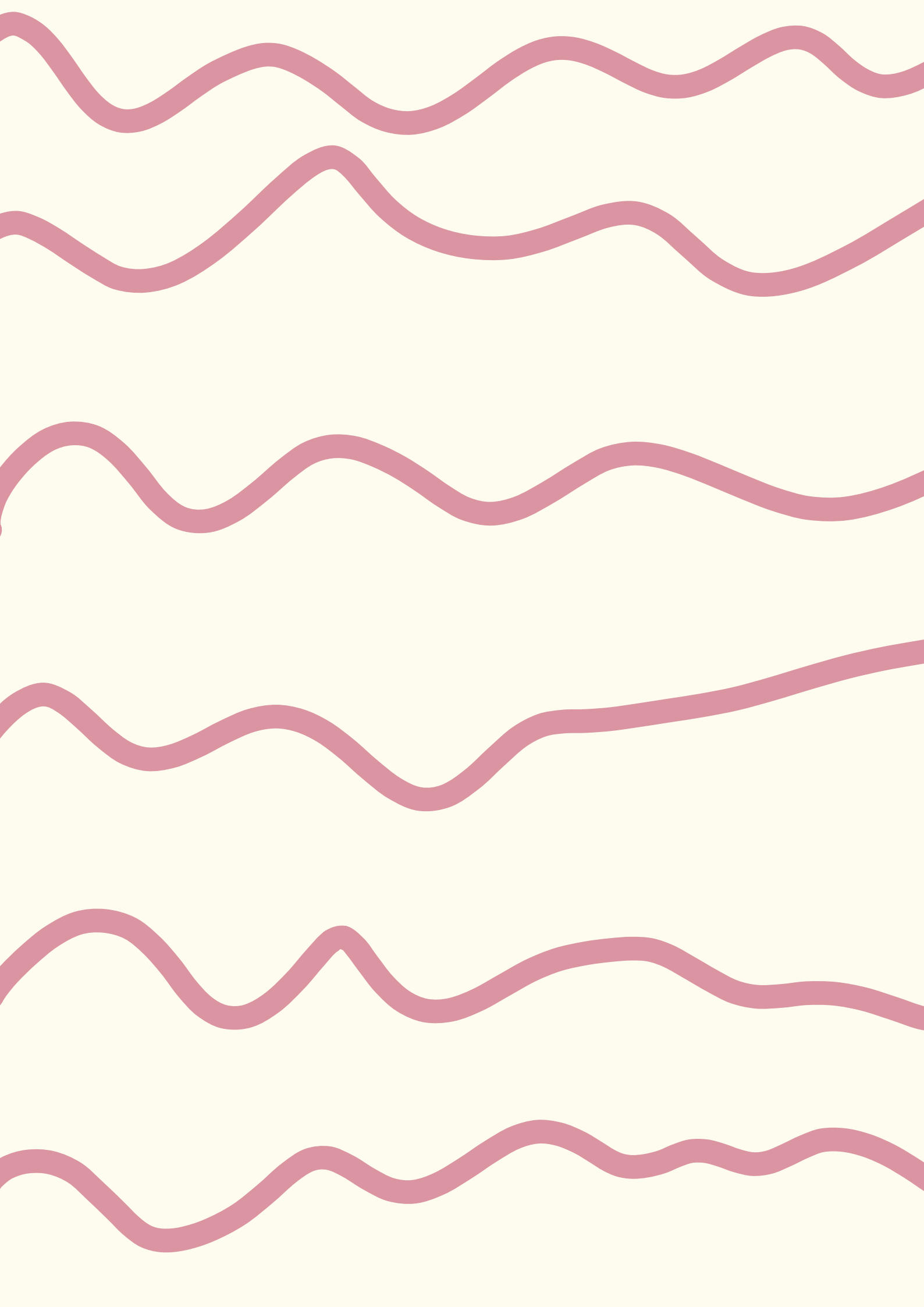
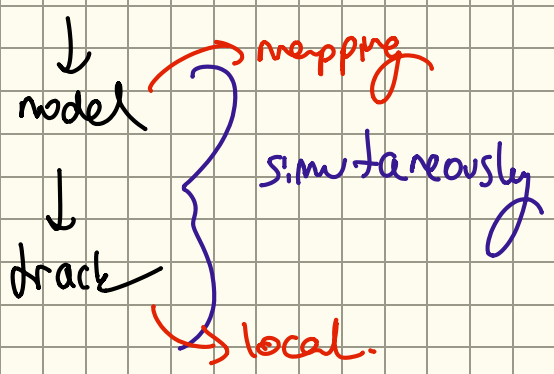


2021-11-02 Part 2 - Camera Calibration

Unknown env.

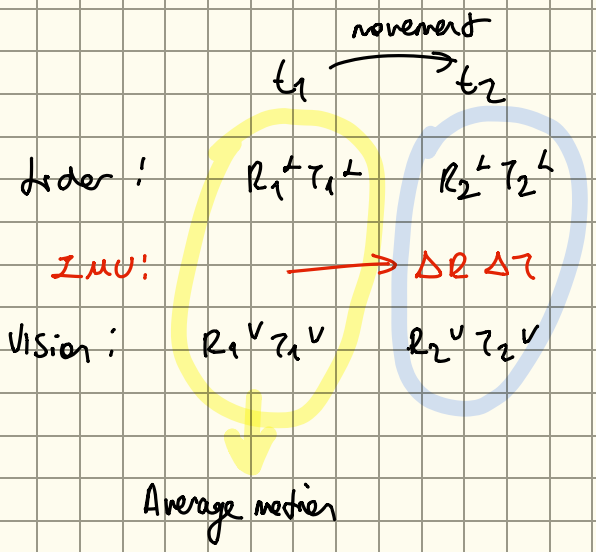


Robotics

↓

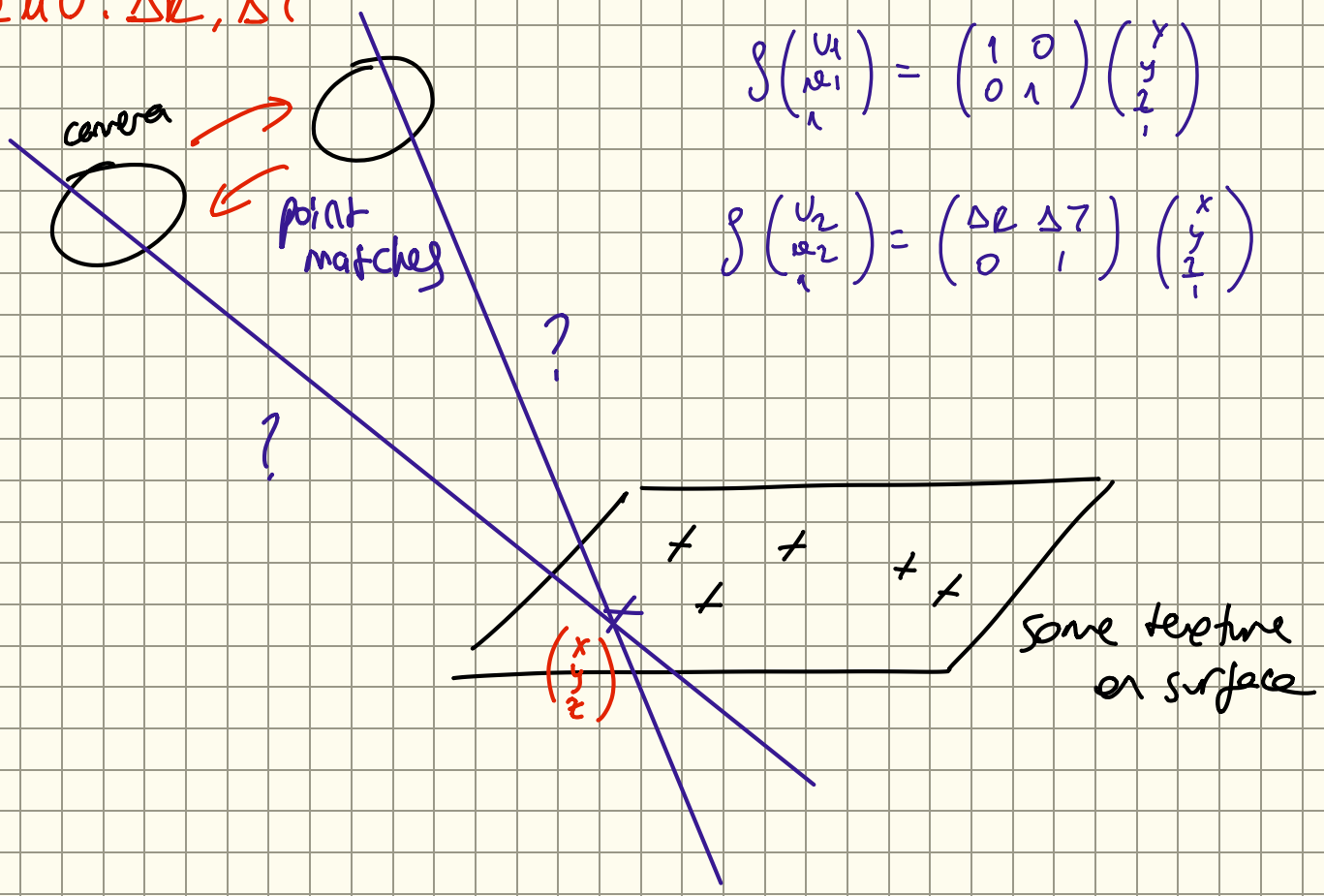
SLAM

Sim. localization & mapping



- **LiDAR** provides the **raw point cloud data.**
- **SLAM** is the overall process that uses this data to **build a map and localize** the robot.
- **ICP** is one of the **core algorithms** inside SLAM to **refine point cloud alignment.**

IMU: ΔR, ΔT



$$s\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$s\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \Delta R & \Delta T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Unknowns:

3 per point (x, y, z)

$\underline{\underline{3n}}$ for n point

6 for camera

Equations:

2 equations per point (u, v)

$\underline{\underline{2n}}$ " "

$$2nm \geq 3n + 6$$

$\downarrow$ 2 camera

$$4n \geq 3n + 6 \approx n \geq 6$$

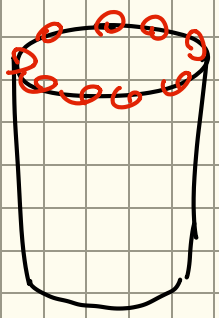
If you give me 6 points from 1st and 6 " from 2nd image, I can solve.

