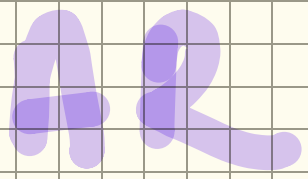


LESSON-1

- AR under the hood?

- My location is important } is an input
  (position)

- Highlighting the road } output

- Giving location on my head is expensive, highlighting is simple.

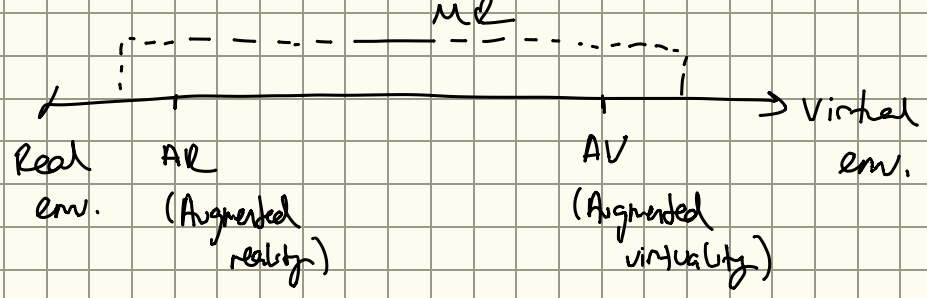
- I can interpret output of my program easier.

- Combine real and virtual

  → I can create a digital flower
  - ↳ I can create its sound
  - ↳ " " its smell

  } 3D related.

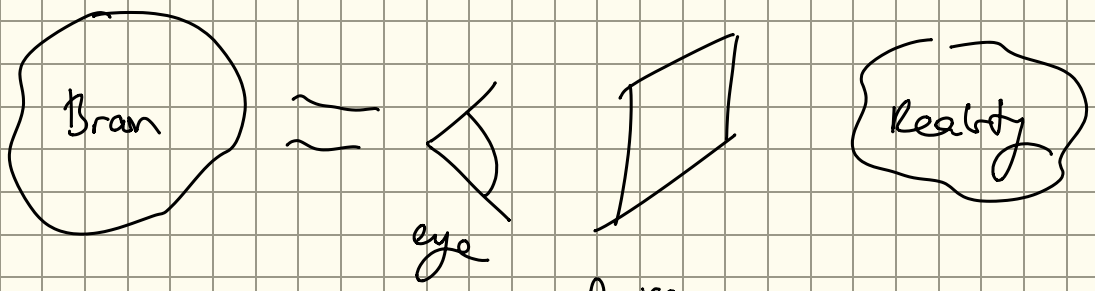
Mixed Reality



- Augmented is computer generated.
  ↓
  Combine real + virtual
  Registered in 3D.

→ In VR, you create env. from scratch.

→ Creating everything in vr is complex. Getting smth from env. makes it less complex.



This something let me perceive the reality in a virtual way.

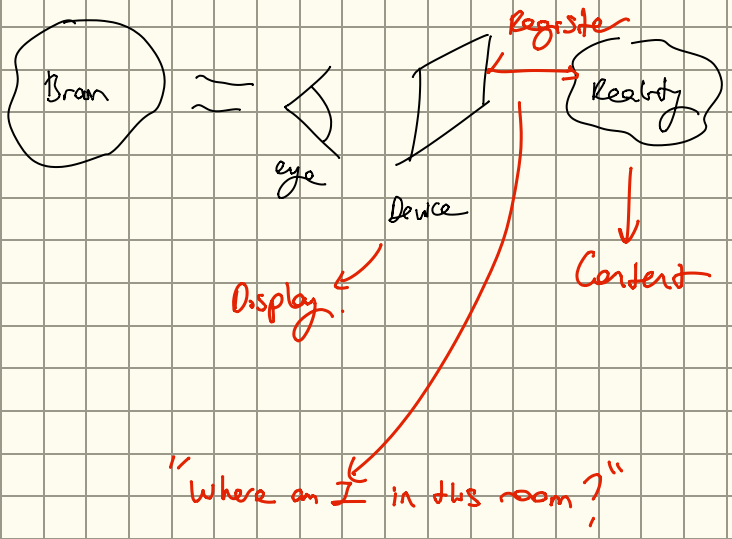
Visual!

- Ag. Optical seethrough → Transparent screen like glass
  - → User can see the real world through transparent display. And virtual image overlays on it.
  - → Hololens

- Ag. Video seethrough → Camera, smartphones / tablets.
  - → Overlays virtual image to real-world with video feed instead of direct view of world.
  - → Meta quest

- Ag. Projected AR → Projector
  - → Virtual object projected on a surface.

- The data has to be virtual, and it should be available.
- "I need to know exactly where I am in real time!" } Registration (Tracking)



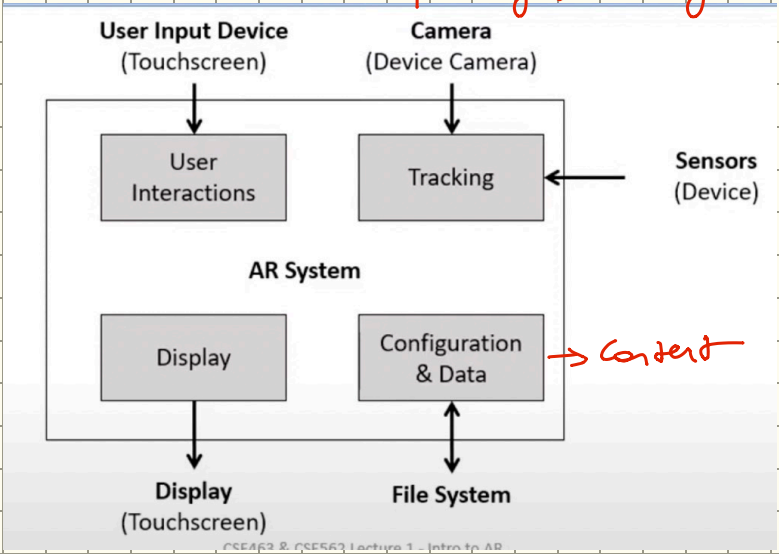
Prerequisites:
- Technology to display
- Content
- Align with reality
- Register in my world

Content is important
↳ What is the benefit of such a system?
↳ My needs!!

LESSON - 2

→ Understand what you have in env.

→ Using for part of reality in AR.



Hardware → display

→ CPU

↳ tracking } We need heavy load on CPU.

↳ Graphics } So, CPU needs to be good.

Optical seethrough : HMD, Hololens

Video seethrough: Some sensors, display, CPU

Tablet, smartphone

Most common usage

Software → graphics

↳ UX } How can I design smth that appeals the user?

↳ Overlay

→ tracking

↳ you need to track your location

↳ it has to blend with the env.

↳ I need to know where exactly things are → Recognition

d Reality is issue

↳ realistically show your flower on top of some surface.

↓

We can't do it without tracking.

d Camera is a good sensor

* Vuforia → ArKit.

AR Tracking

3D tracking → ArKit

3D env → Real world

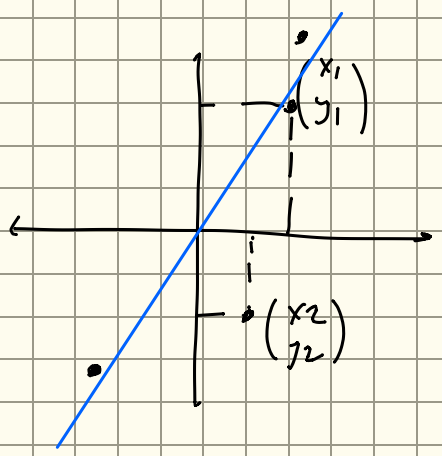
Virtual information → "I want to put flowers in here."

(Scene models context data, virtual 3D data ⇒ Content)

AR Virtualization → Unity → Need to design that interaction.

d Most of the time content is the most expensive and hard.

Linear Algebra



$y = f(x)$

$y = ax + b$ unknowns → a, b

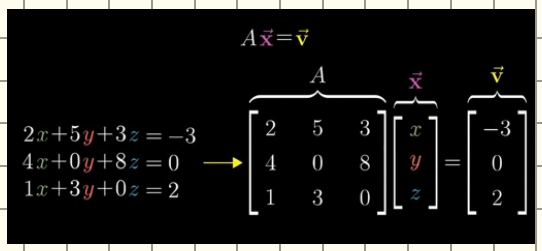
How can I find this?

$$y_1 = ax_1 + b$$
$$y_2 = ax_2 + b$$
$$y_3 = ax_3 + b$$
$$y_4 = ax_4 + b$$

$Ax = b$

unknown

matrix vector



$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \end{bmatrix}_{4\times 2} \cdot \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{2\times 1} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}_{4\times 1}$$

$A x = b$

$A^T \cdot A \cdot x = A^T \cdot b$

$x = \underbrace{(A^T \cdot A)^{-1}}_{\text{pseudo inverse}} \cdot A^T b$ → should be invertable

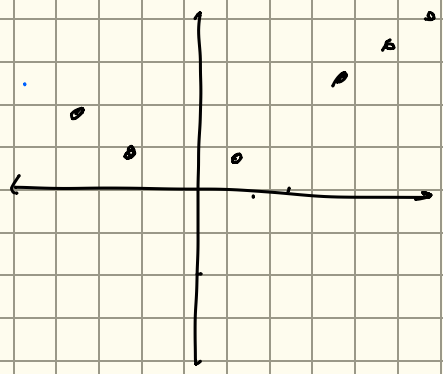
$\det(A) \neq 0 \rightarrow A^{-1}$ exists

Rank → Number of dimensions in the output.

Rank of matrix gives us idea to how bad of the transformation

↳ If the rank is low, the transformation compressing the input in some way, causing loss of information or depth.

Output of transformation is a line (1D) } Rank = 1



$$y = a_2 x^2 + a_1 x + a_0$$

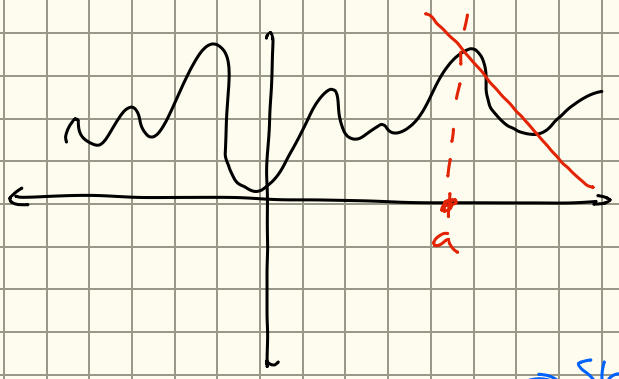
$$y_1 = a_2 x_1^2 + a_1 x_1 + a_0$$

$$y_2 = a_2 x_2^2 + \cdots$$

$$y_3 = - - - -$$

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ 1 & & 1 \end{bmatrix}_{k \times 3} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix}_{3 \times 1} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}_k$$

$$\underset{a_0, a_1, a_2}{\arg\min} \sum_{i=1}^{\hat{}} \left(y_i - (a_2 x_i^2 + a_1 x_i + a_0) \right)^2$$



Slope → Derivative $f'(x)$

Local Min. } not the answer

↳ Slope of function is 0

I know I will find 0 derivative if I go $a + \Delta a$ → Local min
→ Local max

↓

"Gradient Descent Algo" → Start at a point on the function

Move the direction that reduces the value of function

Find lowest point of function

What should I do to find global solution?

↳ Heuristic solution.

Optimization

Solvers : GD, LM

$$\arg\min_{\alpha} \sum_{i=1}^{n} (y - f(x, \alpha))^2$$

Minimize squared difference between actual and predicted values.

Robust Optimization - RANSAC

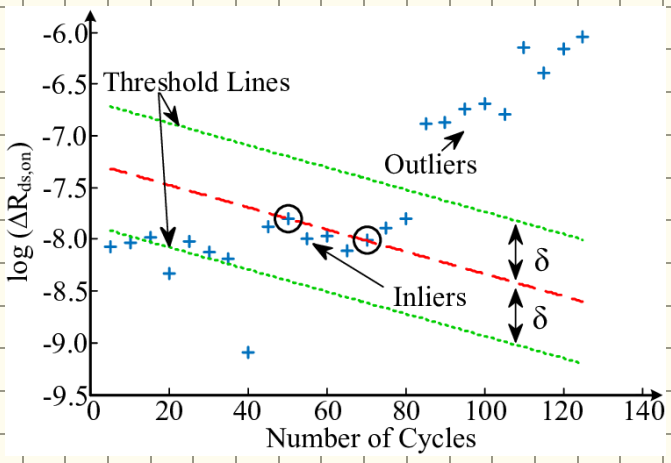
When I have some input x_i, I have corresponding answer y_i.

What if I make mistake?

$x_i \longrightarrow y_i$

= = } Some of them correct, Some of them not. } I can't use arg min anymore.

∝ RANSAC aims to find solutions that perform well even when some data points are noisy (outliers).



n=100, %10 outlier

d=5 unknowns (variables to solve for)

1-) Randomly sample a small subset of data points

2-) fit a model to this subset.

3-) Count inliners.

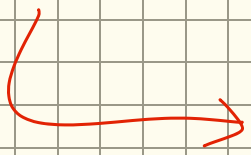
↳ ∝ Majority would tell me the true answer.

What if majority is wrong? → Most of them are outliers.

4-) Keep the best model: After repeating, select the model with most inliners.

n points m min # of points (m unknowns in model, we need at least m point)

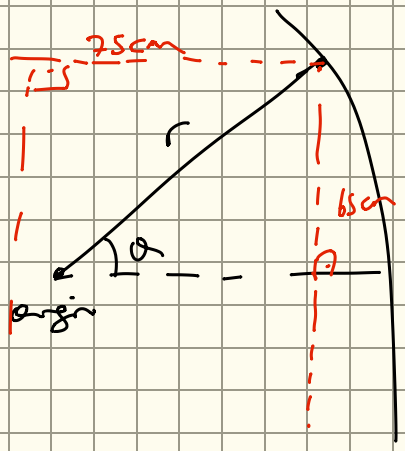
$m \ll n$ → not need to do random sampling again and again so much.



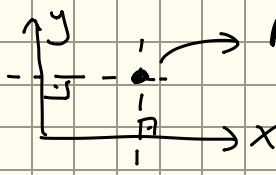
High possibility to choose the inliers.

