Computer Vision

Dr. Syed Faisal Bukhari

Associate Professor

Department of Data Science

Faculty of Computing and Information Technology

University of the Punjab

Reference

These notes are based on

- Dr. Matthew N. Dailey's course: AT70.20: Machine Vision for Robotics and HCI

Geometric Relationships in $\mathbb{P}^3$: Planes, Points, and Lines

- In $\mathbb{P}^3$, there are several geometric relationships between **planes**, **points**, and **lines**. Some key examples include:

1. A **plane** can be uniquely defined by the **join of three points**, or by the **join of a point and a line**, where the points are in general position (i.e., **the points are not collinear**, and in the second case, the **points are not incident on the line**).

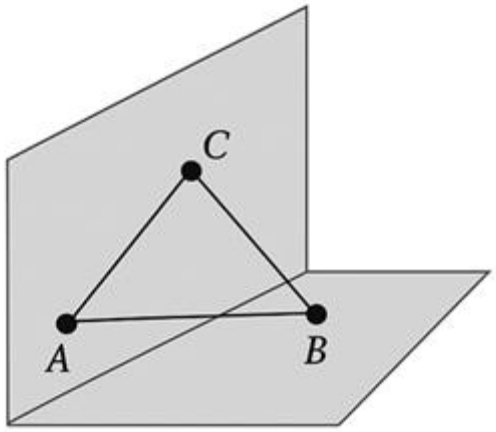
2. Two **distinct planes** always intersect in a **unique line**.

3. **Three distinct planes** intersect in a **unique point**.

- **Three points define a plane.**
- **Three planes define a point.**

Plane: Join of Three Points

The diagram shows **three non-collinear points**, **A**, **B**, and **C**. When these points are not all on the same straight line (non-collinear), they uniquely **define a plane in 3D space**. **This plane** is determined by the **join of these three points**. **The line** connecting these points lies on this plane, and the plane is **uniquely defined by the three points**, as long as they are **not collinear**.



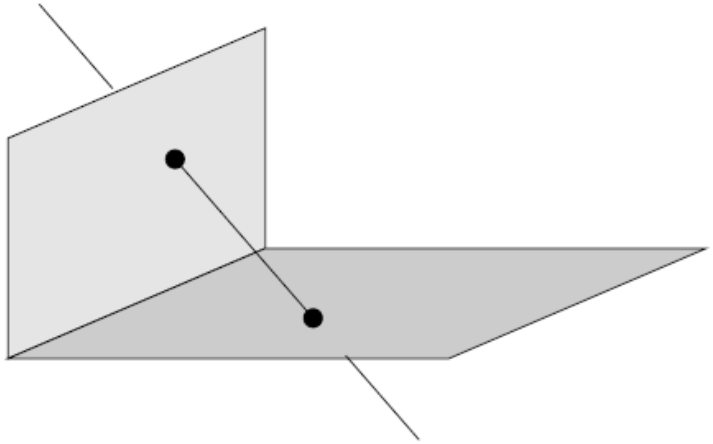
Join of three points

Lines in 3D

- A line is defined by the *join* of **two points** or the **intersection of two planes**. Lines have **4 degrees of freedom** in **3-space**.

- A convincing way to **count** these **degrees of freedom** is to think of a line as defined by its **intersection** with **two orthogonal planes**, as in figure below.

- The **point of intersection** on each plane is specified by **two parameters**, producing a total of **4 degrees of freedom** for the line.



Lines in 3D

- **Representing lines in 3D** is challenging because the natural representation for an object with **4 degrees of freedom** would require a **homogeneous 5-vector**.

- The issue is that a **homogeneous 5-vector** is not easily integrated into mathematical expressions that **involve 4-vectors used for points and planes**.

- To address this challenge, **several line representations** have been proposed, each **varying in mathematical complexity**.

Lines in 3D

- We've previously discussed how **3D lines** can be associated with a **pencil of planes**, each containing two points. This is one way to represent a line mathematically.

- However, there are **numerous ways to represent lines in 3D** space, as **there is no single, obvious method for doing so**.

- **Note**: Planes are typically described as a **vector of coefficients**, where the **dot product between the plane and any point on the plane** equals zero.

Lines in 3D

- When dealing with **something** simple, we typically **don't need alternative representations**.
- However, for **3D lines**, there is **no straightforward representation**.
- We are discussing lines that are embedded in 3D space.
- **Research** has introduced **several methods for representing lines**, each offering different advantages depending on the specific application.