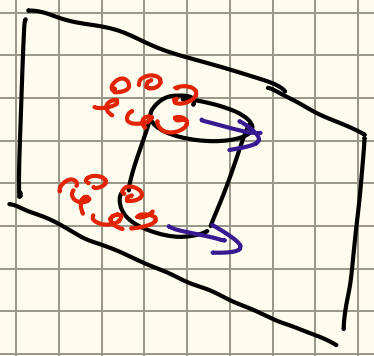
- If $z = 0$: 2 unknown per point (x, y)

$$4n \geq 2n + 6 \approx n \geq 3$$

ARCore, ARKit

2021-11-09 - Camera Tracking





I can actually register them.

Model-based tracking: (CAD)

$R, T \rightarrow$ start with known locations

$+$

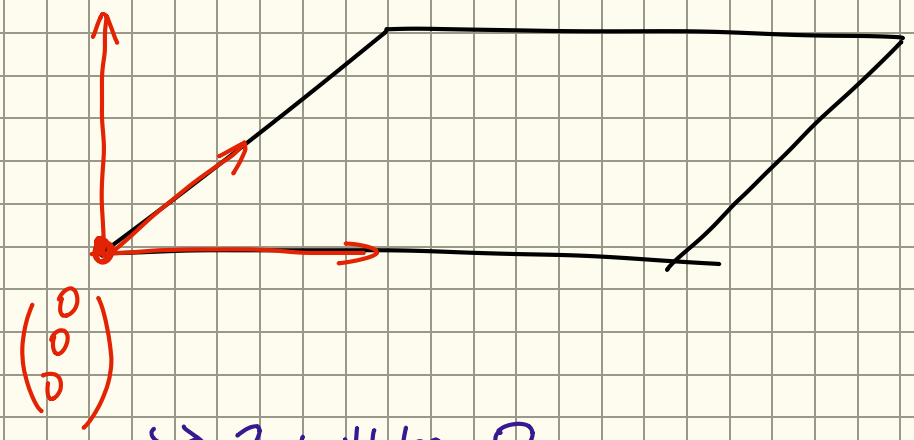
$\Delta R, \Delta T$

Assume that this is the first time you're seeing the env.

We've discussed:

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

We choose a coordinate system to parallel of surface:



$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

↳ z will be 0.

↓ Simplification

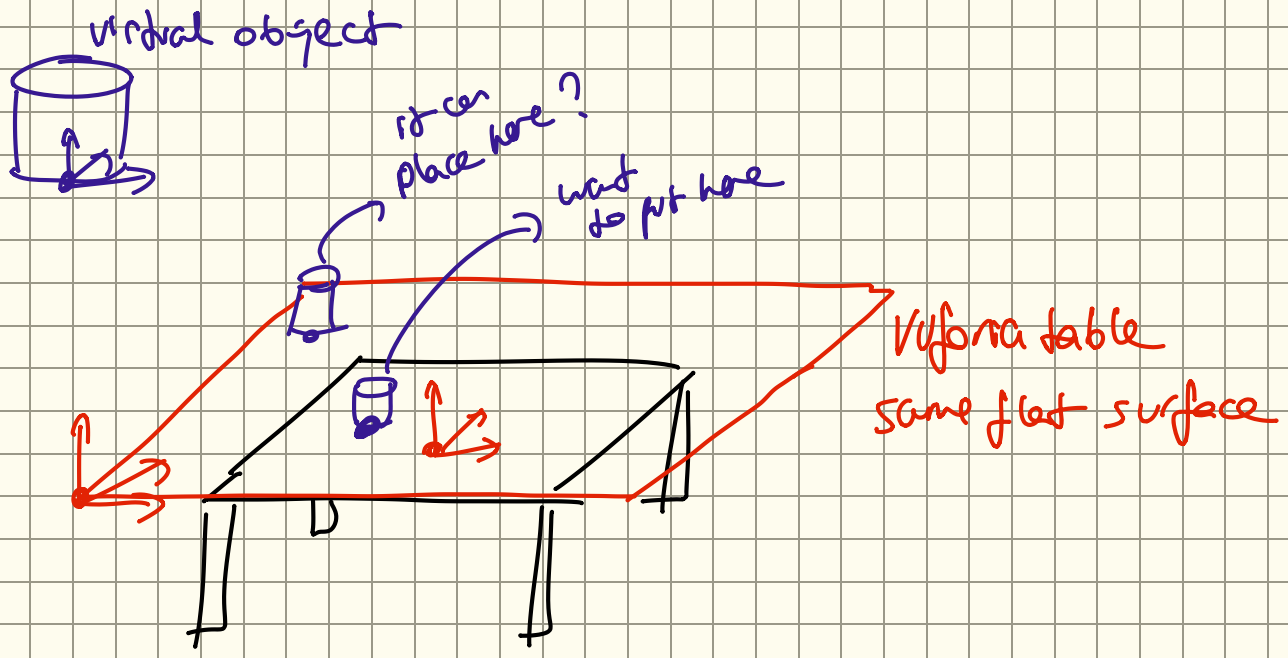
$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = M_{3\times 3}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Affine transformation (Homography): If your world points are on a plane, you don't need to worry about projection formula.

In robotics → SLAM

// visual → SFM

How can I place the virtual object on a real world surface?



- Vuforias and mine coordinate system isn't registering. How do I register it?

When Vuforia or Arkit detects a flat surface in real world, it assigns a coordinate system to that surface.

↳ If we want to put virtual object, since we don't know where the AR systems coord. system origin, the object can place different. So we need to make alignment.

One solution → Raycasting } find intersection between ray and surface.

2021-11-09 – 3D Camera and Tracking

- Planar structure recovery & tracking (Vuforia etc.) → IMU
- 3D SFM

$R_0 T_0$ $R_1 T_1$

→ IMU

↳ will fail in very slow motion

- Image target + IMU
- 3D Sensing → Scanning → SFM/SLAM → not so robust!

↳ Structured light scanners

↳ Lider

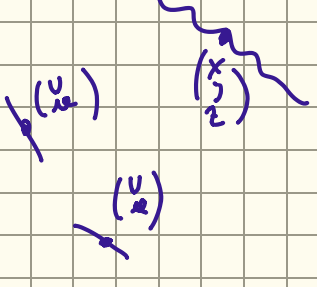
To find the surface:

1st image $s\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

2nd image $s\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \Delta R & \Delta T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

4 equations in 3 unknowns.

↳ someone gave the ΔR and ΔT



Can I make it better?

Projector AR

Other tracking methods;

- Camera → inside out tracking } Camera moving, I am finding locations of what I observing

→ outside in tracking } Sensors stationary

↓

VR

"Motion Sickness"

2021-11-16

- for tracking with 2 cameras, there is a problem!

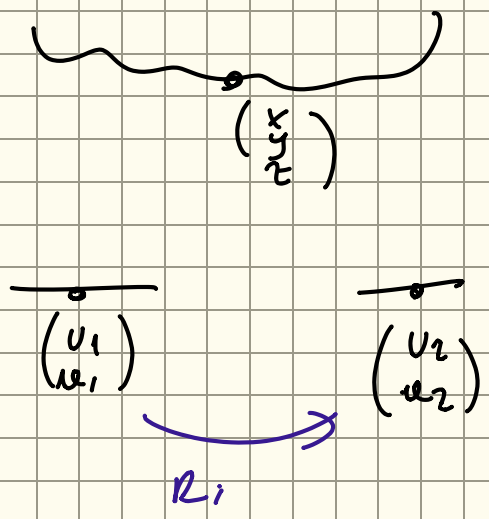
Assume first camera to be world center

$$s\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$s\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \Delta R & \Delta T \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

4 eqs
Unknowns
↓
I can recover my scene.

Stereo:



Physically

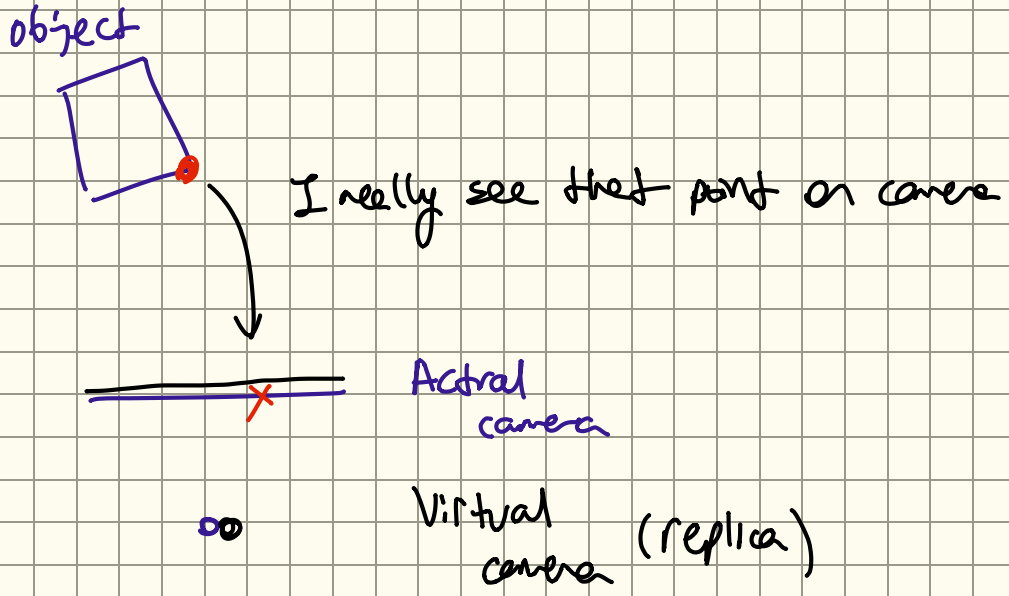
- A projector casts rays on the actual env.
  - ↳ Sends light out to show images
- A camera does inverse raycasting
  - ↳ Collects light to capture images.