LESSON - 3

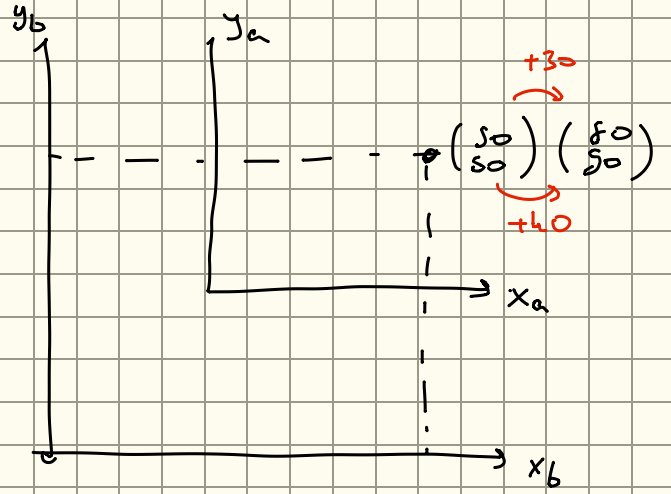
$\begin{pmatrix} r \\ \theta \end{pmatrix}$ → polar coordinates of our points.



→ $\begin{pmatrix} 65 \\ 75 \end{pmatrix}$ Send 2 numbers



Translation:



$P_b = P + \begin{pmatrix} 30 \\ 40 \end{pmatrix}$

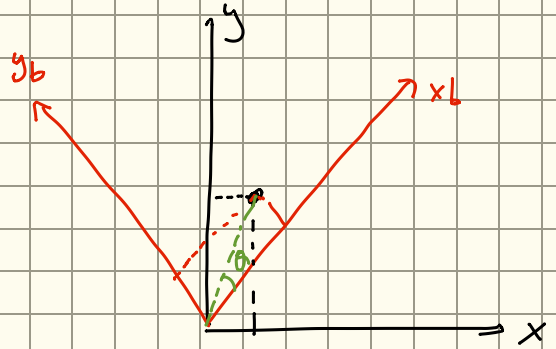
$P = P_b - \begin{pmatrix} 30 \\ 40 \end{pmatrix}$

} if and only if coordinates are parallel.

↓

addition / translation.

Rotation:



$\begin{pmatrix} x_b \\ y_b \end{pmatrix} \longleftrightarrow \begin{pmatrix} x \\ y \end{pmatrix}$

$x_b = \cos\theta x + \sin\theta y$

$y_b = -\sin\theta x + \cos\theta y$

}

$$\begin{pmatrix} x_b \\ y_b \end{pmatrix} = \begin{pmatrix} c\theta & -s\theta \\ s\theta & c\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

↓

rotation, multiplication

Both addition and multiplication,

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = R \begin{pmatrix} x \\ y \end{pmatrix} + T$$

↓

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

↓

→ Transformation of 2 coordinate system define like this

$$P_n = R \cdot P_o + T$$

↳ has 3 degrees of freedom.

Geometric Transformations

2 Image dersinde gördüm.

$$C = \begin{bmatrix} s_x\cos\theta & -\sin\theta & t_x \\ \sin\theta & s_y\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Scale
- Translation
- Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos a & -\sin a & h \\ \sin a & \cos a & k \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3D Transformation

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} & & & t_x \\ & & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rigid Transformation

- Involves only translation and rotation. It will not change its shape.

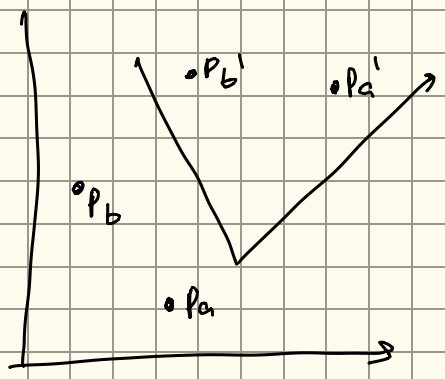
$$C = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Affine Transformation

- Arbitrary sequence of rotation, scale and translation

$$C = \begin{bmatrix} s_x\cos\theta & -\sin\theta & t_x \\ \sin\theta & s_y\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rigid Transformation



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} c\theta & -s\theta \\ s\theta & c\theta \end{pmatrix}}_{R} \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} t_x \\ t_y \end{pmatrix}}_{T}$$

$$\det = c\theta^2 - s\theta^2$$

p_a $$\begin{pmatrix} x_a' \\ y_a' \end{pmatrix} = R \begin{pmatrix} x_a \\ y_a \end{pmatrix} + T$$

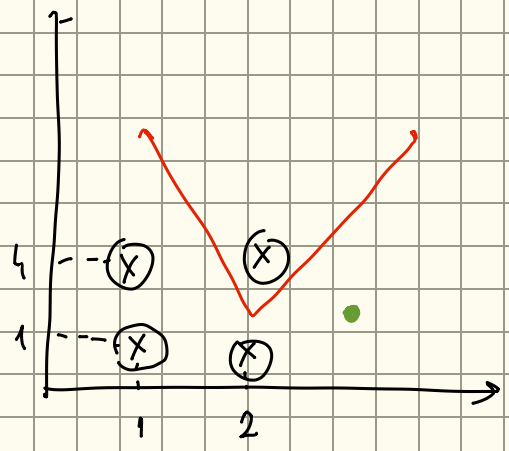
p_b $$\begin{pmatrix} x_b' \\ y_b' \end{pmatrix} = R \begin{pmatrix} x_b \\ y_b \end{pmatrix} + T$$

$$p_a' - p_b' = R(p_a - p_b)$$

If $\det(R) = 1$ $\Rightarrow$ $p_a' - p_b' = p_a - p_b$?

$$c\theta^2 - s\theta^2$$

Example



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} c\alpha & -s\alpha \\ s\alpha & c\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} dx \\ dy \end{pmatrix}$$

$P_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $P_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ $P_3 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ $P_4 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

$P_1' = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$ $P_2' = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ $P_3' = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$ $P_4' = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$

How can I transform another point?

Let write the formula for known points:

$$\left.\begin{array}{l} P_1' = R P_1 + T \\ P_2' = R \cdot P_2 + T \end{array}\right\} \quad \begin{pmatrix} -1 \\ 0 \end{pmatrix} = R \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} + T \quad \therefore \quad \therefore$$

$$P_3' = R \cdot P_3 + T$$

$$P_4' = R \cdot P_4 + T$$

$$\left.\begin{array}{l} -1 = \cos\alpha - \sin\alpha + \delta x \\ 0 = \cos\alpha + \sin\alpha + \delta y \end{array}\right\} \text{ 2 equations}$$

$$x_1' = \cos\theta\, x_1 - \sin\theta\, y_1 + t_x$$
$$y_1' = \sin\theta\, x_1 + \cos\theta\, y_1 + t_y$$

$$x_2' = \cos\theta\, x_2 - \sin\theta\, y_2 + t_x$$
$$y_2' \quad '' \qquad\qquad ''$$

$$x_3'$$
$$y_3' \quad '' \qquad\qquad ''$$

$$x_4'$$
$$y_4' \quad '' \qquad\qquad ''$$

unknowns

θ, t_x, t_y

$A\begin{pmatrix}\theta\\ t_x\\ t_y\end{pmatrix} = b$

θ is not linear.

Non-linear solution → Gradient Descent.

$$\underset{\theta, t_x, t_y}{\arg\min} \sum_{i=1}^{4} \left(\left| P_i' - (R P_i + T) \right| \right) \rightarrow \text{gradient descent.}$$

Initial guess can be obtained from linear solution such that:

$\alpha = \cos\theta$

$\beta = \sin\theta$

Linear solution:

$$\left.\begin{aligned} \cos\theta &= \alpha \\ \sin\theta &= \beta \end{aligned}\right\}$$

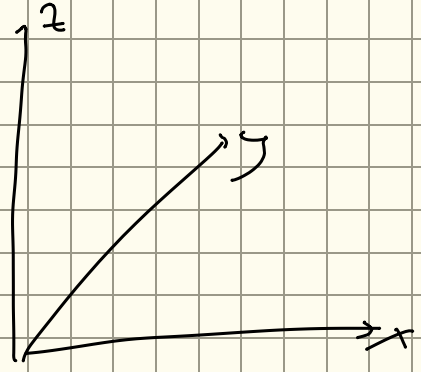
unknowns:

α, β, t_x, t_y

$$\left.\begin{aligned} P_i' &= R P_i + T \end{aligned}\right\} \quad \left.\begin{aligned} P_i &= R^{-1}(P_i' - T) \end{aligned}\right\}$$

$R^T \cdot R = I$

orthonormal.



$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \end{pmatrix} = R \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + T$$

Rotation matrix.

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} R_{2\times 2}$$

$R_{3\times 3}$

$R = R_{\theta z} \cdot R_{\theta y} \cdot R_{\theta x}$

$$\begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

$R \cdot P + T$

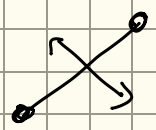
$\theta_x \in [0, 2\pi]$

$\theta_z \in [0, \pi]$

$\begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \end{pmatrix}$ → Euler Angles

6 degrees of freedom → at least need 3 points

↳ 2 points isn't sufficient, we don't know the orientation



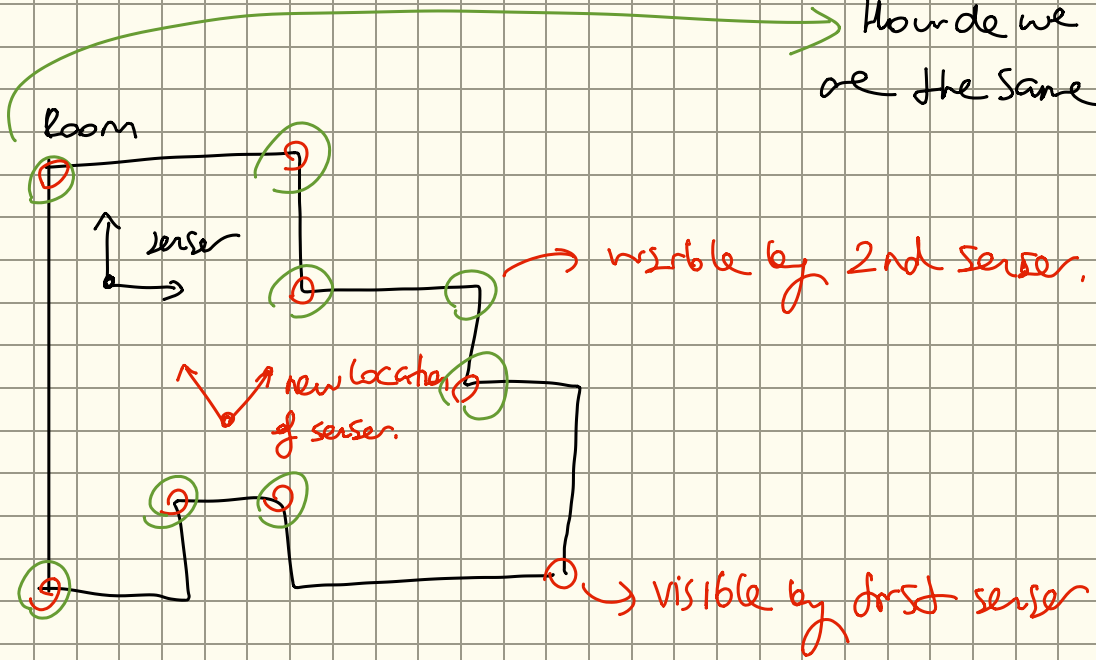
n eq in n unknowns

↓

should be given some information that are not repetitions.

↳ linearly dependent?

Practical application of Rigid transformation:



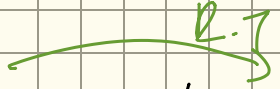
→ If I know transformation between these sensors, I can add invisible points.

↳ I can increase my awareness.

$P_i' = R \cdot P_i + T$ } If actually measured rightly.

∝ With translation and rotation information, you can map the coordinates of the points seen in the old position to the new position. ("translate")

→ If I make a wrong choice:



a_1	b_1	a_1'
a_2	b_2	a_2'
a_3	b_3	
a_4	b_4	
a_5	b_5	
a_6	b_6	
a_7	b_7	

Assume I choose wrong points:

$a_1 - b_3$

$a_4 - b_4$

If distance is sufficiently small, they are correct.

RANSAC

1-) subsample

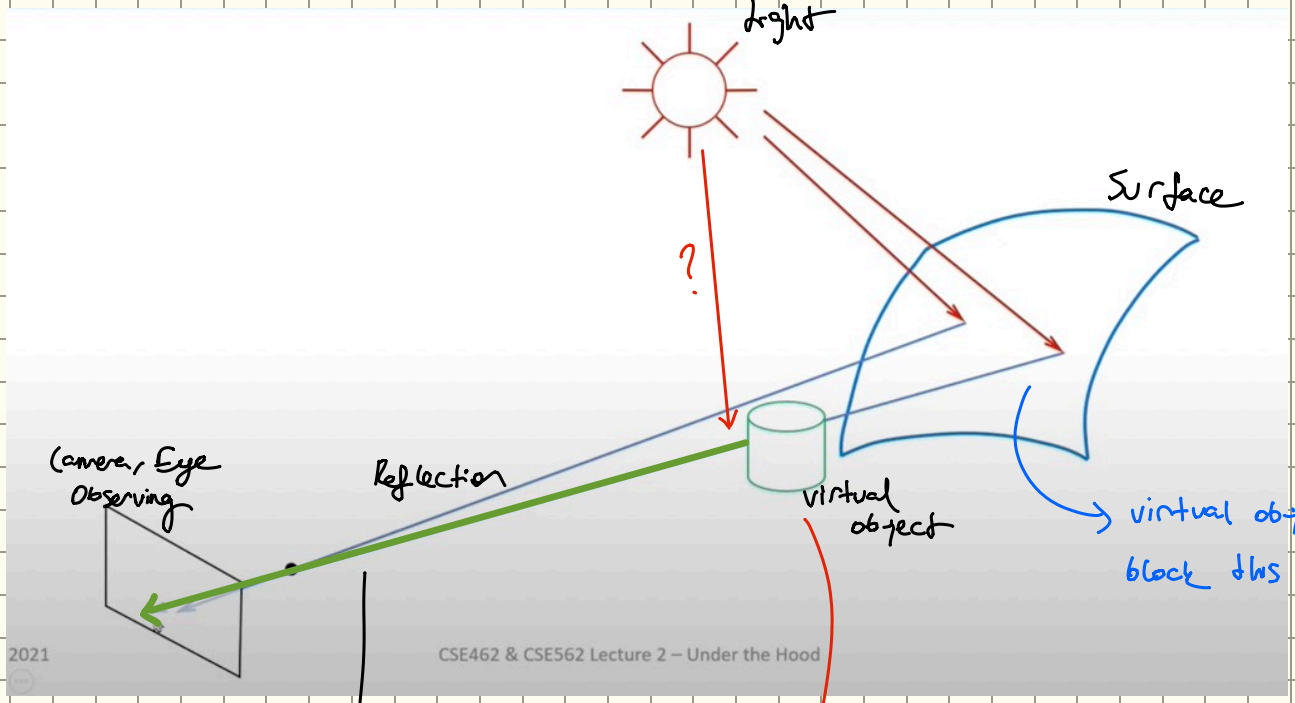
2-) match 2 points

3-) rescale until match.