Lines in $\mathbb{P}^3$: A line may be specified by its points of intersection with two orthogonal planes.

- One way to do describe a line in 3-space is by their **intersection** with **two given planes**. See Figure 3.1

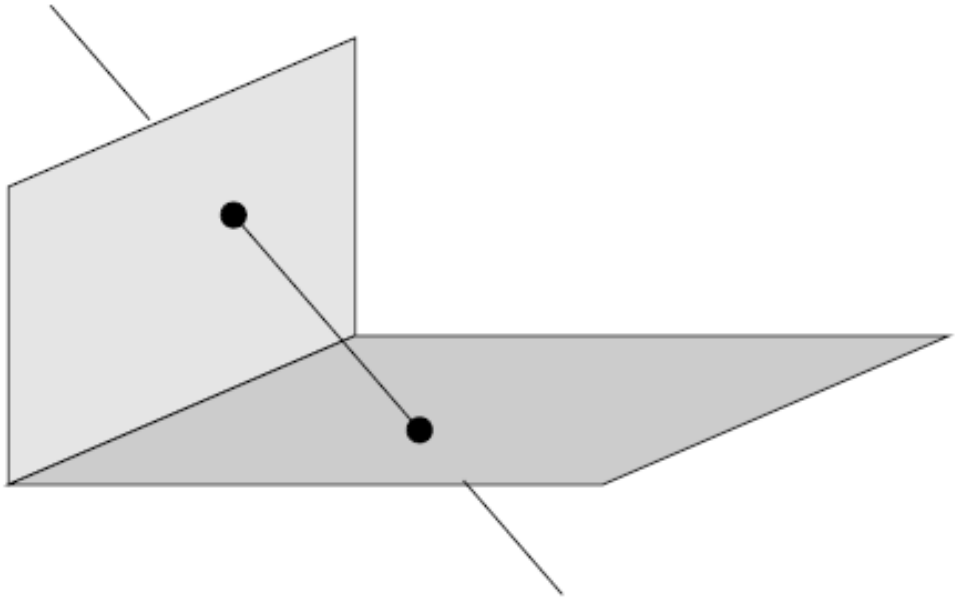


Figure 3.1. A line may be specified by its **points of intersection** with two **orthogonal planes**. Each intersection point has **2 degrees of freedom**, which demonstrates that a line in $\mathbb{P}^3$ has a total of **4 degrees of freedom**.

Lines in $\mathbb{P}^3$: A line may be specified by its points of intersection with two orthogonal planes

- Suppose we have two arbitrary orthognal planes. If we define the **intersection** of a **line** with these **two planes**, we describe the **line in 3D space** through these intersections.

- What does it mean when we say **a line** is represented in **3D** by the **intersection of two planes**?
  - Let $\boldsymbol{\pi}_1$ be the **XY-plane** and $\boldsymbol{\pi}_2$ be the **YZ-plane**. We describe the line L as the intersection of L with both $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$. This representation is not typically used as it is not a practical form for describing a line.

L = $\begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \end{bmatrix}$ X_1, Y_1 belongs to XY-plane and X_2, Y_2 belongs to YZ-plane

Note: Planes should not be **parallel**.

Planes should be **orthogonal**.

Advantages of Representing a 3D Line via Intersection Points of Two Orthogonal Planes

- The one main advantage of this representation is that it is a **minimal representation**.

- We cannot come up with a **smaller vector** that uniquely defines a line.

- We're always looking for at **least four things**, and we can't do less than that.

Disadvantages of Representing a 3D Line via Intersection Points of Two Orthogonal Planes

- If a **line lies within a plane**, the intersection of **the line** and **the plane** is just the entire **line itself**.

- This leads to a **degenerate configuration** where we don't gain any new information by representing the line this way. We are unable to uniquely define the line by doing so.

- **For example:** Imagine a **table representing a plane** and placing a **pen (representing the line)** on it. If the line is very close to or actually lies on **the plane**, we can't distinguish between **the line and the plane.**

- Therefore, no arbitrary point will help us figure out the exact line. This method doesn't effectively represent a line. While it is a **minimal representation**, it doesn't capture all types of lines.

An Application of 3D lines

- Lines are extremely useful. For example, if **we capture an image of PUCIT Building B**, we can **extract the 3D lines** that represent the building's shape and then plot them using OpenGL.

- We also have a **Computer-Aided Design (CAD)** model of PUCIT Building B.