- A virtual camera ≈ a physical camera

"mimicking"

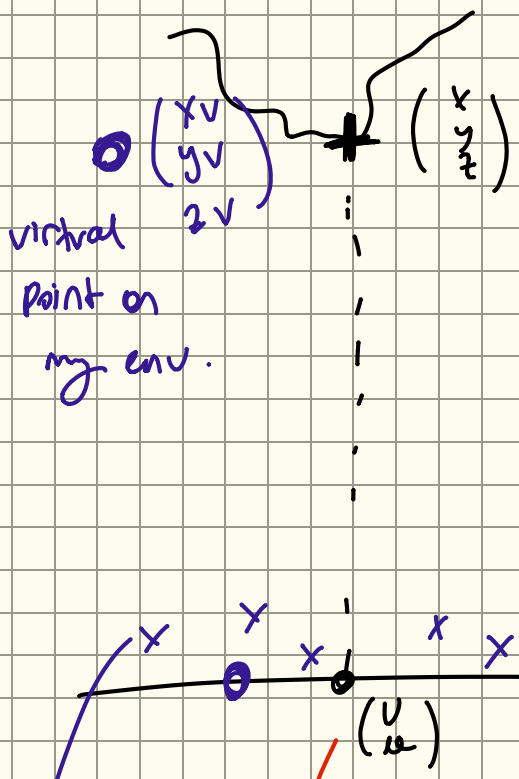
(video seethrough → actual camera
optical " → eye + screen)

"Augmentation"



- The actual camera shows the real-world scene.
- The virtual camera inside the computer tries to replicate the real camera view.
- You need to **align** the virtual objects with the real-world objects so that what you see on your screen **matches reality**.
- If done right, **the virtual object will appear exactly on top of the real object** in your camera view — that's **augmented reality**.

Projection: If you know where Corner A in real-world, you need to place the virtual Corner A in the same position in the camera view.



R, T : transformation from world to camera.

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

- I assume, I <u>callibrated</u> it;
  - 2D points of the marker in the image.
  - System knows Real world 3D positions of these points on the marker.
  - It can calculate K, R, T.

Video Seethrough:

- Now I want to place sepra object to my env.

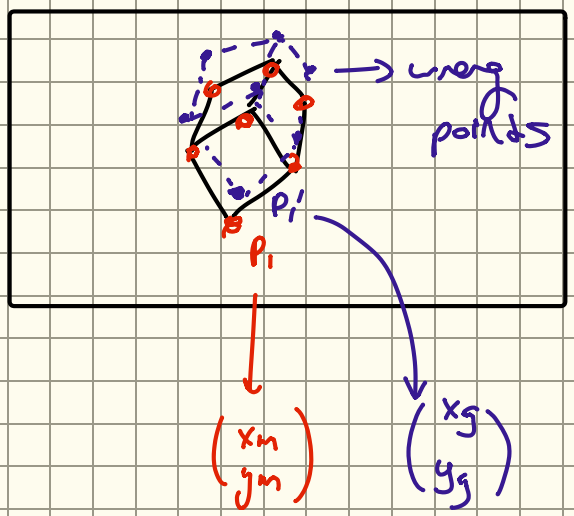
- With Vuforia, you are rendering your 3D and then you're overlaying that onto the image itself.

1. The camera captures a live video feed.
2. The system renders virtual objects in the virtual world.
3. The system overlays the virtual objects on top of the video feed.
4. Visibility analysis ensures that only the visible parts of the virtual objects are shown.

- Visibility analysis

→ Once the camera calibrated, the system knows how to project virtual objects into the real world.

→ In tracking, the virtual object stays in correct place even the camera moves.



world point (x, y, z)

initial guess R_0, T_0, Q_0

↓ project

(x, y) image

- How can I measure the error? → Distance

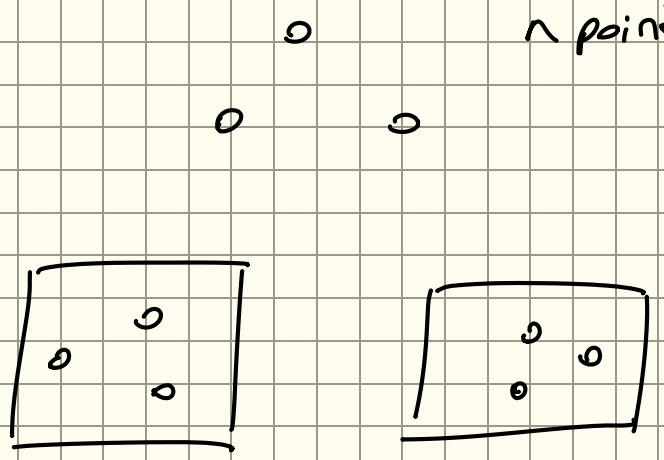
$$MSE = \sum_{i=1}^{n} \left| \begin{pmatrix} x_m \\ y_m \end{pmatrix} - \begin{pmatrix} x_g \\ y_g \end{pmatrix} \right|^2$$

↓ Minimize the error

Optimization problem

$$\arg\min_{R,T,Q} \frac{1}{n} \sum_{i=1}^{n} \left| \begin{pmatrix} x_m \\ y_m \end{pmatrix} - \begin{pmatrix} x_g \\ y_g \end{pmatrix} \right|^2$$

Min # of points and images to calibrate camera in 3D space:



2 types of unknowns:

x, y, z
+
camera params

# of equations: $2(2 \cdot n)$

→ image num

# of unknowns: $3n + 2 \cdot (5+6)$

↳ x, y, z → 3 equations per point

$4n \geq 3n + 2(5+6)$

$n \geq 5 + 2 \cdot 6$ } Fixed 5 intrinsic

↓
image

More images?

n points
m images (camera) }

unknown: $3n + m(5+6)$

or

$3n + 5 + 6m$

equations: $2 \cdot n \cdot m$

↳ u, v → 2 eq per point

$2nm \geq 3n + 5 + 6m$

u: ---- } per point
v: ----

What are the possibilities?

n	m	Solution	
1	1	×	
?	1	×	
1	?	×	
2	?	×	
3	?	×	
4	?	✓	$8m \geq 12+5+6n$

SLAM

- Recovering parameters
- Using some form of SLAM algorithms in AR.

↳ Robust solution → RANSAC

↳ using subset of points

3D - 2D + Internal available → R, T

n point matches with 0% error in matches

n = 100 → 10 outliers

$$\arg\min_{R,T} \sum_{i=1}^{100} |\text{Projected point}_i - \text{Objected point}_i|^2$$

↓ simplify this

Subset:

$$\arg\min_{R,T} \sum_{i=1}^{3} \,''\,\,''^{\,2}$$

If the solution is correct what happen?

- It should answer, correct match
- 50 of them should be good → inliers.

$$\text{So} \rightarrow \sum_{i=1} \text{only for inliers}$$

- If majority is matches, take that as a solution.

Reducing # of unknowns in 3D reconstruction and camera calibration:

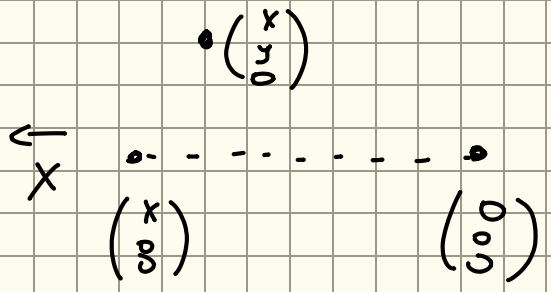
1-) $3n$: # of unknowns for n points.

↓

$3(n-1)$ → I set one of the point to my camera coordinate system. with world coordinate system.

↳ Fix one of them as origin $(0,0,0)$

2-) Assume they have same $z=0$: All points lie on same plane.



$3(n-1) - n$

↓

$z=0$ for all n points

$= 2n - 3$

No z axis!

$3n - 3 - 2 - (n-2)$

↳ x,0,0 same x axis

↳ rest of them have $z=0$

↳ 0,0,0 at each 1 point

generic

↓

$2n - 3$ } # of unknowns for 3D points.

$2n - 3 + 12$ ← cam. parameters

$2n - 3 + (12) \leq 4n$ → for 2 image

$3 \leq 2n$

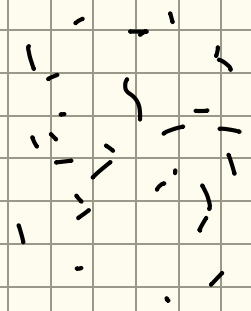
$n \geq 5$

2 x n eq

↓

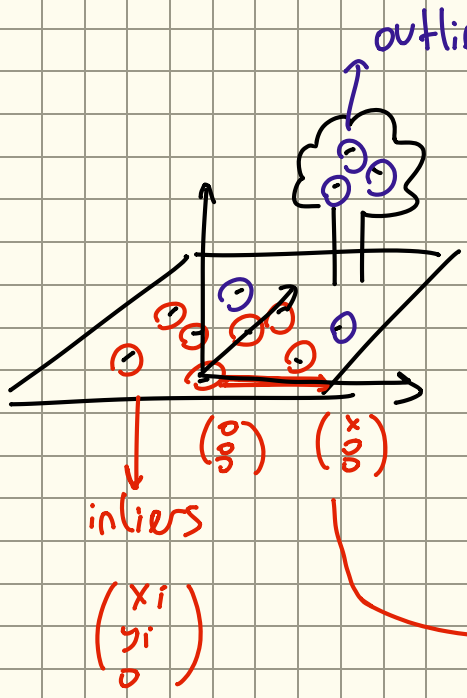
4n

Environment:



mostly 2D planar
with some of plane points

Plane recognition
- 1-) Center it at the worlds origin
- 2-) Define new coordinate system