→ Check how many other points fit this transformation

→ Outliers (incorrect points) won't fit.

Spatial Augmented Reality



Reflected light from object:
"realism, depth, shadows"

Augmented object

→ You need to understand:

- Geometry of object
- Where it is your camera
- " " light source
- Amount of reflection (objects physical interaction with light)

→ Where is light?

- It is important depending on purpose of virtual object.
  - visibility, depth, realism
  - ↓
  - 1-) Red arrow for exit, not that important
- It is important for mismatches, shadows, specific experiment.
- Predicting light in env. is hard. (tracking)

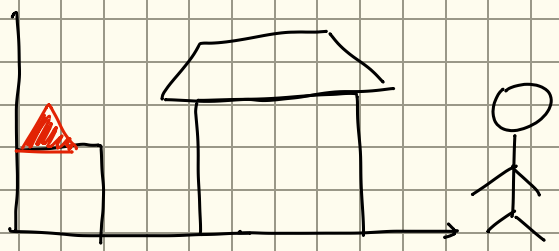
1-) I need to understand where the object is with respect to camera

↳ Tracking

2-) I need to model the object → Authoring

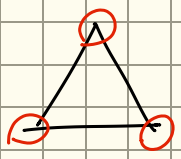
3) " " know the light location → Environment Modeling

2D



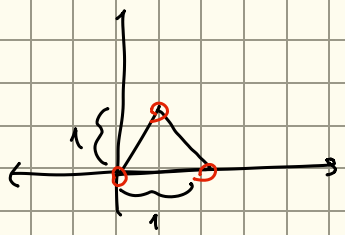
Real env.

∠ You want something to virtual object to bring something to this env.



→ exactly define what this triangle in coordinate system.

↓



- vertices are vectors.

$$\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1/2\\1\end{bmatrix}$$

∠ I want to show this triangle in real world } Agment it.

∠ Where should I put this triangle?

↳ It depends what I want to do.

↳ Define coordinate system for real world. This real and virtual corresponds in one where.

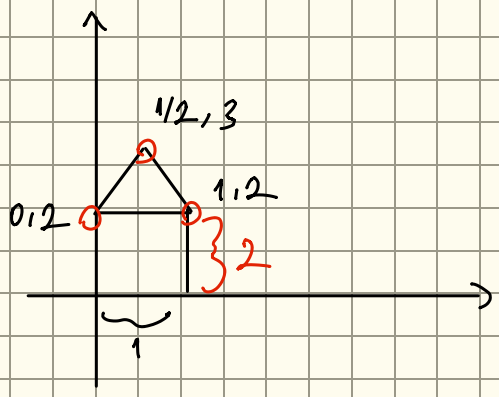
↳ Make it "arbitrary".

How can I make sure that they are in same place?

- Calculating locations one by one

↓

I don't want to do this.



$P_R = R \cdot P_V + T$

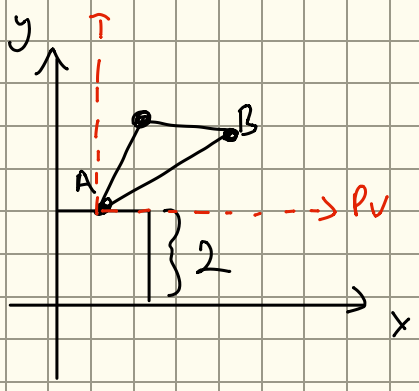
↓ Or ↓ t_x, t_y

Get the transformation:

1-) No rotation → $R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (identity matrix)

2-) $T = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$

! Not practical but I can do it



2 points:

A: $P_V = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ $P_r = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

B: $P_V = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $P_r = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$

I have enough points to measure now.

2 equations

1-) $\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

$\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$ → $t_x = 1$, $t_y = 2$

2-) $\begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

$\begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ } $\cos\theta + t_x = 4$, $\sin\theta + t_y = 4$

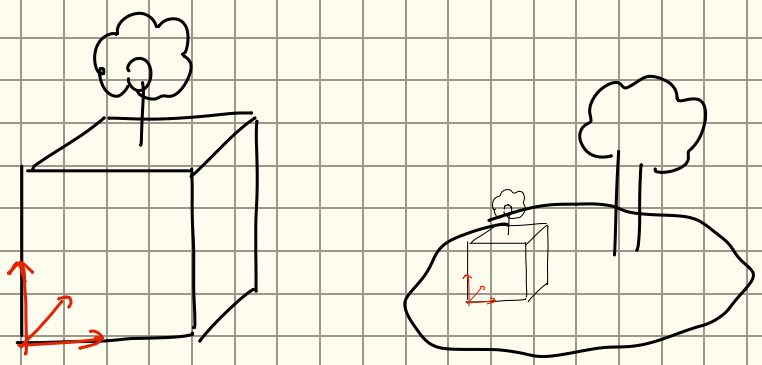
↓

We can find the θ angle with these equations.

∝ In 3D, I need 3 points.

1-) Define coordinate system for real world.

2-) At least find 3 points corresponding ones.

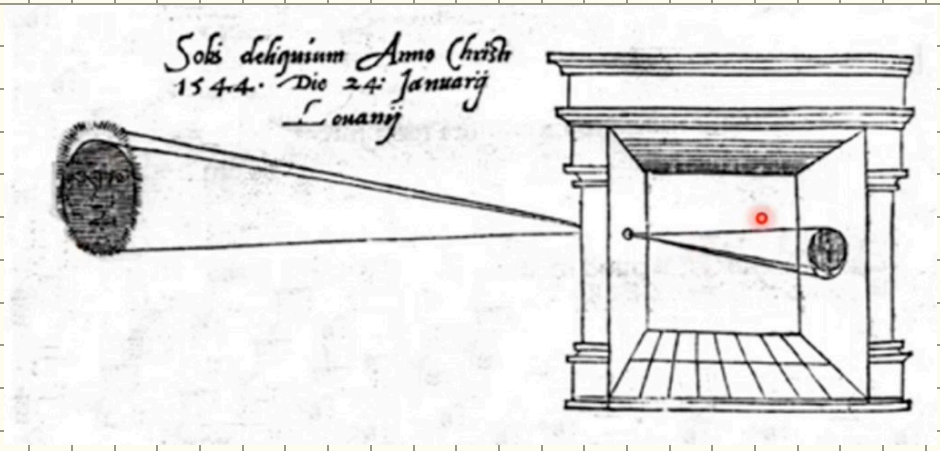


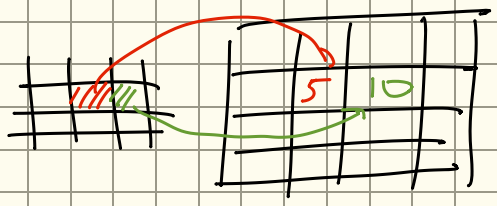
∝ Find a way to associate with reality.

$P_R = R \cdot P_V + T$ } Tracking, computer vision techniques.

∝ Once you know the transformation between reality and virtual, you can apply this transformation to align virtual objects onto real world scene in AR applications.

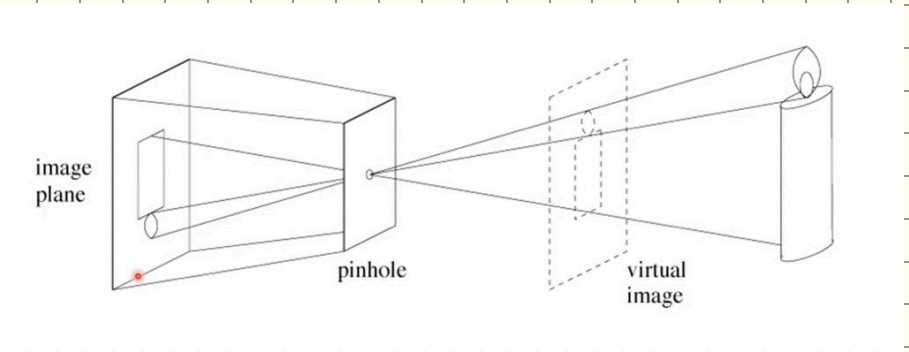
Camera Obscura





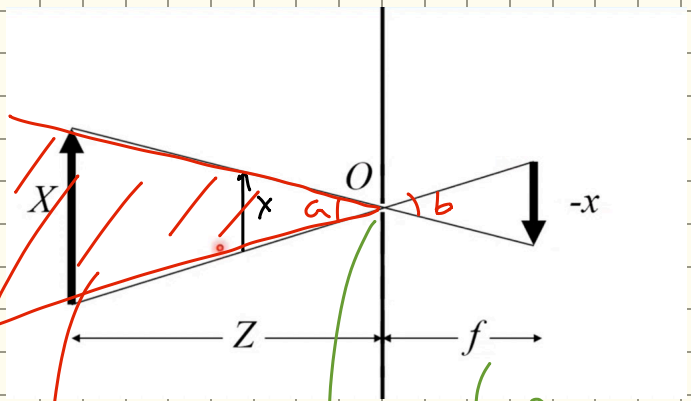
for each location, put sensors.

Pinhole Camera



Perspective Projection

2D:



$$x = X \frac{d}{2}$$

$$\frac{a}{b} = \frac{X}{-x}$$

I have very simple model now.

We can use it to generate or estimate images.

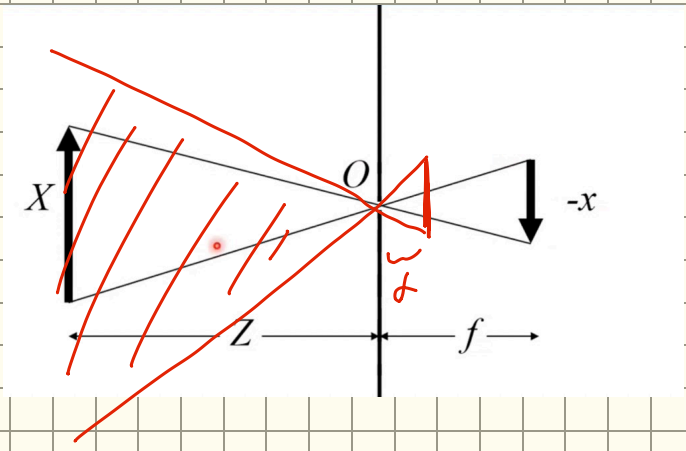
↳ focal length

Field of view depends on focal length

Pinhole

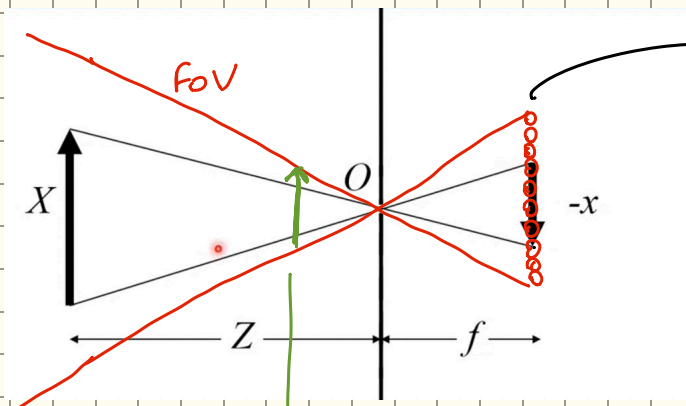
PINHOLE CAMERA VIDEO NOTES

CCD, CMOS image sensors used in cameras to capture light.



smaller focal length

larger FoV



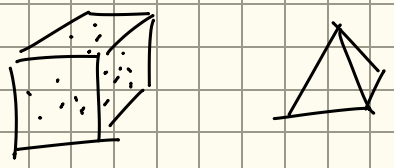
10 sensors → My resolution says I can only measure 10 different locations light.

Object size 1mm → would occupy 10 pixels
High resolution.

1mm → would occupy 2 pixels
low resolution.

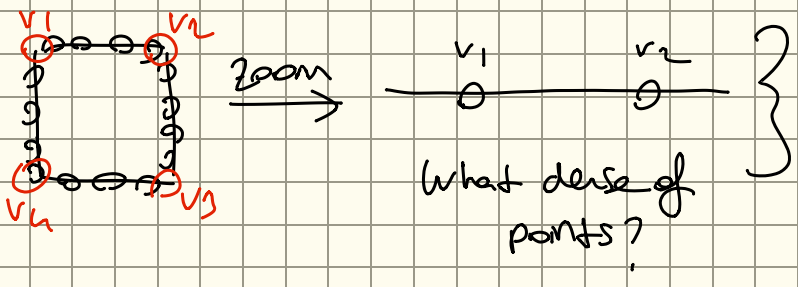
PINHOLE CAMERA MODEL

- How to provide virtual object?
- Virtual object is collection of points in 3D space.
- These points expressed as vectors → (x, y, z)



- To render the object, we only need its surface } Because we want to visualize this. We don't need the inside of that.

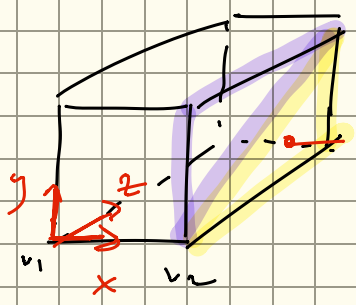
- How many points that I need?



} I don't need to know what between, I can guess it.
Point is a sampling mechanism. (approximate)

$V = v_1, v_2, v_3, v_4$ $\quad$ $T = \langle v_1 v_2 \rangle, \langle v_2 v_3 \rangle \langle v_3 v_4 \rangle \langle v_4 v_1 \rangle$

↳ Triangles is smallest unit to construct surfaces. Bunch of point sampling this space.



$v_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ $v_2 = \begin{bmatrix} 1000 \\ 0 \\ 0 \end{bmatrix}$ $v_3 = \begin{bmatrix} 1000 \\ 1000 \\ 0 \end{bmatrix}$ ...

In coordinate system;

→ Triangles:

v_2, v_6, v_7

v_2, v_3, v_7

} 12 triangles for a cube

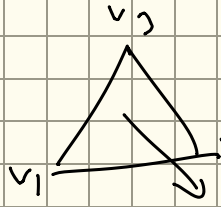
⋮

↳ How will I know the orientation? (Outside or inside surface)

↳ Normals: A vector perpendicular to the surface of triangle

↓ I can avoid of normals.

Surface Tesallation



$\langle v_1, v_2 v_3 \rangle$ } different directions

$\langle v_1 v_3 v_2 \rangle$ }

↳ You are gonna just use 1 right handed system.

↳ Ordering is important now.