- However, working with **lines mathematically can be quite challenging**

Applications of 3D Lines: Problem

- In a literature review, it is observed that **many researchers prefer using point clouds for 3D reconstruction** rather than relying on **3D lines.**

- **Points are straightforward:** For example, on the left, you have a point, and on the right, you have another point. By taking **the intersection of the rays that pass through these two points**, you can **determine the 3D location of the point**.

- Researchers have been working on **3D point clouds for years** to achieve **3D reconstruction**.

- In upcoming lectures, we will discuss the process of 3D reconstruction using point clouds.

2. Homogenous Point Representation (Null space and span representation)

- This representation builds on the intuitive geometric notion that a **line is a pencil** (one-parameter family) **of collinear points**, and is defined by **any two of these points**.

- Similarly, a **line** is **the axis of a pencil of planes** and is defined by the **intersection** of **any two planes** from the pencil.

- In both cases, the actual **points or planes** are not important (in fact, **two points** have **6 degrees of freedom** and are represented by **two points** in a **3D space** can be represented **using 4-vectors**– **far too many parameters**).

2. Homogenous Point Representation (Null space and span representation)

Suppose **A,B** are two (non-coincident) and **homogeneous space points**. Then the **line joining these points** is represented by the span of the **row space of the 2 × 4 matrix** W composed of $\boldsymbol{A^T}$ and $\boldsymbol{B^T}$ as rows:

$$W = \begin{bmatrix} \vec{A}^T \\ \vec{B}^T \end{bmatrix}_{2\times 4}$$

- As we discussed earlier, constructing a $\boldsymbol{2 \times 4}$ **matrix** using **two points** on a line provides a **representation of that line**. Any vector (i.e., a plane) that satisfies the equation **W · plane = 0** defines a plane that contains the line.

2. Homogenous Point Representation (Null space and span representation)

- So, the intersection of **these planes** is actually the line of interest. In which case
  - The span of W^T is the **pencil of points** $\alpha\vec{A} + \mu\vec{B}$ on the line.
  - The span of the **2-dimensional right null-space** of W is the **pencil of planes** with the line as the axis.
- It is evident that two other points, $\boldsymbol{A'^T}$ and $\boldsymbol{B'^T}$, on the line will generate a matrix W' with the same span as W, so that the span, and hence the representation, is **independent of the particular points** used to define it.

3. Dual Homogenous Plane Representation:

There is a dual representation of a **line** as the intersection of **two planes** $\vec{P}$ and $\vec{Q}$

$$W^* = \begin{bmatrix} \vec{P}^T \\ \vec{Q}^T \end{bmatrix}_{2\times 4}$$

This is a dual representation of a **3D line**. We saw we could represent a line by stacking **up two points** that contain that line. **Points** and **planes** are dual to each other in $\mathbb{P}^3$**, i.e., 3D projective geometry.**

If we can do something by stacking up the representation of **two points,** then we should be able to do something by stacking up the representation of **two planes**. It turns out that's also true.

If you take two planes $\vec{P}$ and $\vec{Q}$ then their **intersection** is the **line of interest**.

Properties:

1) The span of W^{*T} is the pencil of planes $\alpha'\vec{P} + \mu'\vec{Q}$ with the line as an axis.
2) The span of the 2-dimensional null-space of W^* is the **pencil of points on the line**.

4. Plücker matrix representation of a 3D line

- **Plücker** representation of **a 3D line** is the most elegant representation of a line. Although it is not easy to work with it. We represent a 3D line as a **4 vector**. Previously, we have also represented a line as a **2 × 4 matrix**.

- **Plucker** representation goes even further: it expresses a line using **a 4×4 matrix**. This involves **16 elements** to describe a line that fundamentally has only **4 degrees of freedom.**

We represent a line through $\vec{A}$ and $\vec{B}$ by a **4×4 skew symmetric homogenous matrix L** defined by

$l_{ij} = A_i B_j - B_i A_j$ (from co-efficient point of view)

$L = \boldsymbol{AB^T - BA^T}$ (or equivalently in vector notation)

L is the **Plucker matrix representation** of a **line**.

4. Plücker matrix representation of a 3D line

Recall: A skew-symmetric matrix is a matrix A such that
A^T = - A

$l_{ij} = A_i B_j - B_i A_j$

- l_{ij} represents the **ith row** and the **jth column** of this matrix. The points are A_i and B_j and B_i and A_j.

