // cycle over all dest px
   for r_prime=1 to num_rows
       x_prime, y_prime = pixel_xy(r_prime,c_prime)   // get dest px's (x,y)
       p_prime = homogeneous_coords(x_prime, y_prime) // convert to homo. coords
       p = inverse(H)*p_prime                         // apply inverse homography
       x, y = euclidean_coords(p)                     // convert to Euclidean
       r, c = pixel_rc(x, y)                          // calculate source px
       dest_image(r_prime, c_prime) = src_image(r,c)  // copy px color
```

1.2 Perspective Viewing

1.2.1 Aperture

Without an aperture, every sensor receives every light, so the image is very blurry. We need an aperture to control the amount of light that enters the camera. The aperture acts as a diaphragm, allowing us to regulate the size of the opening through which light passes. By adjusting the aperture size, we can control the amount of light that reaches the sensor, resulting in a properly exposed image.

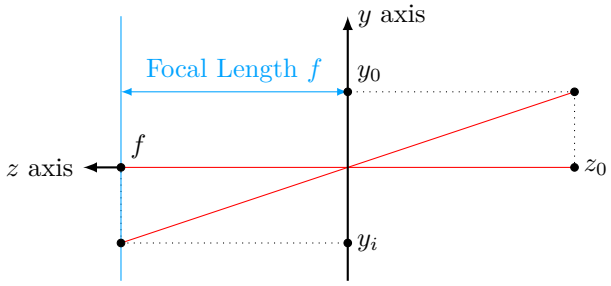
The image we get is up-side-down, and as we increase the aperture size, the image becomes more blurry.

1.2.2 Perspective Projection

Geometry of Perspective Projection

Definition 1.2.1 Focal Length

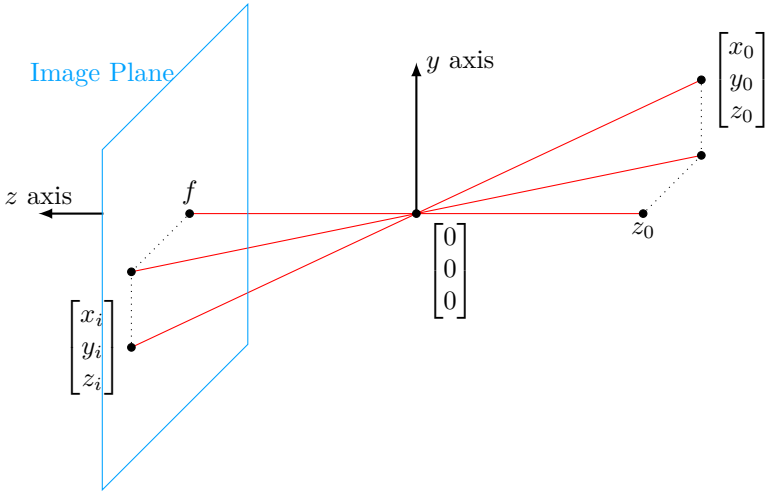
The **focal length** of a lens is the distance from the lens to the point where light rays converge to a point.



From similar triangles, we get

$$\frac{y_i}{y_0} = \frac{f}{z_0} \implies y_i = \frac{f}{z_0} \cdot y_0$$

The same applies to 3-dimensional points.



We have

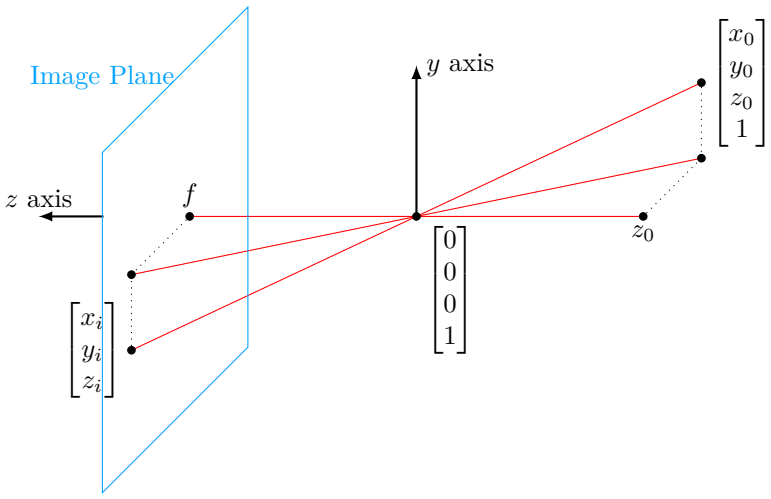
$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \frac{f}{z_0} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

Interpretation of homogeneous equality tells us that "all 3D points have the same projection",

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \cong \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

In other words, $\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ represents a ray through the origin.