- We now have everything we need for input object. (to render)
  - ↳ vertices and triangles

- The object is represented in vertex space

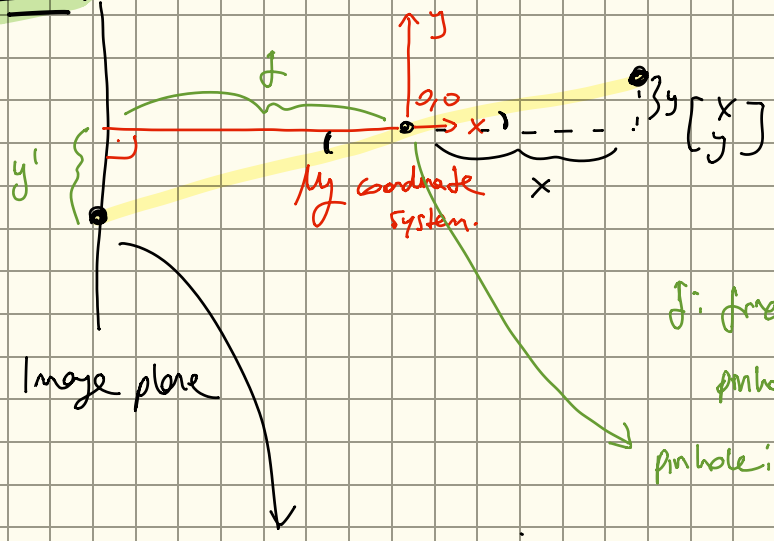
- The representation passed to the rendering system.
  (3D object to 2D image using Pinhole Camera Model)

1-) Object representation → vertices and triangles

2-) Surface orientation

3-) From 3D to 2D

2D



f: fixed distance between my wall and pinhole.

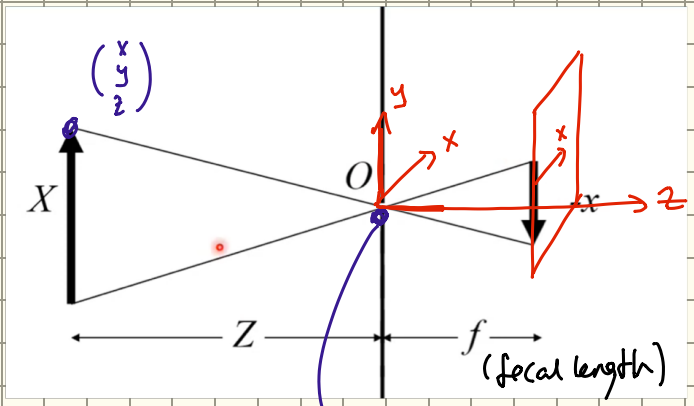
pinhole: center of projection (COP)

If I find this location I can find where the image projecting on this space.

$$\frac{y'}{f} = \frac{y}{x} \qquad y' = f \cdot \frac{y}{x}$$

3D

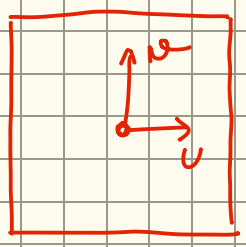
∝ 3D to 2D projection.



∝ 3D point (x, y, z) in the real world is projected onto a 2D image plane (camera sensor) via.

↳ center of camera world, good point to start coordinate.

Image plane.



$f(x,y,z) \rightarrow (u,v)$? → x, y, z coordinate of my world.

$$u = x \cdot \frac{f}{z} \qquad v = y \cdot \frac{f}{z}$$

$$\frac{u}{x} = \frac{f}{z} \qquad \frac{v}{y} = \frac{f}{z}$$

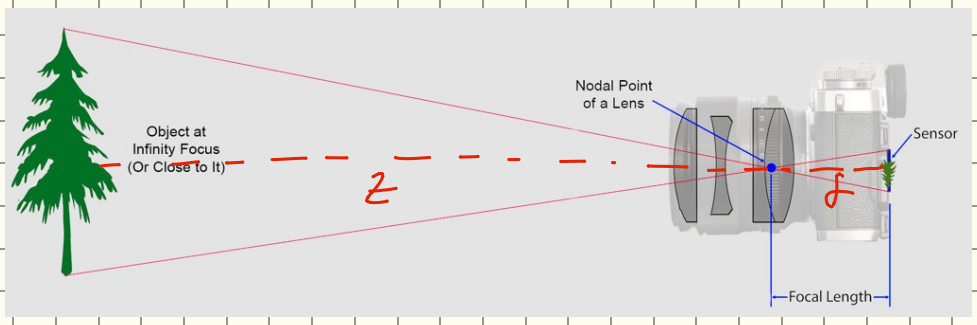
Eg

$x = 2, y = 3, z = 5$ focal length $= 10$

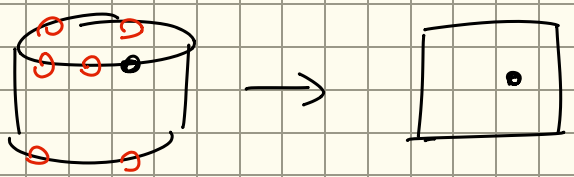
$$\frac{u}{2} = \frac{10}{5} \quad u = 4$$

$$\frac{v}{3} = \frac{10}{5} \quad v = 6$$

(2, 3, 5) in 3D world appear in (4, 6) on the 2D image plane.



∴ Now if you give me x, y, z, f I can find u, v. But what about other points?



∴ I am interested in visible points.

Graphics take 3D objects, break them down into triangles, and use transformation to determine where each triangle should appear in 2D image.

$u = x \cdot \frac{f}{z}$ $\quad v = y \cdot \frac{f}{z}$ } We like more compact representations.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} \quad \text{(divide by } z\text{)}$$

$u = \frac{fx}{z}$, $v = \frac{fy}{z}$, $1 = 1$

Now I have a relationship.

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

dividing z part.

$s \cdot u = fx$, $s \cdot v = fy$, $s = z$

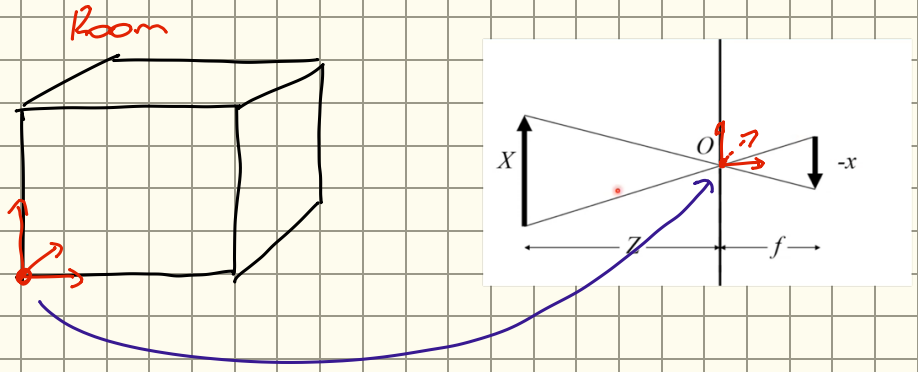
3D to 2D projection equation:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$s \cdot p_I = K \cdot p_w$$

∂ I need to know x, y, z points with respect to center of camera.

I have no idea where my camera is.



Take this coordinate and put it to COC.

$$\rho \cdot P_I = K \cdot [R \cdot P_W + T]$$

∂ Now, I can measure things in my world with respect to coordinate system of my choice.

∂ How many degrees of freedom we have between your and mine coordinate system?

$$\rho \cdot P_I = K \cdot [R \cdot P_W + T]$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\theta_x, \theta_y, \theta_z$

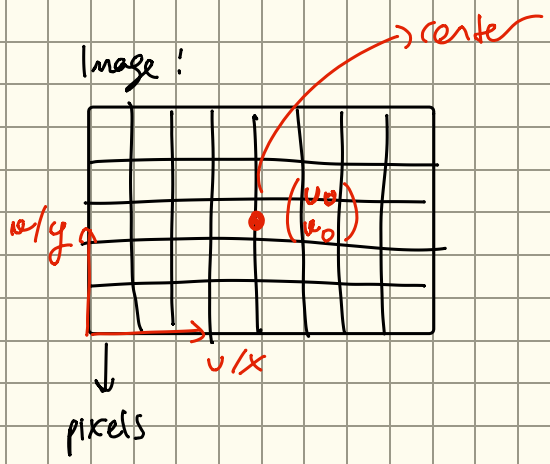
$\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$

6 parameters (DoF)

"extrinsic parameters" R, T

↓

external calibration, position and orientation of camera



camera
K → intrinsic parameters, cameras properties like focal length etc.

$$K = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$K = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \text{translation}$$

↳ Now, my camera has $\underline{=}$ 3 intrinsic parameters.
(focal length + center of image.)

↳ Now, I can't guarantee that my x and y direction have same scale in my image plane. → Maybe there are non-square pixels.

↓ I will add α.

$$K = \begin{pmatrix} \alpha \cdot f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad \left.\right\} \underline{=} \text{ 4 parameters.}$$
(focal length + center of image + scale)

↓ I will add s

$$K = \begin{pmatrix} \alpha \cdot f & s & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

s is shear, adjusts parallelism of image.

$\underline{=}$ 5 parameters.

↳ If your camera is fixed, once you find it, you're done.

Final form:

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{3\times 3} (R \cdot P_w + T)$$

K

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Show in matrix multi.

$$\begin{bmatrix} P_w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Projection

internal external

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_x & s & u_0 & - \\ 0 & \alpha_y & v_0 & - \\ 0 & 0 & 1 & - \end{pmatrix} \underbrace{\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}_{4\times 4} \underbrace{\begin{bmatrix} P_w \\ 1 \end{bmatrix}}_{4\times 1}$$

I need to make this also 4x4.

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \pi_c \cdot T_{wc} \cdot T_{ow} \cdot P_o$$

π_c: camera parameters; T_{wc}: w→c; T_{ow}: o→w

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

From calibration to tracking:

Everytime the camera frame changes or every 30 seconds I need to get R and t.

So, internal is easier.

Weak Perspective Projection:

It simplifies the math by assuming that all points in the scene are at a similar distance from the camera.

"constant average depth"

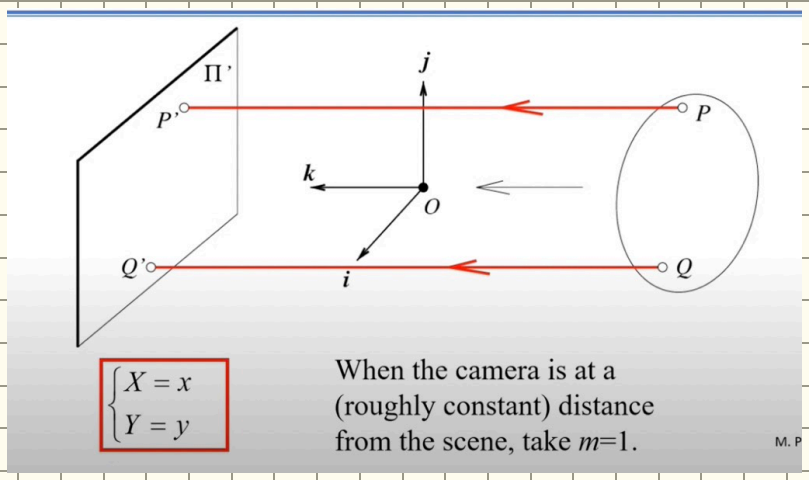
↓

z is fixed.

$$X = x \cdot \frac{f}{z_0 + \Delta z} \approx z_0$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

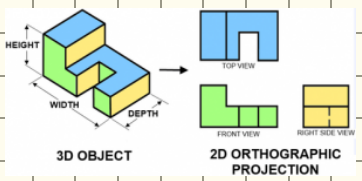
Orthographic Projection:



No depth or perspective. No z.

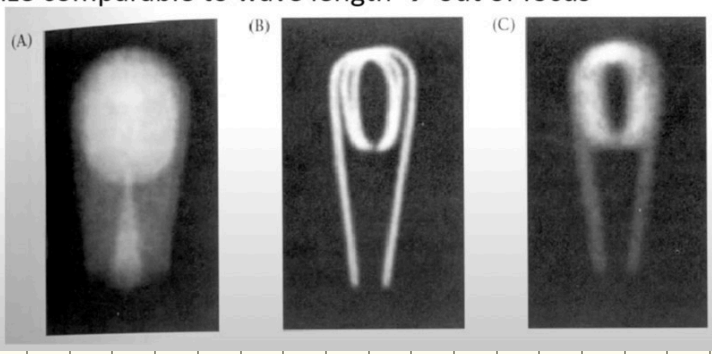
$u = x$

$v = y$



1. Perspective	$x = X\frac{f}{Z}$	$y = Y\frac{f}{Z}$
2. Weak perspective	$x = c\,X$	$y = c\,Y$ → constant
3. Orthographic	$x = X$	$y = Y$

- Hole too large ➔ out of focus
- Hole too small ➔ image too dark
  - Hole size comparable to wave length → out of focus



Circle of Confusion!

Müller-Lyer Illusion!

Projection Equation:

$$\Pi = \begin{bmatrix} -f s_x & s & u_0 \\ 0 & -f s_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{3\times3} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times1} & T_{3\times1} \\ 0 & 1 \end{bmatrix}$$

identity

intrinsics

rotation | translation

I can make "rendering" in here

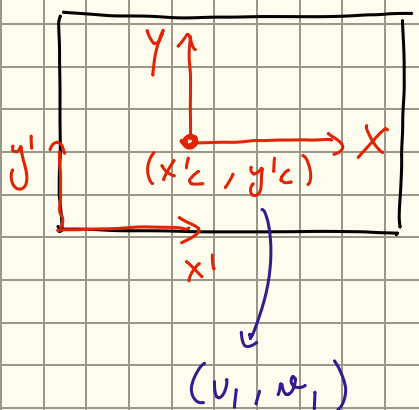
$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}_{4\times4}$$

∴ For fixed camera in my world:

- 6 parameters (unknowns) for externals
- 5 " " " internals

} 11 unknowns

Image:



$$f\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = K \cdot \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

↓ When making the matrix calculations

$$s \cdot u_1 = f(k \cdot R \cdot T \cdot x_1 \cdot y_1, z_1)$$

$$s v_1 = f(k \cdot R \cdot T \cdot x_1, y_1, z_1)$$

$$\cdot s = f(- - -)$$

↓

$$u = x \frac{f}{z} \qquad v = y \cdot \frac{f}{z}$$

$$(u_1, v_1) \begin{cases} u_1 = \dfrac{f_1(K R T x_1, y_1, z_1)}{f_3(\dots)} = x_c \cdot \dfrac{f_x}{z_c} + c_x \\ v_1 = \dfrac{f_2(\dots)}{f_3(\dots)} = y_c \cdot \dfrac{f_y}{z_c} + c_y \end{cases} \quad \text{2 equations}$$

↳ from calculus, I know that — if I have n unknowns, I need n equations.