- We get two points on a **line A and B**; we need to multiply the coefficients of these two points in order to get this matrix.

4. Plücker matrix representation of a 3D line

- L = $\vec{A}_{4\times1}\ \vec{B}^{T}_{1\times4}$ - $\vec{B}_{4\times1}\ A^{T}_{1\times4}$

- We just multiply the vector $\vec{A}$ with $\vec{B}^{\,T}$. It is a **homogenous 4×1 vector** that describe the point A.

- $\vec{B}^{\,T}$ is 1×4 **row vector** representing a homogenous point B.

- Likewise, we multiply $\vec{B}$ with $\vec{A}^{T}$. After subtracting $\vec{A}B^{\,T}$ and $\vec{B}\vec{A}^{\,T}$ matrices, we get a 4×4 matrix L. This matrix is **skew symmetric**. What does it mean?

- Transpose of the matrix is equal to the negation of the matrix.

4. Plücker matrix representation of a 3D line

- How skew-symmetric matrix looks like?
- First think about the diagonal. What number is the negation of itself?

⇒ Zero

$$\begin{bmatrix} 0 & -a & -b & -c \\ a & 0 & -d & -e \\ b & d & 0 & -f \\ c & e & f & 0 \end{bmatrix}_{4\times 4}$$

- A **4×4 skew-symmetric matrix** has **6 degrees of freedom, i.e., a, b, c, d, e, and f**. But if it is homogeneous, then we lose one degree of freedom. So, it has **5 degrees of freedom** in order to represent an object that has **4 degrees of freedom**. In some sense, it is closer to reality. We can do a **statistical estimation** of the **Plücker representation of lines.**

Properties of Plücker matrix representation:

1. The rank of L is 2 due to its construction. We construct this matrix L using two vectors $\vec{A}$ and $\vec{B}$ i.e.,

$$L = \vec{A}_{4\times1}\ \vec{B}^{T}_{1\times4} - \vec{B}_{4\times1}\ A^{T}_{1\times4}$$

It is difficult to get a rank more than 2 using these two vectors. The 2-dimensional null space of this vector is spanned by the pencil of planes that has a line as an axis (in fact $LW^{*T} = 0$, with 0 a 4×2 null-matrix). So, the null space of this matrix is related to the line.

2.It has only **4 degrees of freedom**. Previously, we said that it **has five degrees of freedom** i.e., any skew symmetric 4×4 has 5 degrees of freedom. We lose **another degree of freedom by** enforcing **rank 2 constraint**. So, set of all $4 \times 4$with rank 2 constraint has **only 4 degrees of freedom**.

Properties of Plücker matrix representation:

3.This matrix L = $\vec{A}_{4\times1}\ \vec{B}^{T}_{1\times4}$ - $\vec{B}_{4\times1}\ A^{T}_{1\times4}$ is similar to constructing a line in 2D i.e., $l = \vec{x} \times \vec{y}$ of $\mathbb{P}^{2}$. We take two point on a line and take their cross product

4. L is independent of the point $\vec{A}$ and $\vec{B}$. It means if choose any other point $\vec{C}$ and $\vec{D}$ on the line then we do the same construct of L and we get the same L.

Let C = $\boldsymbol{A + \mu B}$ then the resulting matrix is

$\hat{\mathrm{L}} = \mathbf{A C^{T} - C A^{T} = A\,(\,A + \mu B)^{T} - (\,A + \mu B)\,A^{T}}$

$= \mathbf{A\,(A^{T} + \mu B^{T}) - A\,A^{T} - \mu B\,A^{T}}$

$= \mathbf{A\,A^{T} + \mu A\,B^{T} - A\,A^{T} - \mu B\,A^{T}}$

$= \mathbf{\mu A\,B^{T} - \mu B\,A^{T} = A\,B^{T} - B\,A^{T}}$

- This may be the kicker. The most important reason to represent a line in Plücker form.

Properties of Plücker matrix representation:

5. Under the point transformation $\vec{X}' = H\,\vec{X}$, the matrix L transform as $L' = H\,LH'$.

- This is the **most important reason** to represent a line in Plücker form. If you have a point homography between **2 sets of 3D points** then you can get a **Plucker matrix** homography by simple transformation of the line itself.

- $L' = H\,LH'$**.** We can use a point homography in order to transform lines from one projection to another. There is also a dual form of Plücker. We don't go into its details.