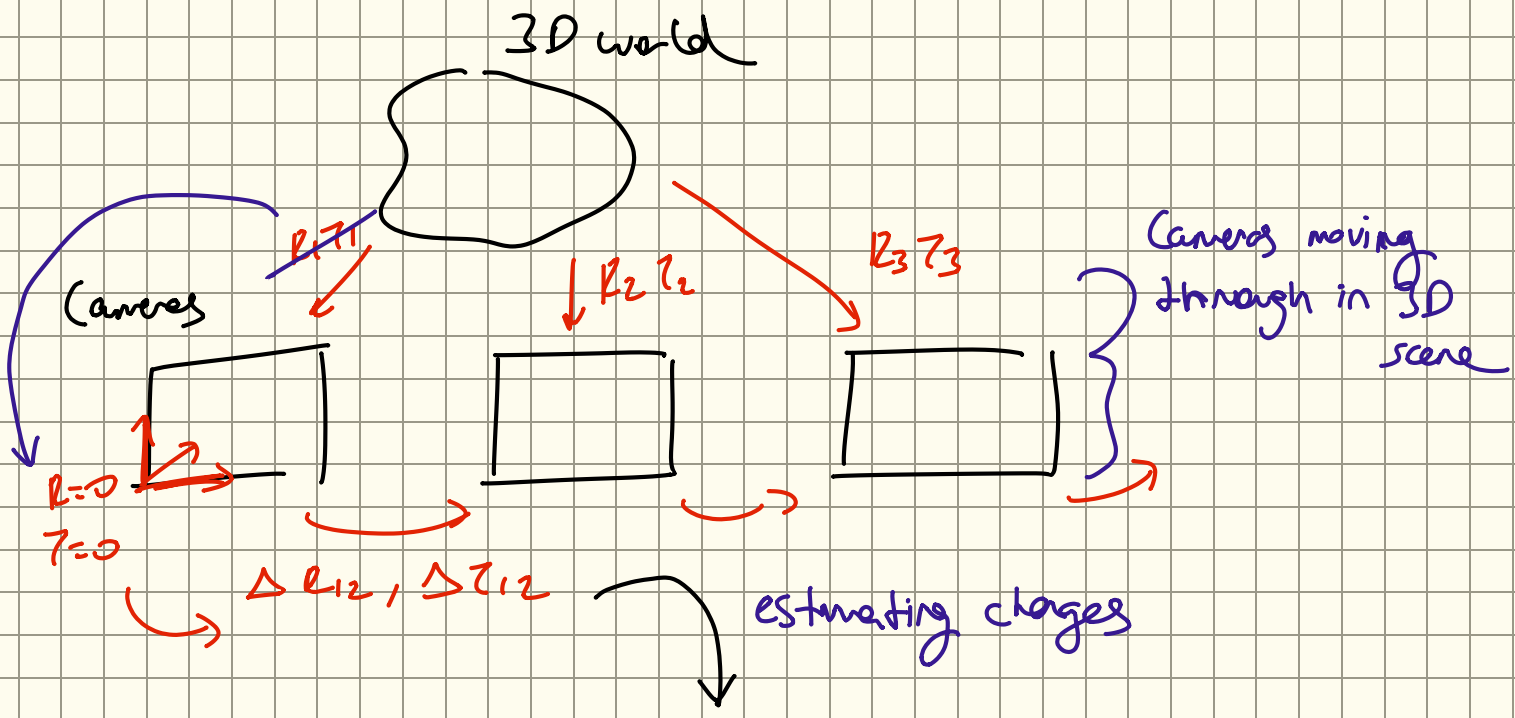
2 possibilities!

1-) Center 1 camera to my world.
↳ (0,0,0) one

2-) Define 1 coordinate system arbitrarily.

↳ You don't need the exact position of plane.
↳ RANSAC will estimate the planes position and orientation

↳ With RANSAC, I can recover the plane. even there are outliers.



∝ In reality, we can know this.

Traditional → SfM

✱ first breakthrough

∝ IMU sensors → acceleration → they give app. position where 2nd or 3rd camera is without seeing the image. Without IMU, it would be so complex.

- inertial
- magnetic

∝ Estimates relative position of 2nd and 3rd camera

✱ Second breakthrough: I can have some other sensors.

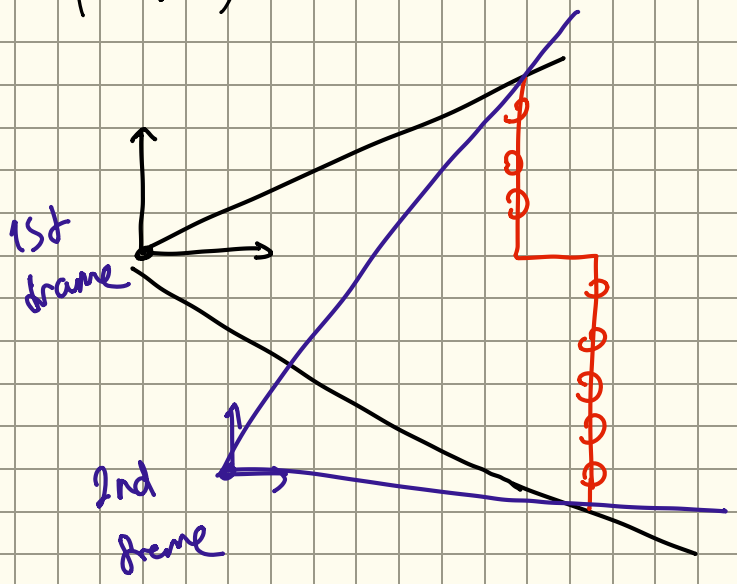
lidar: 200 samples

- lidar: you need textured scenes.

provides → ∝ Point cloud data

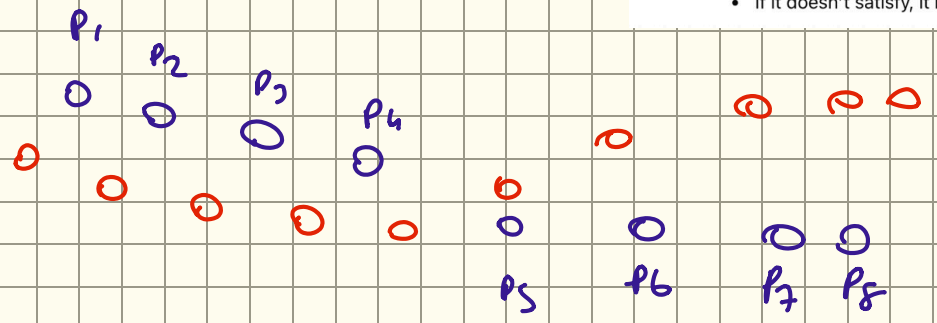
A 3D model from multiple images, you capture point clouds from different camera positions. Aligning 3D point clouds:

(point cloud data) per frame



Task: fit a plane through these points.

3. **RANSAC for Plane Detection:**
   - **Randomly pick 4 points** from the point cloud.
   - Solve for the plane equation.
   - Check if other points in the cloud satisfy the plane equation:
     - If a point satisfies $ax + by + cz + d = 0$, it is an **inlier** (point on the plane).
     - If it doesn't satisfy, it is an **outlier**.



A plane in 3D:

$ax_1 + by_1 + cz_1 + d = 0$

$ax_2 + by_2 + cz_2 + d = 0$

$ax_3 \dots$

$ax_4 \dots$ → (x_i, y_i, z_i)

- 4 points at random
- Find a plane

↓

If $0 = 0$, my point is on plane.

- If they fits, we can look other points.

- 4 points, then 10 points, then 15... } Expanding the first.
(or 3)

$\binom{n}{3}$ → P_1, P_2, P_3

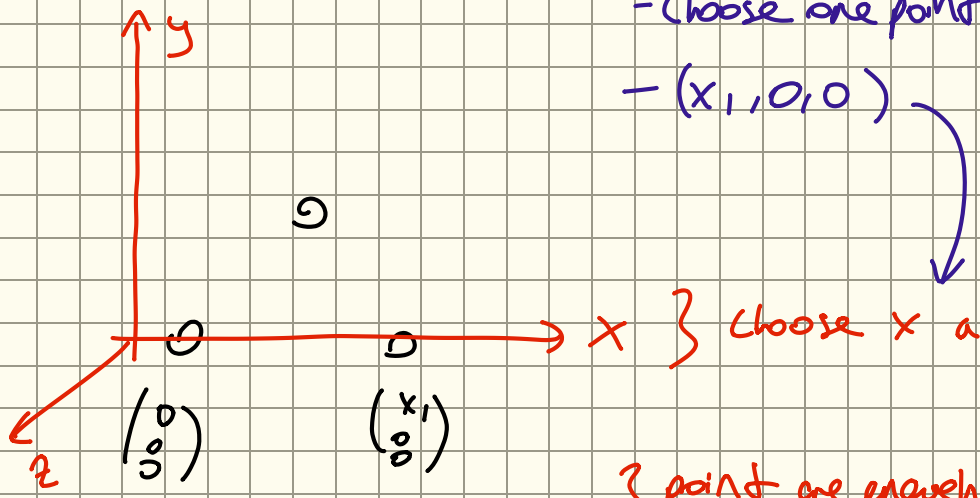
↓

initial plane equation

↓

apply plane eq. to all other

- if fit → label inliers
- does not → outliers.

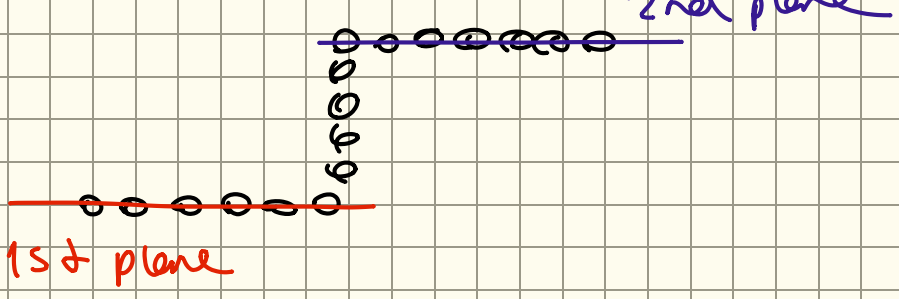


- Choose one point as origin
- $(x_1, 0, 0)$

} Choose x axis arbitrarily.