↓

If I measure 6 points → I have 12 equations. } This can solve my problem.

↓

We can make calibration. Then tracking?

Calibration: Done once during setup.
Focuses on determining K (intrinsic)

Tracking: Happens continously during AR operation.
Dynamically updates R, T (extrinsic) in real time.

↳ If I somehow make the calibration: I have 6 unknowns (R, T)

↓

3 points now.

1. World coordinates: $P_{\text{world}} = [2, 3, 10, 1]$.
2. Extrinsics: Assume:

$$[R|T] = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

3. Intrinsics: Assume:

$$K = \begin{bmatrix} 800 & 0 & 640 \\ 0 & 800 & 360 \\ 0 & 0 & 1 \end{bmatrix}$$

$$p\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = K \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

1-) $\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 13 \end{bmatrix}$

2-) $\begin{bmatrix} 800 & 0 & 640 \\ 0 & 800 & 360 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 5 \\ 13 \end{bmatrix} = \begin{bmatrix} 10720 \\ 7480 \\ 13 \end{bmatrix}$

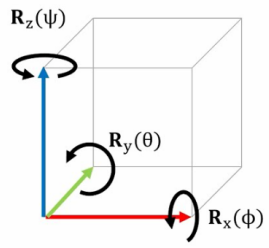
3-) $\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix}$ $\left\{ \begin{array}{l} 10720/13 \simeq 824 \\ 7480/13 \simeq 575 \end{array} \right\}$ $\begin{pmatrix} 824 \\ 575 \\ 1 \end{pmatrix}$

The full rotation matrix would be:

$$R = R_z(\theta_z) \cdot R_y(\theta_y) \cdot R_x(\theta_x)$$

Where R_x, R_y, and R_z are the individual rotation matrices for each axis:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}, \quad R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \quad R_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Ref

for external parameters;

Rotation for Z-axis (R_z);

$$\begin{bmatrix} \cos & -\sin & 0 \\ \sin & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{\theta = 90^\circ} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = R$$

$\cos(90) = 0$

$\sin(90) = 1$

Assume that camera positioned $(x, y, z) = (2, 3, 5)$ in world coordinate system.

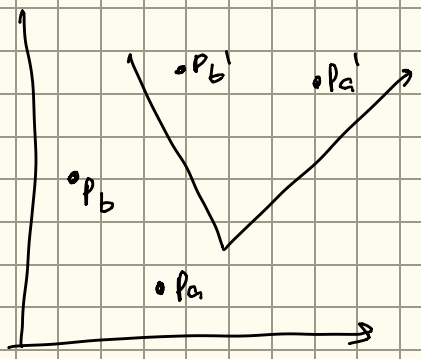
$$T = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 2 \\ 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times4}$$

Rigid transformation: 3D → 3D

$$P_{camera} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot P_{world}$$

camera position and orientation



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} c\theta & -s\theta \\ s\theta & c\theta \end{pmatrix}}_{R} \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} t_x \\ t_y \end{pmatrix}}_{T}$$

Perspective Projection: 3D → 2D

$$P_{image} = K \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot P_{world}$$

↳ Reduces 3D data 2D, introducing perspective effects to simulate how cameras and human eyes perceive the world.

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \pi \cdot T_{wc} \cdot T_{ow} \cdot P_o$$

- π: Camera parameters
- T_{wc}: w → c
- T_{ow}: o → w
- $\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$

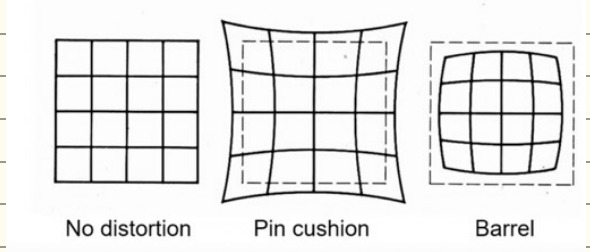
↓ Manipulated by Vuforia.

If Vuforia sees the real, it can calculate T_{wc}

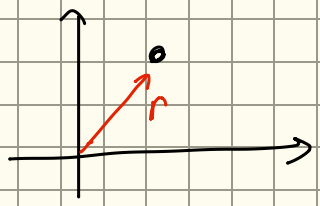
(real → virtual)

object space
↓
world space
↓
camera "
↓
image "

Radial Distortion:



Modeling Distortion.



Project $(\hat{x}, \hat{y}, \hat{z})$ to "normalized" image coordinates	$x'_n = \hat{x}/\hat{z}$ $y'_n = \hat{y}/\hat{z}$
Apply radial distortion	$r^2 = x'^2_n + y'^2_n$ $x'_d = x'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$ $+k_3 r^6 \ldots$ $y'_d = y'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$
Apply focal length translate image center	$x' = f x'_d + x_c$ $y' = f y'_d + y_c$

- To model lens distortion
  - Use above projection operation instead of standard projection matrix multiplication

Ex/

$P_{camera} = [1, 2, 5]$

Normalize } $x'_n = \frac{1}{5} = 0,2$ $\quad y'_n = \frac{2}{5} = 0,4$

$k_1 = 0,1 \quad k_2 = 0,01$

$r^2 = 0,2^2 + 0,4^2 = 0,2$

$x'_d = 0,2\,(1 + 0,1 \cdot 0,2 + 0,01 \cdot 0,04) = 0,20408$

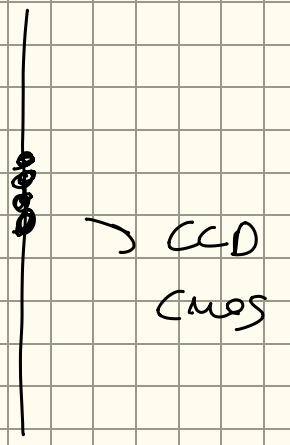
$y'_d = 0,4\,(1 + 0,1 \cdot 0,2 + 0,01 \cdot 0,04) = 0,40816$

$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} \text{intrinsic} \end{bmatrix} \cdot \begin{bmatrix} 0,20408 \\ 0,40816 \end{bmatrix}$ } $u' = f \cdot x'_d + x_c$, $v = f \cdot y'_d + y_c$

- Computer vision : Detects real world env.
  Reconstructs the 3D world
  2D (images) → 3D (world reconstruction)

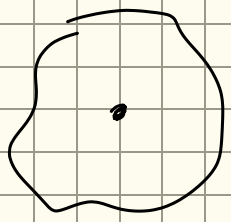
- Computer graphics : Create 2D images or videos from 3D data.
  3D (scene) → 2D (image rendering)

22 feb

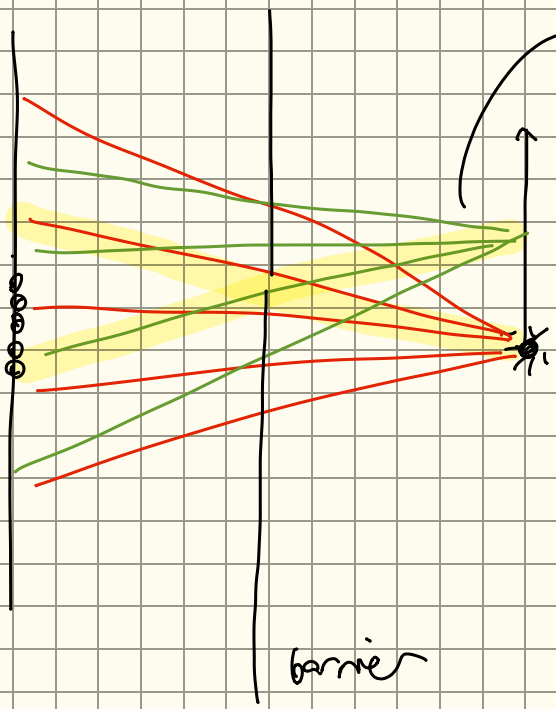


→ CCD
CMOS

Sensor
(light measuring)



< In order to form an image of this object.



A very simple imaging device. We want to formulate this.

Input objects → their images.

(Computer graphics) to generate virtual views

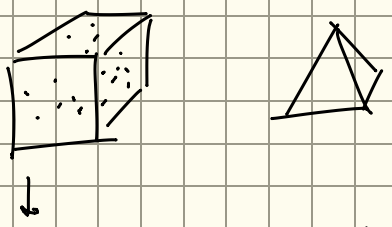
Can I reverse this? Image → actual objects in the scene that generates this view

Computer vision. where to put this virtual world ↓ tracking.

↳ AR deals both of them.

↳ How can I provide the virtual object?

↳ How can I describe the image?

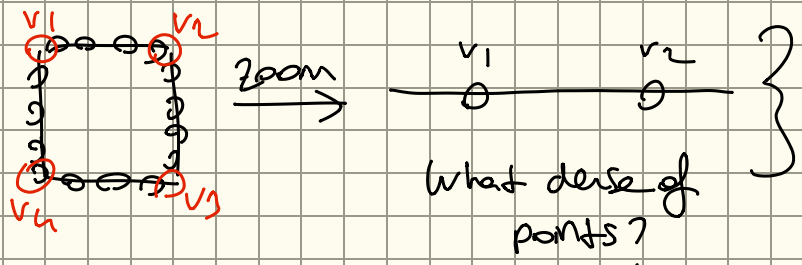


I have bunch of points. So I need vectors.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

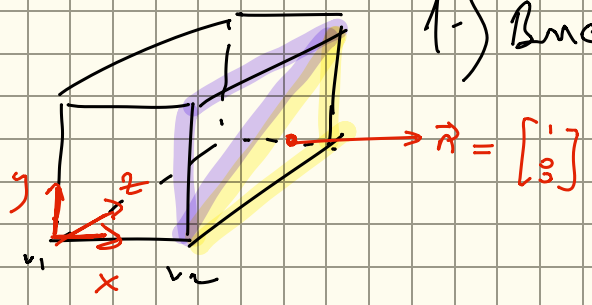
↳ We need surface of the object. } Because we want to visualize this. We don't need the inside of that.

↳ How many points that I need?



} I don't need to know what between, I can guess it. Point is a sampling mechanism.

$V = v_1, v_2, v_3, v_4$ $\quad T = \langle v_1 v_2 \rangle, \langle v_2 v_3 \rangle \langle v_3 v_4 \rangle \langle v_4 v_1 \rangle$



1-) Bunch of points sampling the space.

$v_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ $v_2 = \begin{bmatrix} 1000 \\ 0 \\ 0 \end{bmatrix}$ $v_3 = \begin{bmatrix} 1000 \\ 1000 \\ 0 \end{bmatrix}$ ...

In coordinate system:

Vertices:

→ $v_1 \dots v_3$

→ Triangles:

v_2, v_6, v_7

v_2, v_3, v_7

} 12 triangles.

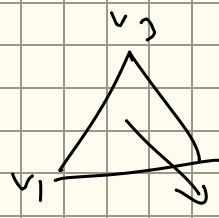
⋮

& How will I know the orientation? (Surface is inside or outside).

Normals: towards outside. } 12 normals

Surface Tesalation

I can avoid of normal:



$\langle v_1, v_2, v_3 \rangle$ } different directions

$\langle v_1, v_3, v_2 \rangle$ } directions

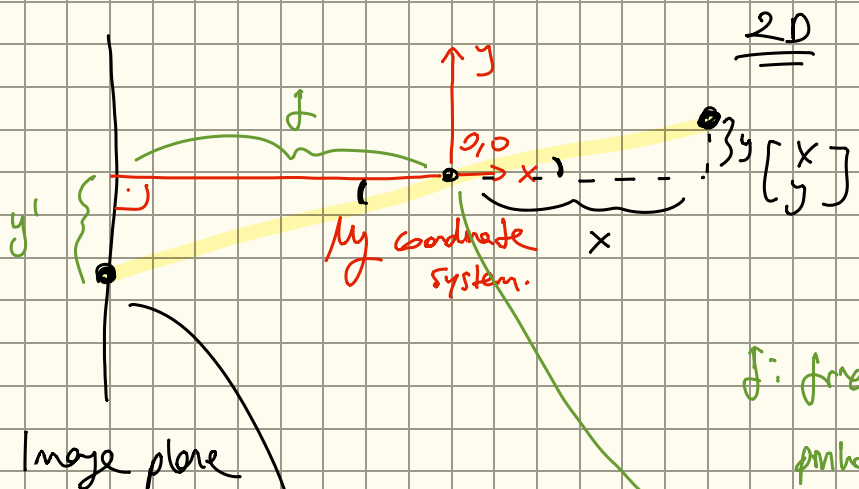
& You are gonna just use 1 right handed system.

∝ Ordering is important now.

∝ We have know everything we need for input object..

input objects → then merges.

∝ Representation is in vertex place. Where is obtained my image?



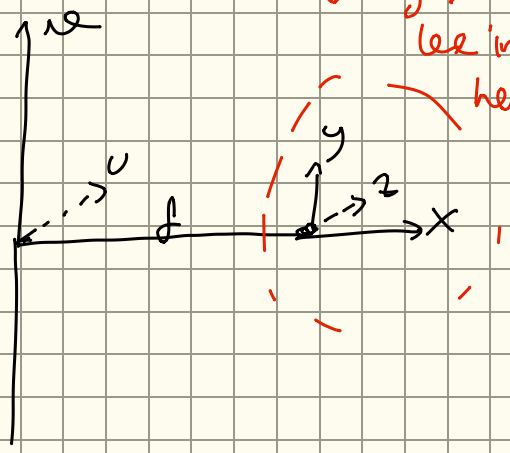
f: fixed distance between my wall and pinhole.

pinhole: center of projection (COP)

If I find this location I can find where the image projecting on this space.

$$\frac{y'}{f} = \frac{y}{x} \qquad \boxed{y' = f \cdot \frac{y}{x}}$$

3D



Vertices and triangles will be in here

$$u = f \cdot \frac{x}{z}$$

$$v = f \cdot \frac{y}{z}$$

Not a very convenient formula.