5. Plücker Line Coordinates

This is another way to represent a line. We can write Plücker matrix as 6 non-zero elements

$$\begin{bmatrix} 0 & -a & -b & -c \\ a & 0 & -d & -e \\ b & d & 0 & -f \\ c & e & f & 0 \end{bmatrix}_{4\times4}$$

- We can write **a, b, c, d, e, and f** in a vector, and we called them as a representation of the line.
- It is not quite nice because of it a 6 vector. At least it is a compact representation. We can go from this representation to Plücker matrix representation easily.
- That's why we have so **many different 3D line** representations because there's no good way to represent 3D lines.

5. Plücker Line Coordinates

The **Plücker** line coordinates are the six non-zero elements of the 4 × 4 skew-symmetric **Plücker** matrix L, namely

$\mathcal{L} = \{l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}\}$

This is a **homogeneous 6-vector**, and thus is an element of $\mathbb{P}^5$. It follows from evaluating det L = 0 that the coordinates satisfy the equation

$l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0$

A 6-vector *L* only corresponds to a line in 3-space if it satisfies **above equation**.

Note: The element l_{42} is conventionally used instead of l_{24} as it eliminates negatives in many of the subsequent formulae

Estimation – 2D Projective Transformations

- We spend a lot of time on statistics. We put **statistics** and **geometry** together, we get **computer vision**. We will start with a problem of how to **estimate homography automatically** even when there is **a noise in the image**.

- In computer science, we don't study statistical estimation as rigorously as in electrical engineering. Now we are starting chapter 4.

- In computer vision, we are frequently confronted with **estimation problems** in which **parameters of some functions** must be **estimated from measurements**.

Following are the fundamental problems in computer vision:

1.2D homography: Given a set of points x_i in $\mathbb{P}^2$ and corresponding points x_i' in $\mathbb{P}^2$ find a homography taking each x_i to x_i'

2.3D to 2D camera projection: Given a set of points X_i in 3D space and corresponding points x_i in an image, find the 3D to 2D projective mapping taking each X_i to x_i .

3.Fundamental matrix computation: Given a set of points x_i in one image and a set of corresponding points x_i' in another image, find the fundamental matrix F relating the two images.

4.Trifocal Tensor computation: Given a set of point correspondences $x_i \leftrightarrow x_i' \leftrightarrow x_i''$ across three images, compute the trifocal tensor T_i^{jk} relating points or lines in three views.

“**Estimation** is all about figuring out some unknown quantities given noisy measurements related to that object.”

- **1. 2D homography:** So, the first fundamental problem is **2D homography**. We have already studied it. We have also looked into the case that we have 4 correspondences between two planes. We can compute exactly the 2D homography between these two planes.

- But the problem is that **we never have exact measurements** of points in computer vision. We have **errors in measurements.** The fundamentally, image center is an array of discrete pixels how it is possible to measure the position of the point on this image sensor with less than one pixel of accuracy. But there is a limit on how accurately we estimate the position of an object on this image plane. We have noises. How can we take **multiple measurements** and then **try to minimize error as much as possible** by using multiple independent measurements?

2. 3D to 2D camera projection: The second problem that is quite important in computer vision is estimation from 3D to 2D camera projection.

What is 3D to 2D camera projection?

Basically, it is a **matrix called P**. It is 3×4. It models all the steps involved **in going from a world scene** to a **camera coordinate system** to 3D to 2D projection into an image plane.

We will learn how to estimate P from data

3. Fundamental matrix computation: The Fundamental Matrix is a new concept. It relates to two views of the same scene. In this case, an arbitrary scene. We already talked about a planar homography. We will model transformations between planar objects or images of an arbitrary scene taken with only rotations, but the origin of our camera is the same.

If we want to model transformations from one image to another image when the scene is arbitrary, and the camera position is arbitrary, then we need to know about the fundamental matrix.

Why Use the Fundamental Matrix in CV?

- Homography assumes **planar scenes or pure rotation**.
- **Fundamental matrix** handles general **3D scenes** and camera motion.
- Homography maps **points** directly; Fundamental matrix maps **points to epipolar lines**.
- Fundamental matrix is essential in **stereo vision**, **epipolar geometry**, and **3D reconstruction**.

Why Use the Fundamental Matrix in CV?

Aspect	Homography (H)	Fundamental Matrix (F)
Assumes	Planar scenes / rotation	General 3D scenes
Maps	Point to point	Point to epipolar line
Use Case	Image stitching /planar scenes	Stereo vision, 3D reconstruction
Valid When	No parallax	With parallax / depth