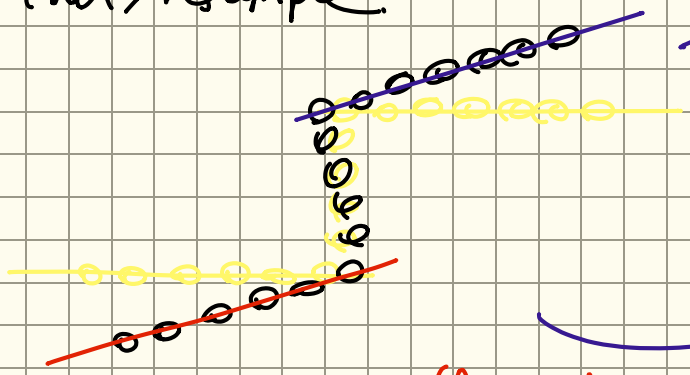
} 3 point are enough to define a plane.

- Once you have the first image, you have still 2nd image



Align point clouds from different frames:

- Then, resample!



→ When I move the camera, it captures different point clouds from different positions

(Direct alignment not possible) → We can't make rigid trans. directly to this because they are different points.

We should align on same plane first.

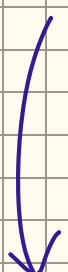
But if they dense enough, we can do that?

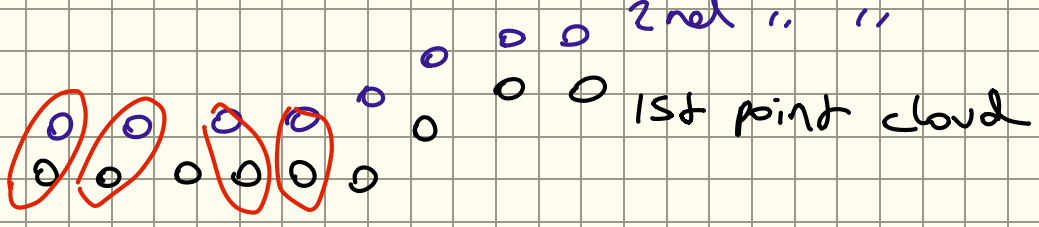
"ICP" (Iterative closest point)

$$\arg\min_{R,T} \sum \|A_i - B_i\|$$

Align 2 point clouds by minimizing the distance between corresponding points.

∠ To make ICP works, align planes first.





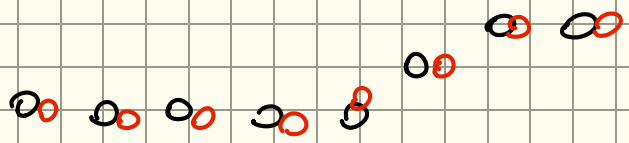
↳ Planes now aligned but not exact.

↳ I am not expecting 0 error, but as good as possible.

↳ Look closest one and align with them.

↳ Then make rigid transformation.

↳ Once the alignment has done:



1. **Plane Fitting**: Detect a plane from 3D point cloud data using **RANSAC**.
2. **Point Cloud Alignment**: Align planes from different frames.
3. **ICP**: Use **Iterative Closest Point** to refine the alignment by minimizing the distance between corresponding points.
4. **Sensor Fusion**: Combine data from **LiDAR**, **VSLAM**, and **markers** to improve accuracy in 3D mapping.

Plane fitting → Point cloud alignment (Rigid trans. R, T) → ICP (Refine the alignment)

Plane fit + ICP → bigger point
(initial R, T)

} SLAM based AR systems
IMU → to build the map of env.

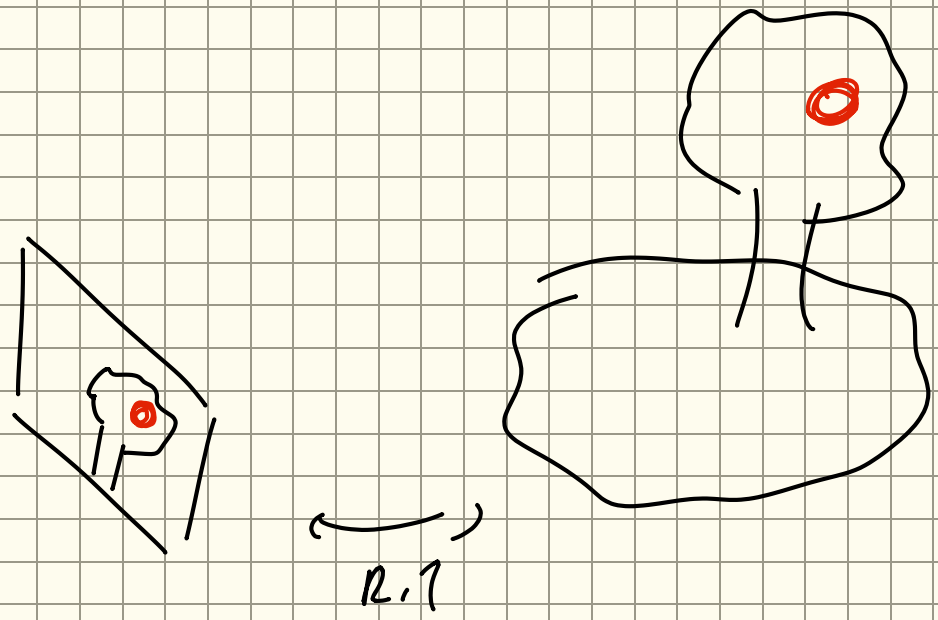
Lider + VSLAM (IMU) + Marker
(Captures 3D point cloud) (2D/3D) (If you lucky) (Improve alignment)

Step	Goal
Plane Fitting	Find a flat surface in the point cloud
Point Cloud Alignment	Align planes across frames
ICP Refinement	Refine the alignment by minimizing point distances

Moving Camera ?

Video seethrough is basic -> Vuforia

SLAM

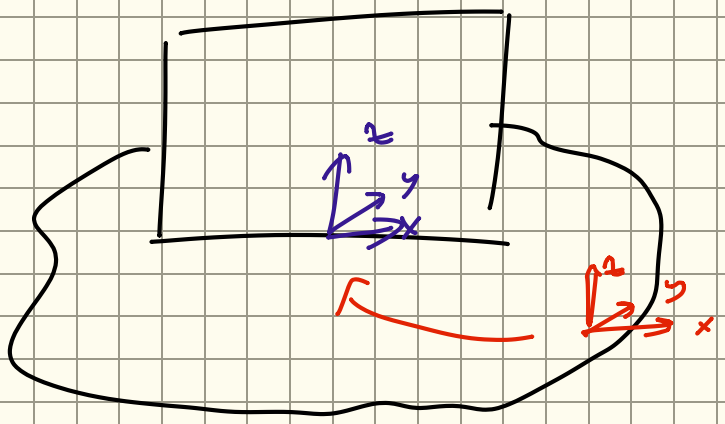


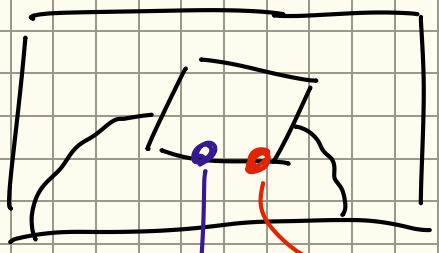
$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \pi \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Video Seethrough:

Overlaying virtual objects on a live video feed.

Prepare your environment to alignment:





② Pick another location

Now I've done the alignment.

① $\begin{pmatrix} u \\ v \end{pmatrix}$ Pick this location must $\perp$ ground

find $\begin{pmatrix} x \\ y \end{pmatrix}$ on ground

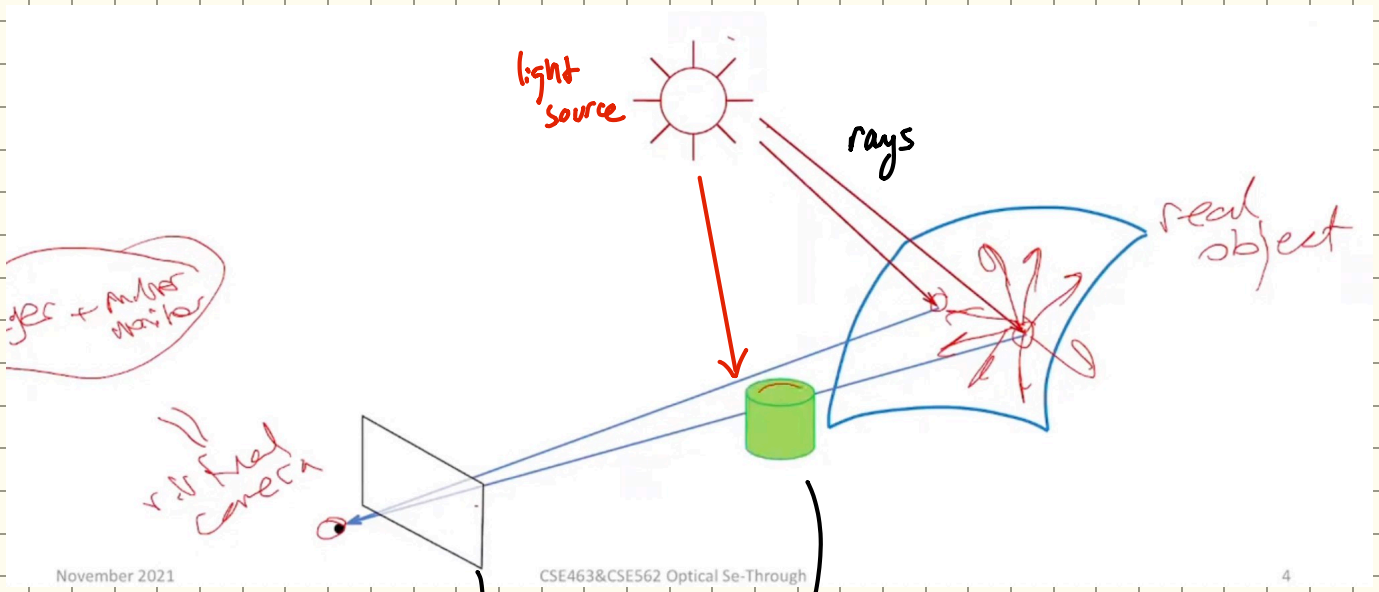
Carry the coord. system to $\begin{pmatrix} x \\ y \end{pmatrix}$

$$S\begin{pmatrix} uw \\ vw \\ 1 \end{pmatrix} = \pi \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \text{is fixed}$$

1 point isn't enough for alignment.