We want like this $\Rightarrow P_2 = [\,]\,P_3$

I have a projection matrix now.

$$f \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Pi} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{P} \in \mathbb{R}^4$$

Bruyteleer rule

$$fu = a_1^T \cdot P$$

$$fv = a_2^T \cdot P$$

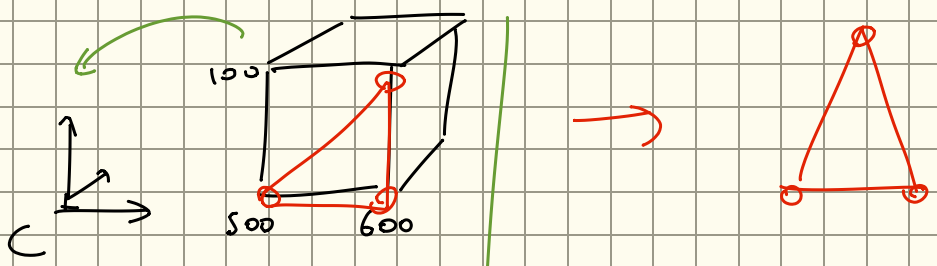
$$f = a_3^T \cdot P$$

$$u = \frac{a_1^T \cdot P}{a_3^T \cdot P} \in$$

$$v = \frac{a_2^T P}{a_3^T P} \in$$

$$f \cdot p_i = \Pi \cdot P_{camera}$$

rigid transformation $W \to C$



$$P_C = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_W$$

Model view.

$$s p_i = \Pi \cdot \begin{bmatrix} K & T \\ 0 & 1 \end{bmatrix} P_C$$

Projection

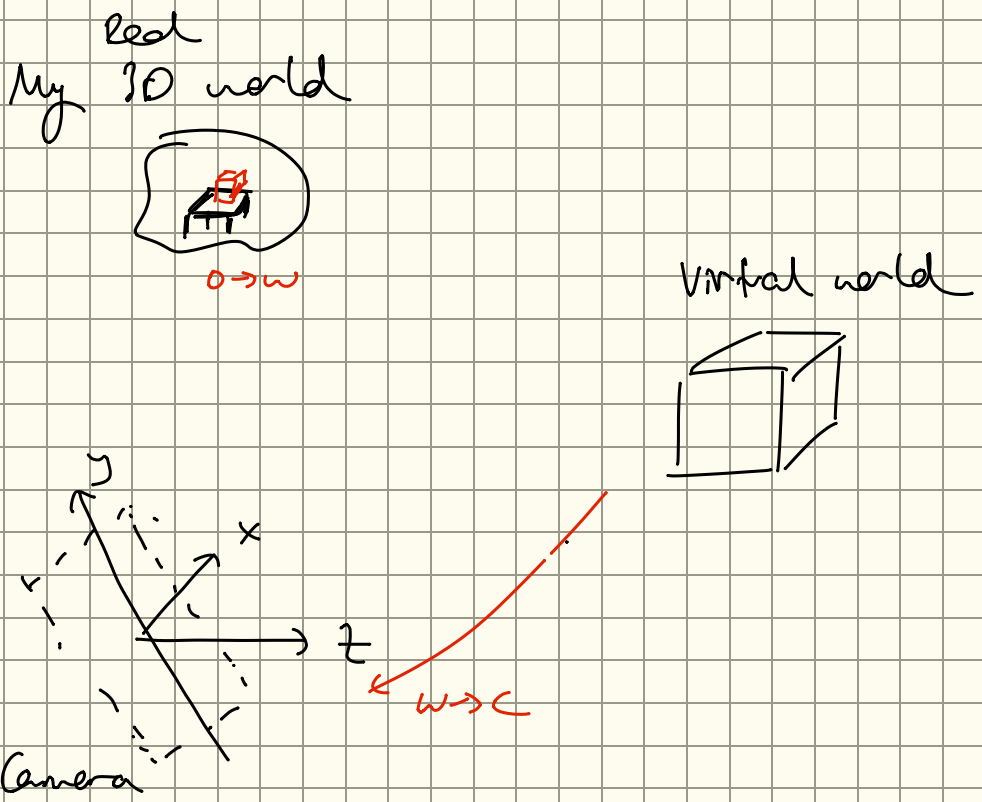
(Not rigid transformation.)

Distances are no longer meaningful.

↳ It models our pinhole camera problem.

HW → Change your scene!

Image target works with same eyes!?



$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot T_{wc} \cdot T_{ow} \cdot P_o$$

- K: Camera parameters
- T_{wc}: W→C
- T_{ow}: O→W
- $\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$

↓ Manipulated by Vufora.

If Vufora sees the real, it can calculate T_{wc}

(real → virtual)

Bradon Telor Dale

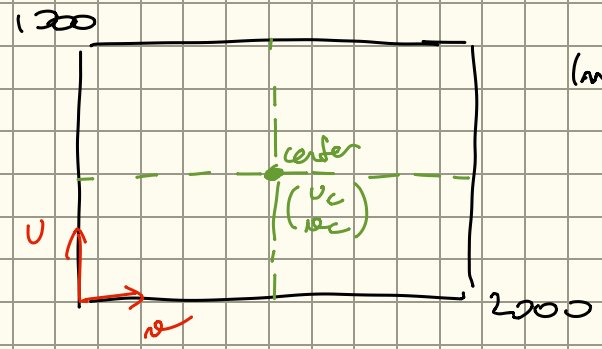
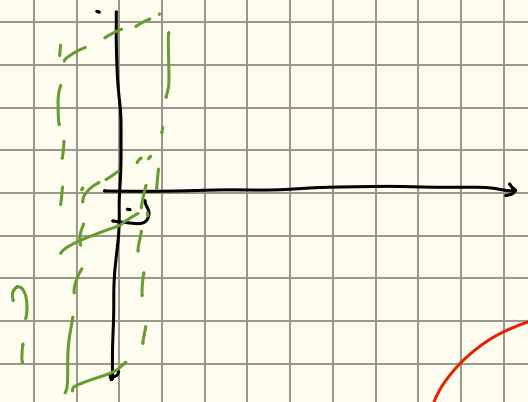


Image plane

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \Pi \cdot T_{oc} \cdot P_o$$

$$\Pi = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

f scale
f sheer

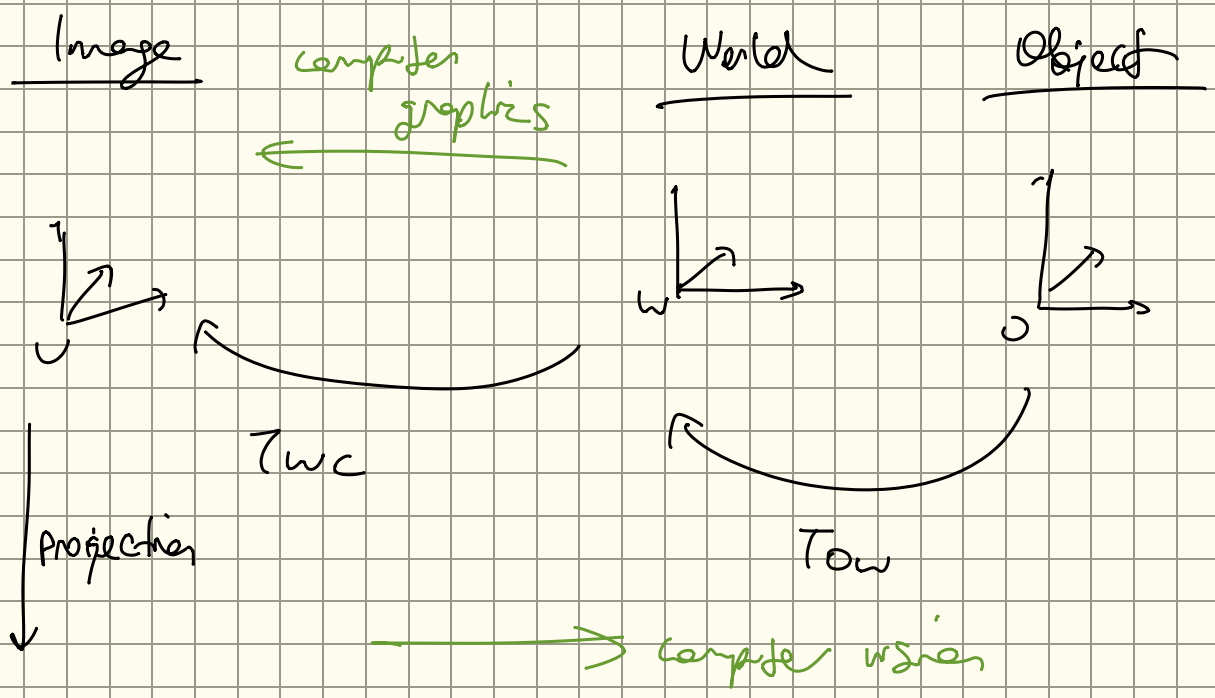


Most of time unity used take this into account.

aspect ratio?

center of image?

$$\Pi = \begin{bmatrix} S_x f & S & u_c & 0 \\ 0 & S_y f & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

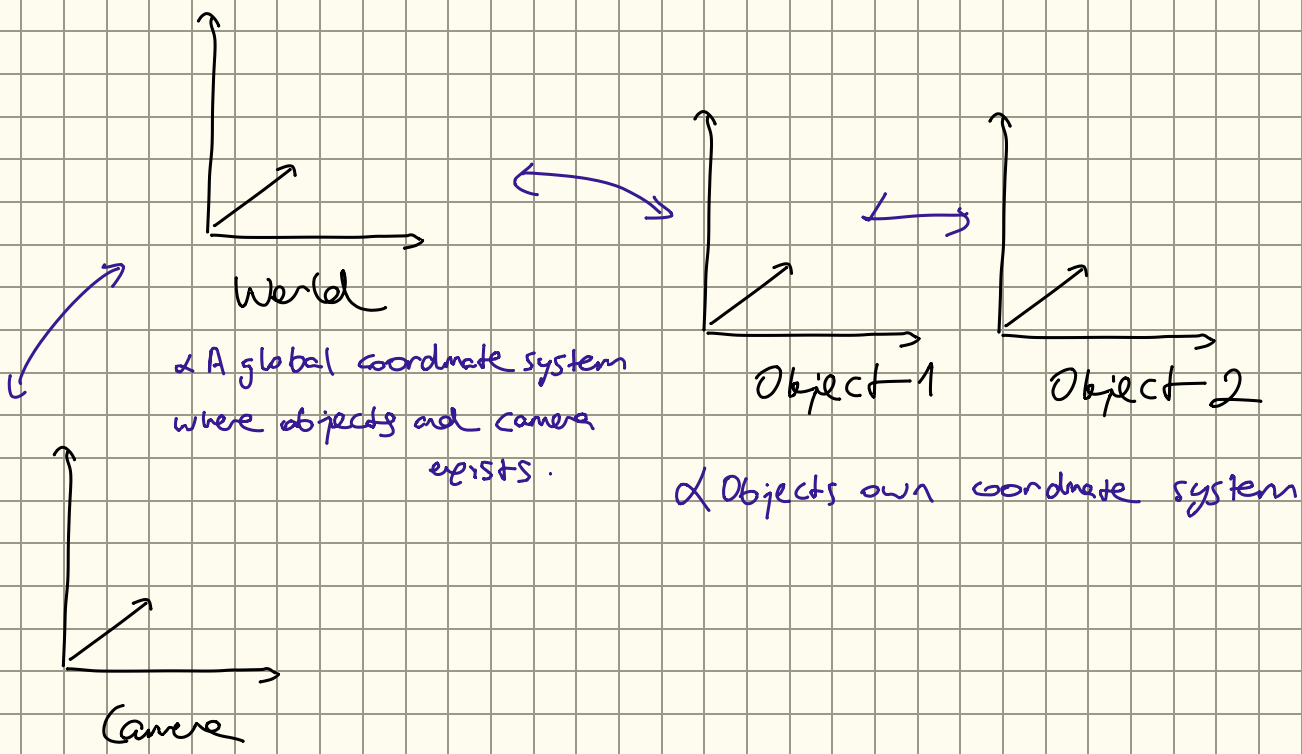


$$s_i \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & s & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} * T_{wc} * T_{ow} * \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

This is in euclidean coordinate system

RENDERING PIPELINE

3D Trasformation:



∝ A global coordinate system where objects and camera exists.

∝ Objects own coordinate system

Rigid (3D) trasformation:

$$P_2 = R \cdot P_1 + T$$

↓

Inverse trasformation:

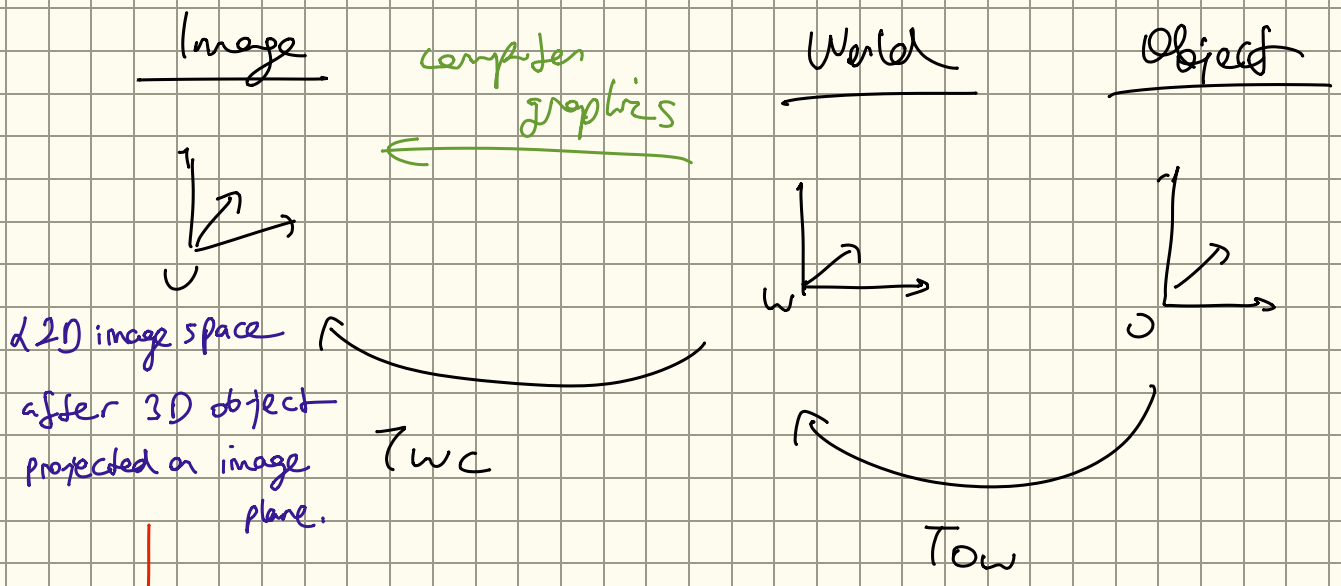
$$P_1 = R^T (P_2 - T)$$

$$R^T \cdot R = I$$

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \xrightarrow{\text{inverse}} \begin{bmatrix} R^T & -R^T \cdot T \\ 0 & 1 \end{bmatrix}$$

$$P_2 = \mathcal{T} \cdot P_1$$

$$P_1 = \mathcal{T}^{-1} \cdot P_2$$



projection ⟶ computer vision

$$s_i \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & s & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} * T_{wc} * T_{ow} * \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

This is in euclidean coordinate system

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \Pi \cdot T_{wc} \cdot T_{ow} \cdot P_o$$

- Π: camera parameters
- T_{wc}: w→c
- T_{ow}: o→w → $\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$

↓ Manipulated by Vuforia.