Homogeneous 3D Coordinates



Similar to 2D homogeneous coordinates, we can represent 3D points in homogeneous coordinates

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \cong \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

To convert from the homogeneous coordinates to the Euclidean coordinates, we divide the first three coordinates by the fourth coordinate, i.e.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} / w_0$$

Remark

The point

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 0 \end{bmatrix}$$

is at infinity in 3D space, in the direction of the vector $\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$.

Homogeneous coordinates provide us a convenient way to express the projective of a 3D point into a 2D image,

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{z_0} \cdot x_0 \\ \frac{f}{z_0} \cdot y_0 \\ 1 \end{bmatrix} \cong \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

where $\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$ is the homogeneous coordinates of the 2D projection of the 3D point $\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$, and the only difference between the 2D and 3D projection matrices is the scaling factor $\frac{f}{z_0}$.

1.3 Alignment and Stitching

1.3.1 Estimating Homography from Point Correspondence

Claim. Given sufficiently many point correspondences we can compute tha homogeneous transformation H that aligns two images. We need at least 4 correspondences for a unique solution.

Given a point correspondence $\begin{bmatrix} {x_i}' \\ {y_i}' \\ 1 \end{bmatrix} \cong \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$, we can write the following equation

$$\begin{bmatrix} {x_i}' \\ {y_i}' \\ 1 \end{bmatrix} \cong H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \iff \begin{bmatrix} {x_i}' \\ {y_i}' \\ 1 \end{bmatrix} \cong \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

which allows us to establish a system of linear equations

$$\begin{bmatrix} {x_i}' \\ {y_i}' \\ 1 \end{bmatrix} \cong \begin{bmatrix} ax_i + by_i + c \\ dx_i + ey_i + f \\ gx_i + hy_i + 1 \end{bmatrix} \iff \begin{bmatrix} {x_i}' \\ {y_i}' \\ 1 \end{bmatrix} \cong \begin{bmatrix} ax_i + by_i + c/gx_i + hy_i + 1 \\ dx_i + ey_i + f/gx_i + hy_i + 1 \\ 1 \end{bmatrix}$$

That is, one correspondence gives us two linear equations. We have 8 unknowns, so we need 4 correspondences to solve the system of linear equations.

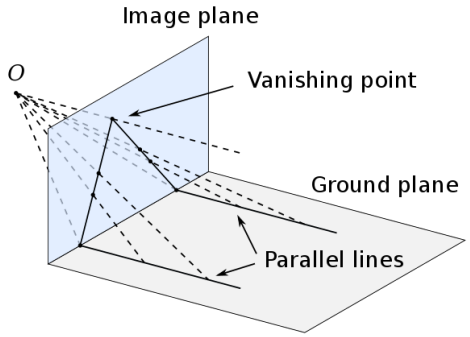
1.4 3D Metrology

1.4.1 3D Reconstruction from 2D Images

1.4.2 Vanishing Points

Definition 1.4.1

A **vanishing point** is a point on the image plane of a perspective rendering where the two-dimensional perspective projections of mutually parallel lines in three-dimensional space appear to converge.



Every direction in space has a unique vanishing point. These vanishing points are the intersection of the image plane with the projective space, and they all lie on the horizon line.

CHAPTER

2 IMAGE FILTERING

2.1 Linear Shift-Invariant Filters

2.1.1 Definition

A filter transforms one signal into another. Filters are used to describe image formation (lens burring, sensor noise, etc.), as well as to implement operations on images (edge detection, noise reduction, etc.).

Definition 2.1.1 Linear Transformation

A transformation T is **linear** if and only if it satisfies

$$T[a_1 f_1(x) + a_2 f_2(x)] = a_1 T[f_1(x)] + a_2 T[f_2(x)]$$

for any $a_1, a_2 \in \mathbb{R}$ and continuous functions f_1, f_2.

Every linear transformation between functions f, g can be expressed as an integral of the form

$$g(x) = \int_{-\infty}^{\infty} h(x, \tau) f(\tau)\, d\tau,$$

where the $h(x, \tau)$ is the contribution of position τ of the input to position x of the output.

Definition 2.1.2 Shift-Invariant Filter

A filter $h(x, \tau)$ is **shift-invariant** if and only if shifted inputs produce identical but shifted outputs.

Imagine we have the shifted input

$$f'(x') = f(x - x_0)$$

and the shifted output

$$g'(x') = g(x - x_0).$$

The **shift invariance property** states that

$$T[f(x - x_0)] = g(x - x_0)$$

2.1.2 The Impulse Function

CHAPTER 3

MODEL FITTING

CHAPTER

COLOR IMAGING AND DISPLAYING 4

Part II

Image Representation for CV

CHAPTER

5 CONTINUOUS

CHAPTER

6 VECTOR-BASED

CHAPTER 7

MULTI-SCALE

Part III

Appendices

BIBLIOGRAPHY