If you know 2 points, you can find camera R, T. So that we can transform real world coordinate to virtual coordinate.

2021-11-23 - Part 1 - Optical Seethrough

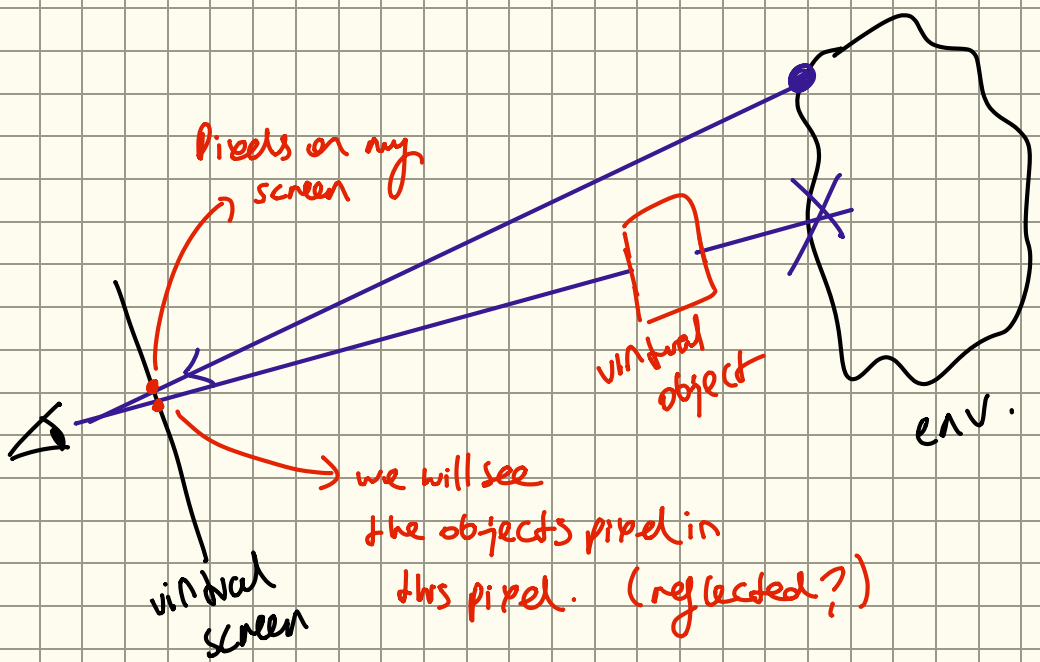


transparent

virtual object

Identifying where light source and how they effect the scene is difficult

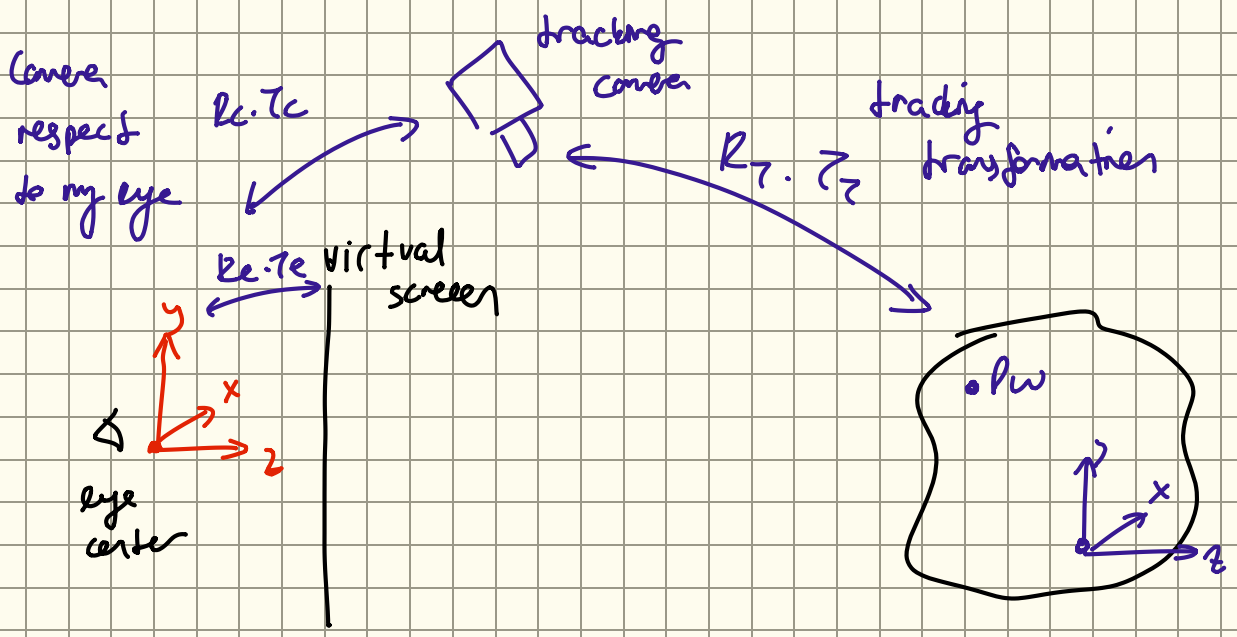
↳ there are multiple light sources



Passing through rays → Black pixel, the light coming from env. is going to pass through. I won't see anything virtually.

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{K_{3\times3}\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}_{\text{eye position}} \cdot P_w$$

object pos.



∠ I need to know where the world is with respect to my camera (eye)

∠ $R_c \cdot T_c$ and $R_e \cdot T_e$ won't change once they calibrated.

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = k \cdot \begin{pmatrix} R_e & T_e \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_c & T_c \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_T & T_T \\ 0 & 1 \end{pmatrix} P_W$$

Fixed (Assume fixed) Fixed changing

$R_e T_e$ → Head Mounted Display → $R_c T_c$ → Camera → $R_T T_T$ → W

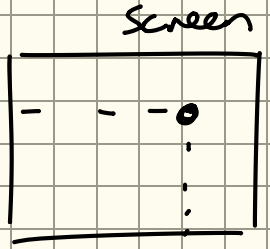
How I am going to calibrate?

camera view | model view

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = k \cdot \begin{pmatrix} R_e & T_e \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_c & T_c \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 & T_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- $s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$: we know (from markers?)
- $k \cdot \begin{pmatrix} R_e & T_e \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_c & T_c \\ 0 & 1 \end{pmatrix}$: $\Pi_{3\times4}$ → 11 unknowns
- $\begin{pmatrix} R_1 & T_1 \\ 0 & 1 \end{pmatrix}$: changing all the time
- $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$: Assume we know this

→ I need 6 point projections (12 equation) to solve this



To create them, I may ask to use to align it also in somewhere else.

→ Now we can find 11 unknowns → Calibration ✓

$2n \geq 11$

↓ Simplify

∝ Small changes for R_e T_e and k. → I cant move HMD that much.

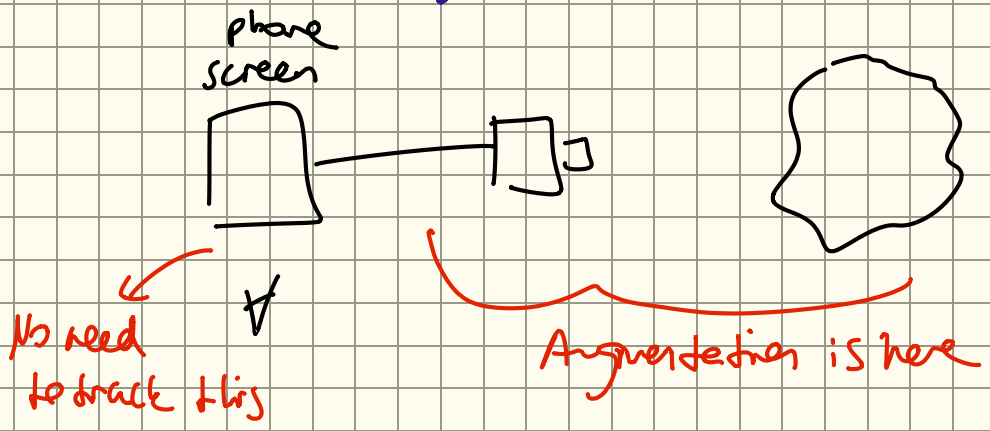
∝ R_c T_c always fixed.

* Average calibration for general use, or once calibrated then reusing again.

∝ Human vision FOV → 100°

∝ AR/VR FOV → 20° { Easier to calibrate; Limits the usability } → Hololens is limited in first times.

Video seethrough way:



Video: $s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \cdot P_w$

"

Optical: $s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot (\,) \cdot (\,) \cdot \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \cdot P_w$

"

Projection part

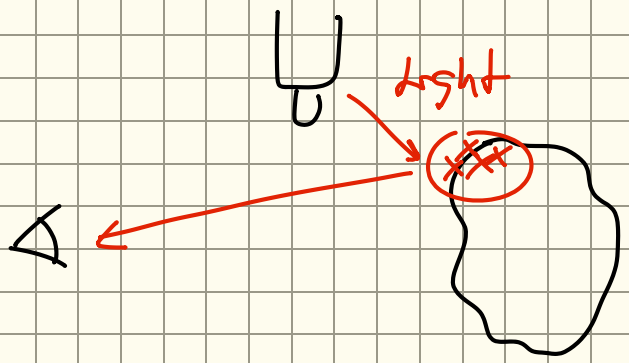
How to 3D world projected to 2D

FOV, aspect ratio ...

world view part.