If Vuforia sees the real, it can calculate T_{wc}

(real → virtual)

Transformation between spaces:

1-) Object to World:

- Maps point from the object coordinate system to world ... ...

$$T_{ow} = \begin{bmatrix} R_{ow} & T_{ow} \\ 0 & 1 \end{bmatrix}$$ - Rigid transformation

2-) World to Camera:

$$T_{wc} = \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix}$$ - ... ...

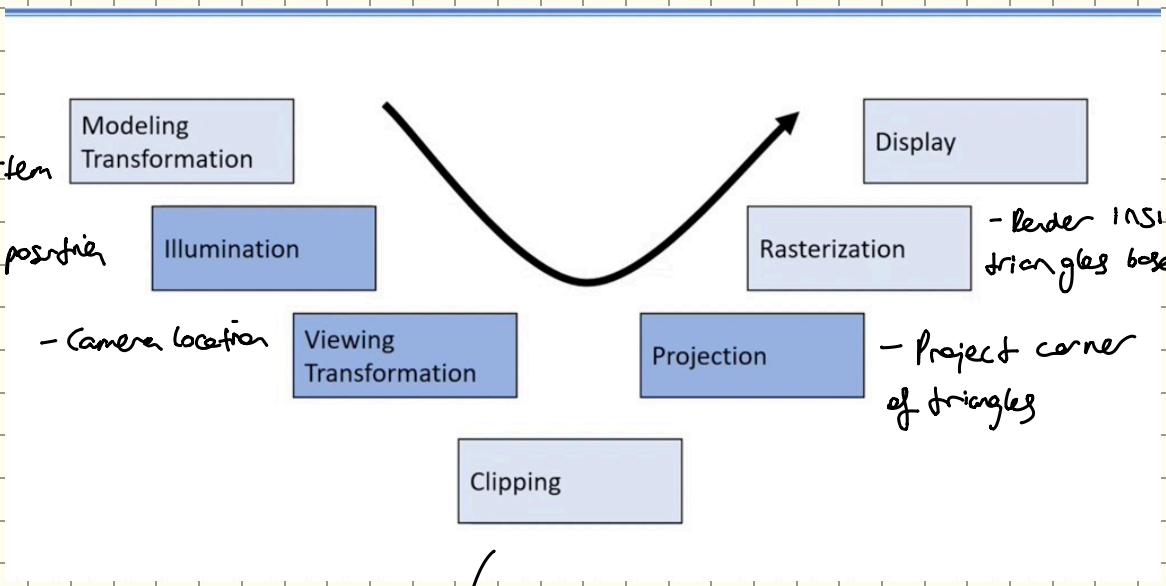
3-) Camera to Image Projection:

- Maps 3D point in the camera coordinate system to 2D image space.

$$\Pi = \begin{bmatrix} \alpha \cdot f_x & s & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

& 3 rotations (Rx, Ry, Rz)

Rendering Pipeline:



- A global coordinate system

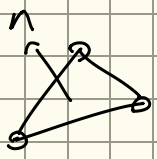
- Amount and position of light

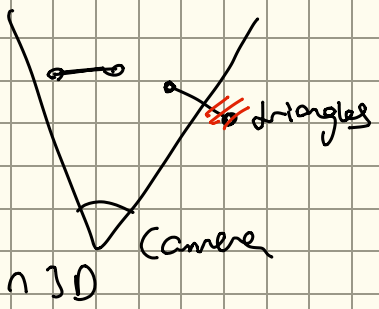
- Camera location

- Render inside of triangles based on light and normals.

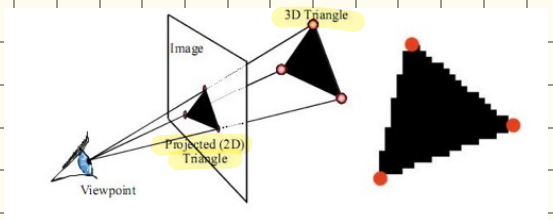
- Project corner of triangles

- Clipping triangles





Rasterization:

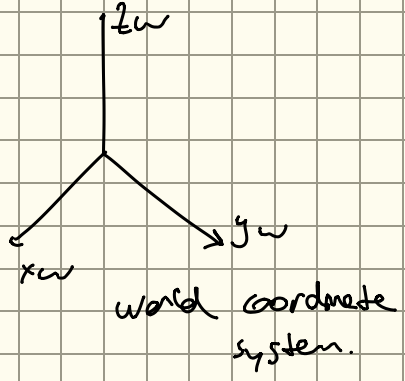


& OpenGL is implementation of this pipeline.

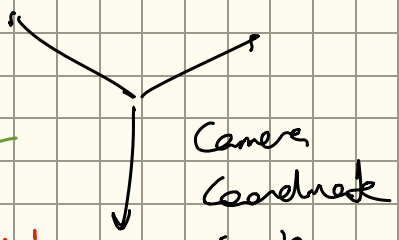
Calibration Issues

12 Kasım 24

Unity has fixed coordinate system for real world.



R, t



$$s\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} -f_{sx} & s & x_c \\ 0 & -f_{sy} & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

camera placement

Intrinsic parameters (Camera calibration)

Extrinsic parameters

How someone externally put the camera.

Where your camera's coordinates in real world.

Computer graphics

Computer vision

If we knew these, can we guess other parameters?

- Calibration

↳ We want to recover them.

$$s\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

$P_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$ ↓ $q_i \begin{pmatrix} u_i \\ v_i \end{pmatrix}$

Scale that I don't know.

known

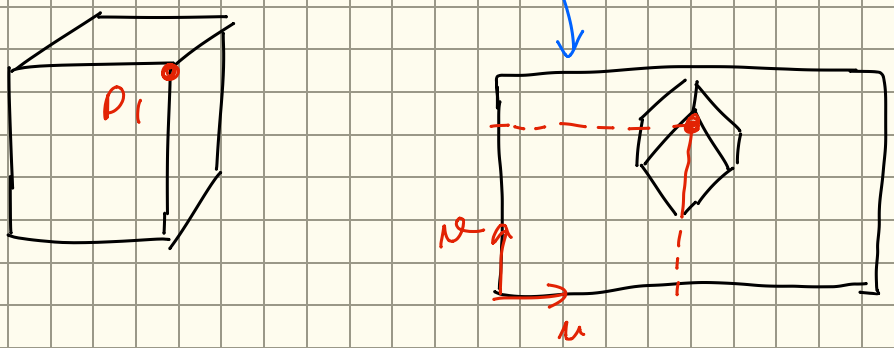
↳ We are gonna recover the parameters. Lets simplify things.

Assume we know R and T:

$$s_1\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

↓ known, measured

↓ known, it is given to me



$$s_1\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

$$s_2\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix}$$

⋮ ⋮ ⋮

- 5 unknowns

α_u, α_v → focal lengths of x, y

u_0, v_0 → center of camera

s → scale

- 5 unknowns

(n) relations → 4 camera intrinsic parameters

n scale

Matrix multr:

$$\rho_1 \cdot u_1 = \alpha u \cdot x_1 + u_0 \cdot z_1$$

$$\rho_1 \cdot v_1 = \alpha v \cdot y_1 + v_0 \cdot z_1$$

$$\rho_1 = z_1$$

3 equations per point

n point → 3n equations

→ 4 + n unknowns

$3n \geq 4+n \approx n \geq 2$

I can solve it.

For n unknowns, I need n equations.

If we also put extrinsic parameters:

3D world ↔ image

$P_i \leftrightarrow \Theta_i$

$n_{points} \rightarrow 3$ eq.

→ n unknowns → n: Scale factor for each point P_i

→ 5 unknowns intrinsic → 5: Intrinsic f_x, f_y, s, x_c, y_c

6 " extrinsic → 6: R + T

$3n \geq n + 5 + 6$

Non linear equations.

R → has cos, sin

If we don't have R, it's still not linear, it is bilinear.

$n \geq 6$ } If you give me 6 points from this world,

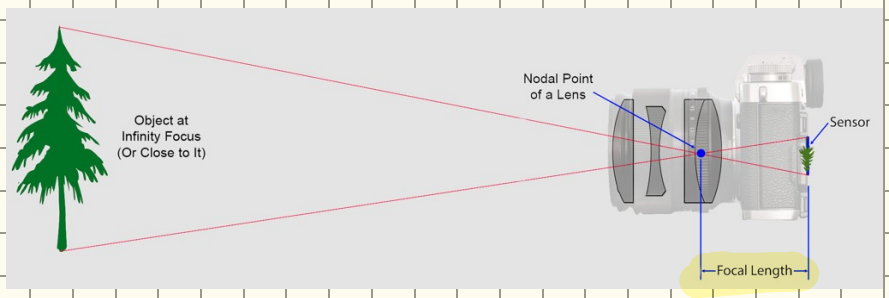
I can recover this.

↓

Vuforia works like this. It gives those points.

- The f is different for each point because it depends on the depth (Z_c) of camera coordinate system. ⟶ perspective, scaling based on depth

- Focal length doesn't depend on the location or depth of 3D points.
- Determined by camera's lens.



- Distance between camera's optical center (lens) and camera sensor.

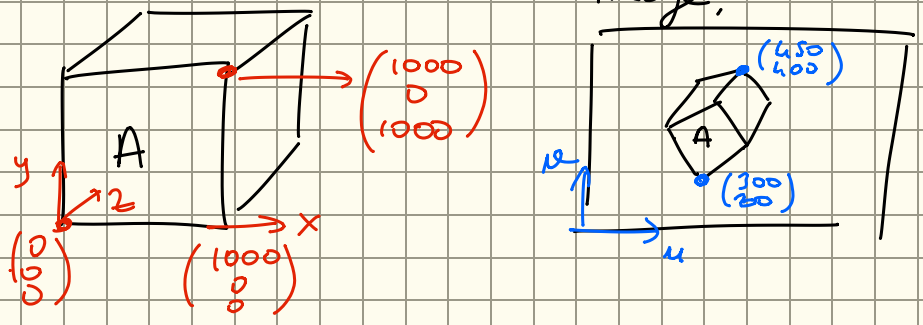
* How do we solve non-linear equations → Gradient-Descent.

↓

Initial guess is important.

- local min, max

* If we move the camera, just R and T changes. Problem has simplified.



$$G\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} -f_{sx} & 6 & x_c \\ 0 & -f_{sy} & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

↳ image üzerindeki noktalarımız.

Single image:

$3n \geq 11 \approx n \geq 6$ → both int. and ext. parameters

Another image:

$3n \geq n + 6$ → 5 is gone bcs we only moved the camera.

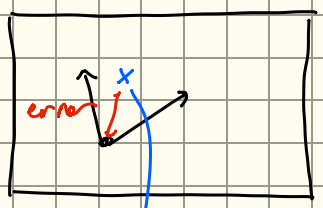
ss

↓

intrinsic

$n \geq 3$

3n equations with n+11 unknowns ?
(non-linear)



- Start with an initial guess and minimize iteratively.
(random number for 6 parameter)

My initial guess is wrong.

How much error → Distance of these points

Euclidean distance:

$$\left| \begin{pmatrix} u \\ v \end{pmatrix} - \begin{pmatrix} u' \\ v' \end{pmatrix} \right|^2$$

↓

$$\frac{1}{n} \sum_{i=1}^{n} \left| \begin{pmatrix} u_i \\ v_i \end{pmatrix} - \begin{pmatrix} u_i' \\ v_i' \end{pmatrix} \right|^2$$

The total error across all n points is averaged

2 What is the initial guess? → Its important bcs we have complex function.

n points in m images

$$5 + 6m \leq 2nm$$

Number of constraints

rotation and translation of camera changes image by image

2D projection → (x, y) → 2 equations per point (x, y)

m images, each n points

↳ images from different viewpoints can resolve ambiguity better.

↳ More images, simpler problem

2 If intrinsic is available → $6m \leq 2n \cdot m$

- intrinsic
- internal
- calibration

- extrinsic
- motion of camera.
- tracking

m	n
1	≥ 6
2	$\geq \frac{12}{4} \approx 5$
3	$\geq \frac{12}{6} \approx 4$
⋮	⋮
∞	≥ 3

More images reduce the # of points required to solve the system.

Objective function : (optimization)

$$\sum_{j=1}^{m} \sum_{i=1}^{n} \left(O_i - P(k_{ij}, P_i) \right)^2$$

Minimize sum of squared errors between observed and predicted points.

O_i : Observed data points in 2D images

$P(k_{ij}, P_i)$: Predicted data points (Projected)

P_i : 3D coordinates of ith point

k_{ij} : Parameters for the jth image (int, ex)

Sol

m = 2 images , n = 3 30 points

1. Observed Data (Given Inputs):

- $Q_{1,1} = (150, 200)$: The observed position of point 1 in **image 1**.
- $Q_{1,2} = (160, 210)$: The observed position of point 2 in **image 1**.
- $Q_{1,3} = (170, 220)$: The observed position of point 3 in **image 1**.
- Similarly, observed points for **image 2**:
  - $Q_{2,1} = (130, 180)$, $Q_{2,2} = (140, 190)$, $Q_{2,3} = (150, 200)$.

2-) Predicted points!

$$P(k_{ij}, P_i) = \left(\frac{fX_i}{Z_i} + c_x, \frac{fY_i}{Z_i} + c_y \right)$$

→ Projection matrix

3-) Initial guess:

- Random guesses! $f = 1000$, $c_x = 0$, $c_y = 0$ → Intrinsics

  Extrinsics → random

- 30 coordinates → given

4-) Compute errors:

$Q_{11} = (150, 200)$

$P(k_{11}, P_1) = (160, 190)$

Error $= (150-160)^2 + (200-190)^2$

$= 200$

5-) Total error:

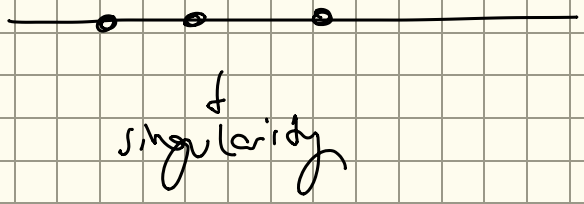
Sum over all points and images:

images → 2, points → 3

$$\text{Error} = \sum_{j=1}^{2} \sum_{i=1}^{3} (Q_i - P(k_{ij}, P_i))^2$$

Using our example data (simplified):

- **Image 1**: Total error = $200 + 150 + 180 = 530$.
- **Image 2**: Total error = $220 + 140 + 210 = 570$.
- **Overall Total Error**: $530 + 570 = 1100$.