Camera position and orientation

2021 11 23 - Part 1 - Projector AR



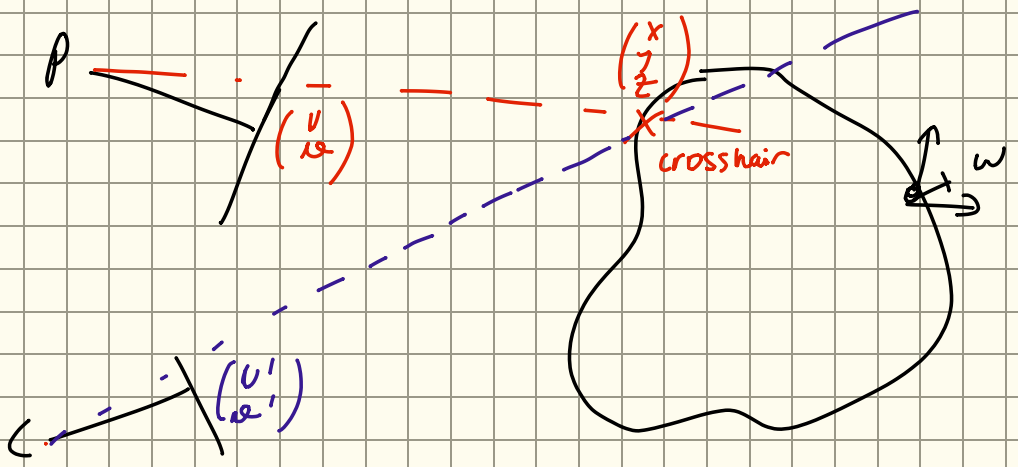
Optical, Video

- Augmented content is virtual
- Requires image plane calibration to align virtual objects into real world.
- Uses a virtual display (image plane)

Projector

- Aug. content is physically projected on the object.
- No image plane calibration needed, the projector modifies the physical object directly. (Cast light directly to object)

Projector AR Calibration: The crosshair appears on 3D surface as a 2D projection.



To observe the crosshair in the image. } Compare actual vs expected and adjust transformation.

Camera

$$s\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = K_c \begin{pmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2 eq.
6+5 unknown

known

Projector

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K_p \begin{pmatrix} R_{wp} & T_{wp} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2 eq.
6+5 unknown

known

C → 6+5

P → 6+5

Each point → 3n

eq per point = 2

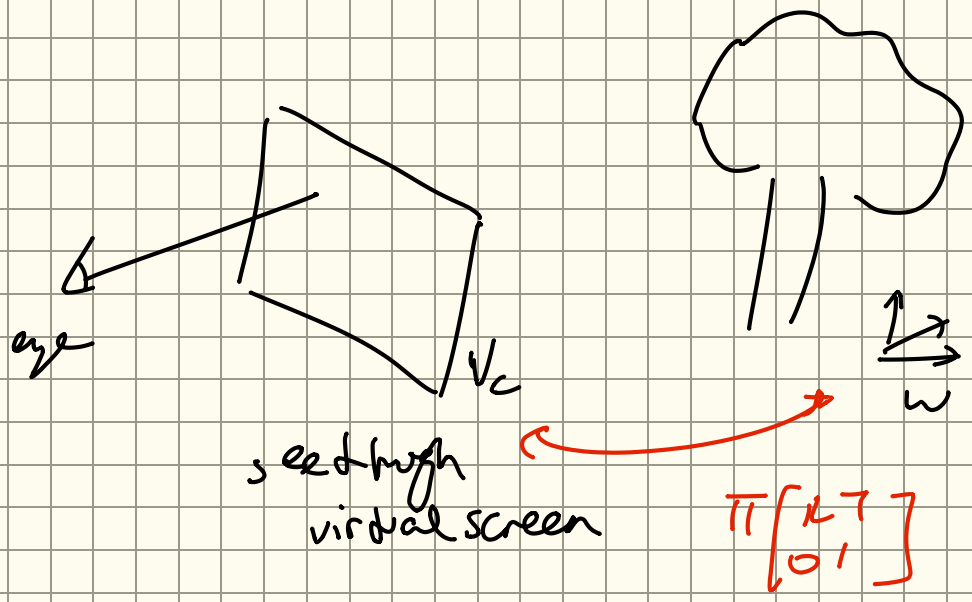
↓

2n

$11 + 11 + 3n \leq 4n$ $\quad n \geq 22$

Optical seethrough:

eye = camera



$$eye = [K] \dots$$

$$S\begin{pmatrix} u_e \\ v_e \\ 1 \end{pmatrix} = K \begin{pmatrix} R_{VC} & T_{VC} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_{WV} & T_{WV} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$$

unknown

How to calibrate?

Brute force

Video = see through

↳ smart phones, tablets → tracker → ArKit
→ ARCore
→ Vuforia
⋮

Optical see ..

↳ Hololens (MS)

↳ Magic Leap

↳ Vision Pro

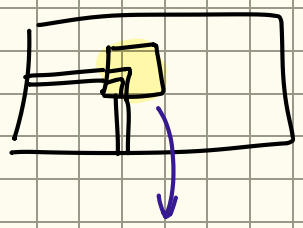
↳ Meta

⋮

α transparent screen

α calibration

α FOV is narrow



visible part.

Video see through device
but head mounted

not exactly optical see through but
you can get close to it.

α You need natural-looking interaction mechanism.

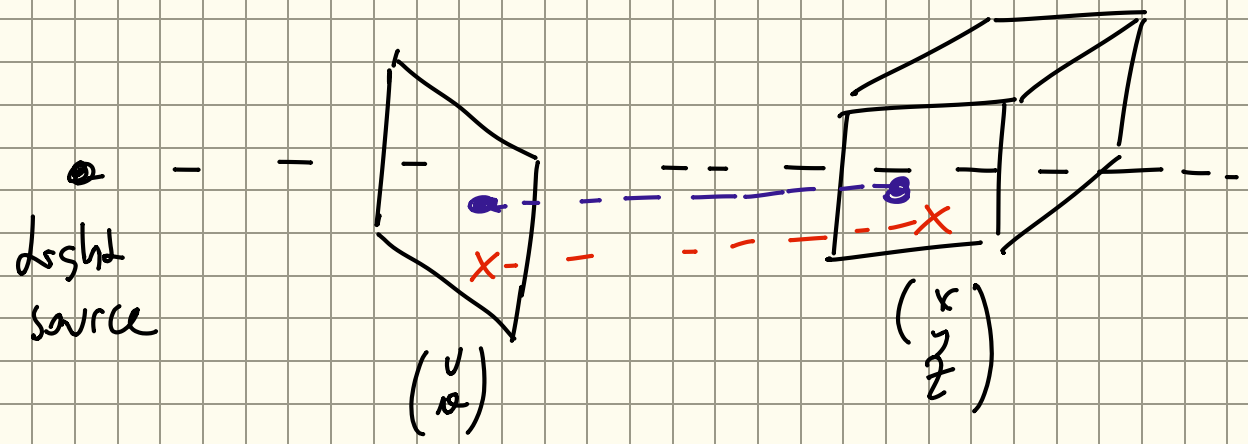
Head is relative

Gaze is not ..
?

"Motion sickness"

Projection

↳



∝ If you give me some known points we can find R, T.

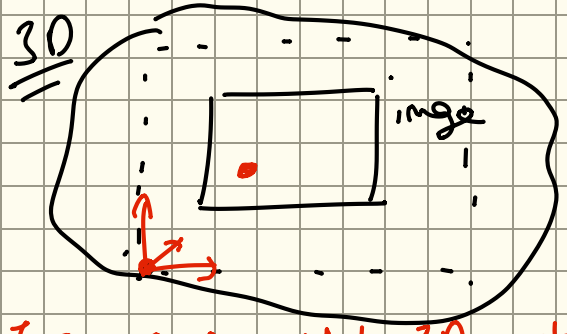
(x, y, z) (u, v)

∝ How do find those points?

↳ u, v → Generated image

↳ I can find them because I am generating them.

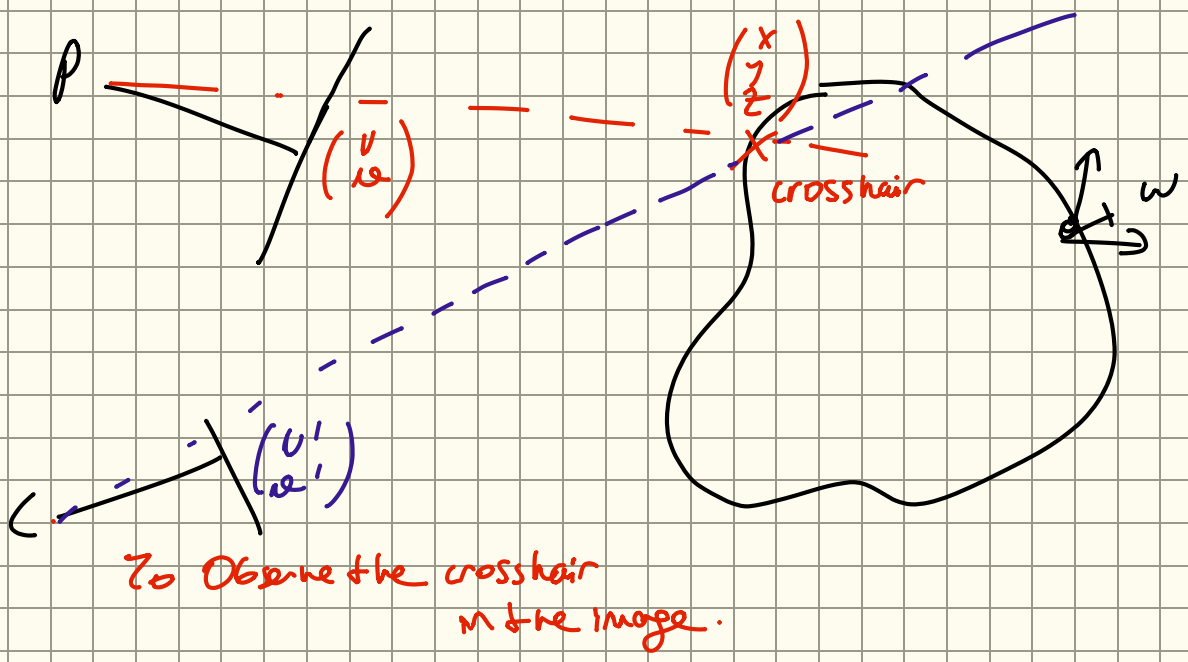
(Image)



I can make with 3D world coordinate

I can use image coordinate bcs I generated it.