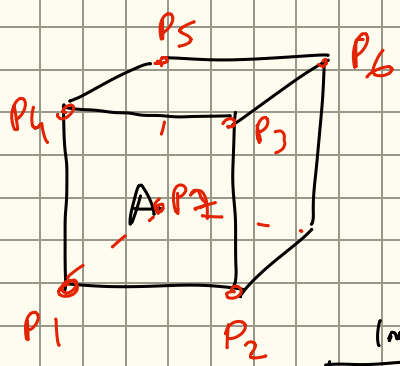
- Points on a line, impossible to understand depth.
- Problematic for 3D reconstruction or calibration
- This algo. will fail.

Updated Objective function:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} \left(\left(u_i^j - \underbrace{\frac{a^i P_i}{c^i P_i}}_{\text{Predicted projected points on 2D}} \right)^2 + \left(v_i^j - \frac{b^i P_i}{c_i P_i} \right)^2 \right)$$

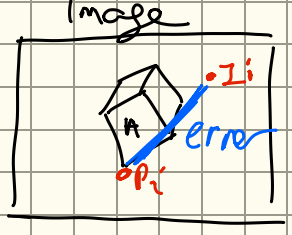
α a, b, c parameters → more robust and accurate calibration

α Who gives me point correspondences? → Big problem



$P_1 \rightarrow I_1$
$P_2 \rightarrow I_2$
$\vdots \quad \vdots$
$P_7 \rightarrow I_7$

What if a point associated wrong?



$$\frac{1}{n} \sum_{i=1}^{n} (P_i - I_i)^2$$

I will get non-zero. One of the points will give me a distance. } outlier

RANSAC:

1-) Take a random sample with 3 point matches.

2-) Estimate the parameters (R, T) using these points.

3) Check how many other points agree with the estimated model ('inliners')

4-) Repeat until the best model with most inliners found (majority)

- Vision algorithms often fail during point matching and reconstruction steps.
  - ↳ Image projection
  - ↳ Calibration
- Points are ambiguous
- Singularity: planes do not provide enough depth information
- Noise in data

- I don't know the world → (x_w, y_w, z_w)

  Can I do anything?

  Somehow I need to learn the world as well as the calibration.

↳ Now the problem is estimating both 3D world points and calibration parameters.

- n points observed throughout in m images.
- I_i^j → image location of i^{th} (unknown) point of j^{th} image.

  $2nm$ → num of equations

  Num of unknowns: Assume I have internally calibrated.

  $6m$ cameras (R, T)

  $3n$ points, (x_w, y_w, z_w)

  No intrinsic

n: # of 3D points

m: # images (cameras)

- for every point P_i observed in 1 image, we get 2 equations: x, y coordinate

  n points m images } 2nm equations

- Each P_i has 3 unknowns. $P_i = (x_i, y_i, z_i)$

  n point 3 3n unknowns

- Each camera has 6 extrinsic unknowns

  m images } 6m unknowns

$2nm \geq 6m + 3n$

(x', y')

n	m	?
3		$6m \geq 6m + 9$ ✗
6		$12m \geq 6m + 18 \approx m \geq 3$ ✓
4		$8m \geq 6m + 12 \approx m \geq 6$ ✓
⋮		

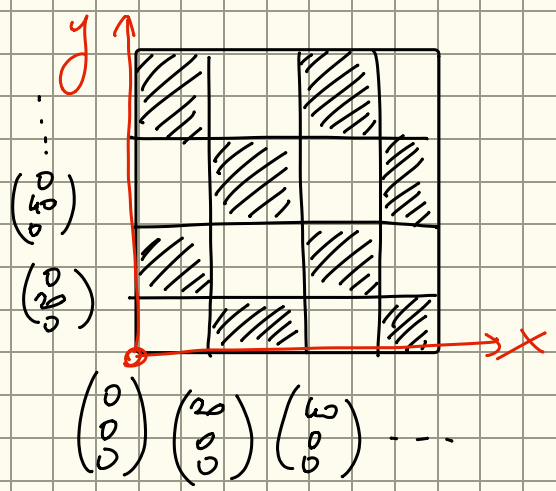
At least: $n \geq 4$ $m \geq 3$

} Now we have a solution

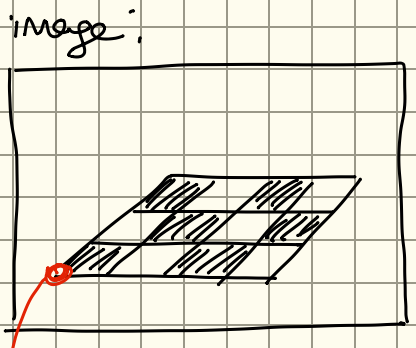
We can both recover the scene (3D points) $3n$

and calibrations. (extrinsic) $6m$

Checkerboard pattern:



$6 \times 6 = 36$ points



2×36 equations in 11 unknowns

(x, y)

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

↓ K is known

$$s \cdot K^{-1} \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

normalized coordinates

↓ 6 unknowns

x, y → 2 equation per point

$2n \geq 6$

$n \geq 3$

↳ How do we define these at least 3 points?

∝ Tracking

$2n \geq 6$ } Solving for extrinsic parameters (R, T)
$2n$: Each 3D-2D correspondence provides 2 equations (x', y')
6: Total unknown.

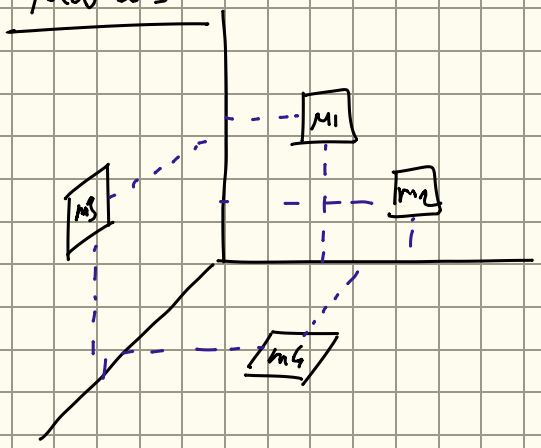
$3n \geq n+11$ } Solving for int and ep parameters

$2nm \geq 3n+6m$ } // // 3D reconstruction and Calibration
$3n$: 3D points
$6m$: extrinsic

Tracking → time dependent, when camera moves R and t changes.

I need to identify camera position or orientation at every frame.

Markers:



$m_1 \rightarrow x_1\ y_1\ z_1$

$m_2 \rightarrow x_2\ y_2\ z_2$

⋮ ⋮

Algorithm that finds all markers in the image → (u, v)

↓

$2n \geq 6$ if you have 3 or more markers will give you exact location of the camera.

Using scene instead of markers:

↳ Automatic recognition isn't easy., Computer vision problem.

Gİlkİ SORULARI

1.1. What is Spatial Continuum?

1.2. Difference between Natural User Interface in AR vs VR?

1.3. Difference between Augmented Reality vs Diminished Reality?

1.4. Difference between Optical AR and Video See-Through AR?

1.5. Explain how to go from task domain to spatial axes?

1.6. How would you define space in 3D UI?

2. You are given a projector and a camera AR system. Projector and camera share a really large field of view. (Camera's field of view intersects with projector's FOV.) Propose a solution to calibrate the camera with the projector. You are allowed to place one object somewhere in the field of view once. Explain and propose details about algorithms of computer vision and to calibrate those. Provide algebraic details of your work.

→ Calibration → Md, ep

3. Questions about usability heuristics

3. 1. Match between system and the real world; why we match or don't match system and real world?

3.2. Explain and provide details to why we use recognition rather than recall in AR and VR systems.

3.3. Why consistency and standards are important?

4. You are given a 3d scene from 2 different cameras and you know points on scene
p1= [1, 0, 0]t
p2= [0, 1, 0]t
p3= [0, 0, 1]t
p4= [a, b, 0]
p5= [1, 1, 0]
p6= [c, d, 1]
Given orthogonal space find a, b, c, d.

TELAFİ SINAVI:

1.1. What does motion sickness have to do with VR? How does it affect VR experience?

1.2. How does a VR system map a large virtual space into a smaller physical space?

1.3. How does "match between system and the real world" heuristic can be used in a VR application. Give examples.

1.4. What are the similarities and the differences between a projector AR system vs a cave-type VR system?

In an AR application, you observe significant errors in the 3D reconstruction due to incorrect point correspondences. Explain how you would use RANSAC to identify and remove these outliers. Provide an example with 4 points and 2 images.

1-) Make a small subset of points

2) Model estimation: find parameters

3-) Inlier check: Calculate error.

4-) Repeat until find highest num of inliers

$$\frac{1}{n}\sum_{i=1}^{n}\left|\begin{pmatrix} u_i \\ v_i \end{pmatrix} - \begin{pmatrix} u_i' \\ v_i' \end{pmatrix}\right|^2$$

$$\text{error} = (P_i - I_i)^2$$

- **Point correspondences:**
  - Image 1 points: $P_1 = (100, 200), P_2 = (150, 250), P_3 = (300, 400), P_4 = (500, 600)$
  - Image 2 points: $P_1' = (102, 198), P_2' = (152, 248), P_3' = (305, 395), P_4' = (700, 800)$

outlier

Step 1: Random Sampling

- Randomly select 2 correspondences (minimal subset for the fundamental matrix):
  - (P_1, P_1') and (P_2, P_2').

Calculate distances.

Non-zero will be outlier

- **Identified Inliers:**
  - $(P_1, P_1'), (P_2, P_2'), (P_3, P_3')$.
- **Outlier:**
  - (P_4, P_4'), which caused significant errors

What are the minimum requirements (in terms of the number of points and images) for recovering both the scene structure and camera calibration? Derive the relationship between n and m for solvability.

3D world points

extrinsic

Each point has X, Y, Z unknown

6 unknown per image (camera pose)

n points

m image

n point has $3n$ unknown

m image has $6m$ unknown

2 equations per point

$2n \cdot m$ equations

$3n$ unknown $6m$ equations

$3n + 6m \leq 2nm$

Inverting Perspective Function

$$K = \begin{bmatrix} 1000 & 0 & 320 \\ 0 & 1000 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

$(u, v) = (400, 300)$

$z = 5$ meters

1-) Normalized Coordinates → removes the effect of focal length, pixel scaling etc. making 2D coordinates independent of camera internal parameters.

Camera's (0,0) is origin

$$K^{-1} = \begin{bmatrix} 0.001 & 0 & -0.32 \\ 0 & 0.001 & -0.24 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cdot K^{-1} = \begin{pmatrix} 0,08 \\ 0,06 \end{pmatrix}$$

2-) Compute 3D coordinates:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = z \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$= 5 \cdot \begin{bmatrix} 0,08 \\ 0,06 \\ 1 \end{bmatrix} = \begin{bmatrix} 0,4 \\ 0,3 \\ 5 \end{bmatrix}$$

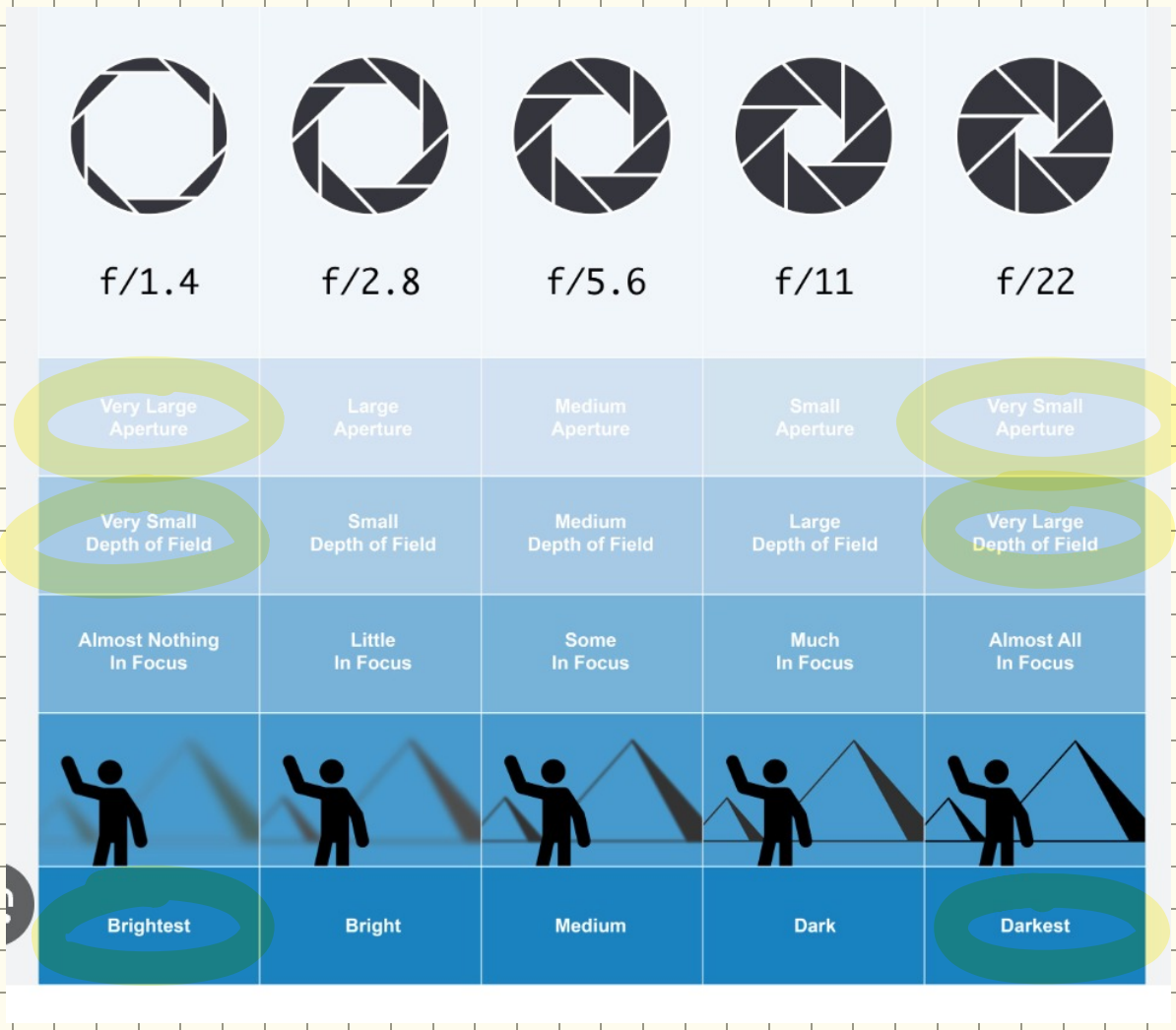
Without z, solution would be ambiguous as a single 2D point corresponds to infinite # of points along a ray in 3D space.

Q Difference between perspective projection and rigid transformation?

3D → 2D

3D → 3D

α



Depth of field → Hole (aperture) 크기

Focal length → Field of view

Short f, wider fov, smaller object