Projector AR Calibration:



To Observe the crosshair in the image.

Camera

$$s\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = K_C \begin{pmatrix} R_{WC} & T_{WC} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2 eq.
6+5 unknown

known

Projector

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K_P \begin{pmatrix} R_{WP} & T_{WP} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2 eq.
6+5 unknown

known

C → 6+5

P → 6+5

Each point → 3n

eq per point = 2

↓

2n

$11 + 11 + 3n \leq 4n$

$n \geq 22$

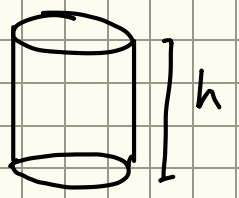
β out of picture } instead of $3n$ equation we said $2n$ equation. That's why there are no β in unknown part (if there would it would be $+2n$ for n point)

Computer Graphics pipeline:

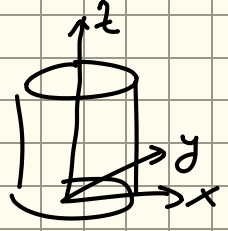
∝ How do we take real object and model as virtual one?

- Appearance, necessary part is visible part.
- Model is mathematical,

Mathematical model of cylinder: →



radius, height



x, y, z

$0 \leq z \leq h$

x, y supposed to be on a circle.

$x = r \cdot \cos\theta$

$y = r \cdot \sin\theta$

Any point on cylinder will be → $\begin{pmatrix} r\cdot\cos\theta \\ r\cdot\sin\theta \\ h \end{pmatrix}$

$0 \leq \theta \leq 2\pi$

$0 \leq h \leq$ height

$x^2 + y^2 \leq r$

Bottom $\begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$

Top $\begin{pmatrix} x \\ y \\ height \end{pmatrix}$

∝ These shapes are not uniform when viewed from different perspectives.

How to represent a surface using triangles:

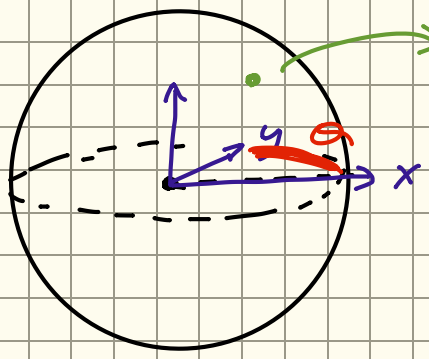
Set of triangles. → minimum surface patch (3 points)

↓

$n \times 3$ parameters

How can we do the triangulation?

(How would you triangulate the surface?)



$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ on the sphere

↳ constraints

$$x^2 + y^2 + z^2 = r^2$$

Better representation:

$\theta \rightarrow$ angle in xy plane

$\alpha \rightarrow$ vertical angle

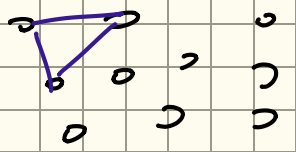
$$x = r \cdot \cos\theta \cdot \cos\alpha$$

$$y = r \cdot \sin\theta \cdot \cos\alpha$$

$$z = r \cdot \sin\alpha$$

} Now I have a random point.

If it was on 2D:

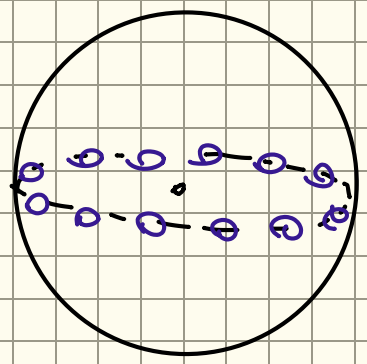


it is easy, find closest points

But it is now 3D.

There are algorithms for that

Instead of randomly, do better:

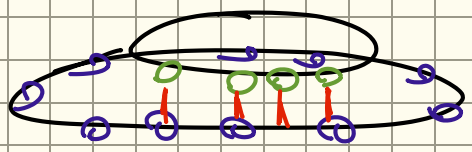


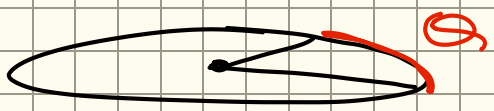
$$x = r \cdot \cos\theta \cdot \cos\alpha$$
$$y = r \cdot \sin\theta \cdot \cos\alpha$$
$$z = r \cdot \sin\alpha$$

$z = 0$ means points lie on the circle equator.

$\alpha = 0$:

$$x = r \cdot \cos\theta$$
$$y = r \cdot \sin\theta$$
$$z = 0$$





$$\begin{pmatrix} r\cdot\cos\theta \\ r\cdot\sin\theta \\ 0 \end{pmatrix} - \begin{pmatrix} r\cdot\cos\Delta\theta \\ r\sin\Delta\theta \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} r\cdot\cos\theta\cdot\sin\Delta\alpha \\ r\cdot\sin\theta\cdot\sin\Delta\alpha \\ r\cdot\cos\alpha \end{pmatrix} - \begin{pmatrix} r\cdot\cos\Delta\theta\cdot\sin\Delta\alpha \\ \vdots \end{pmatrix}$$

The equation shows how to calculate the **difference between adjacent points** on the surface to form a triangle.

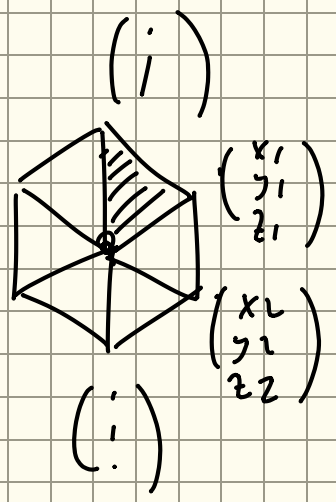
For two adjacent points, the difference is calculated as:

$$\begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \end{bmatrix} - \begin{bmatrix} r\cos(\theta+\Delta\theta) \\ r\sin(\theta+\Delta\theta) \\ 0 \end{bmatrix}$$

Similarly, for points in 3D on the sphere:

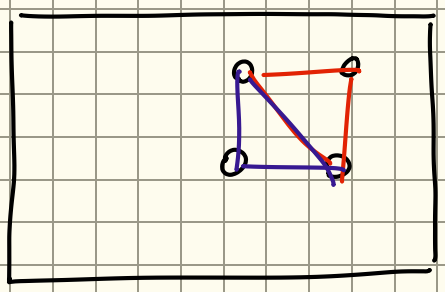
$$\begin{bmatrix} r\cos\theta\sin\Delta\phi \\ r\sin\theta\sin\Delta\phi \\ r\cos\Delta\phi \end{bmatrix} - \begin{bmatrix} r\cos(\theta+\Delta\theta)\sin\Delta\phi \\ r\sin(\theta+\Delta\theta)\sin\Delta\phi \\ r\cos\Delta\phi \end{bmatrix}$$

→ Simple algo. to build triangles.



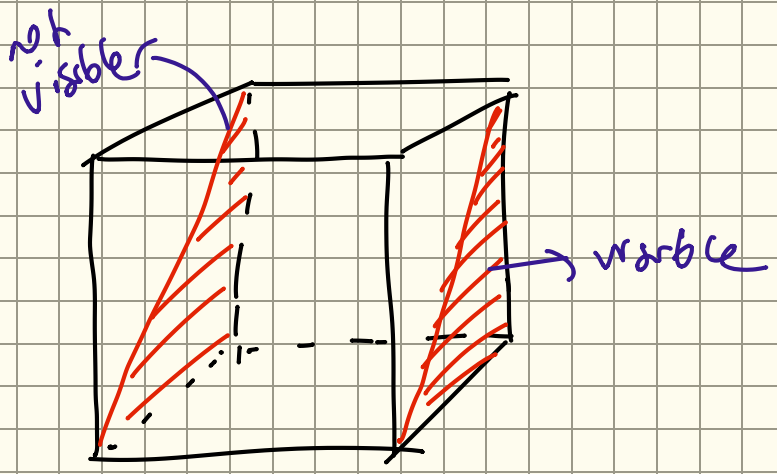
- Some of points will be reused.
- Once calibration is done:

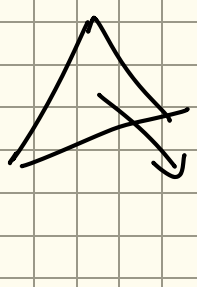
$$s\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = K \cdot \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$



- I have a basic idea I know where these triangles are on my image.

- How do I know given triangle is visible or not?





- I need to orient my surfaces.

finding the Surface Normal

Triangulation

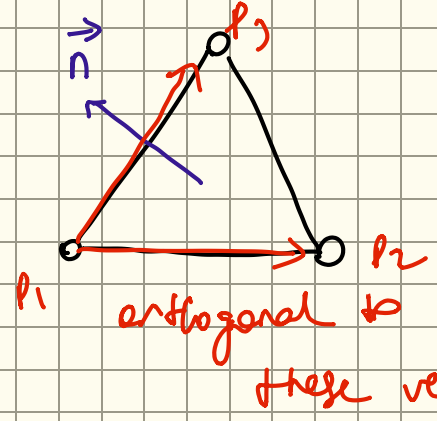
$v_1 \dots v_n$ + per each triangle

$t_1 \dots t_m$ order / direction

↓

surface normal } is a vector that perpendicular to surface

How can I find the normal?



$$\vec{n} = |\vec{P_1 \cdot P_2}| \times |\vec{P_1 \cdot P_3}|$$

normalize

$$\vec{P_1 \cdot P_2} = P_2 - P_1$$

$$\vec{P_1 \cdot P_3} = P_3 - P_1$$

Which order they actually in? Clockwise, Counterclockwise ?

T_1 : P_1, P_2, P_3 → If I tell you this is the correct order, you can use this to find normal.

+

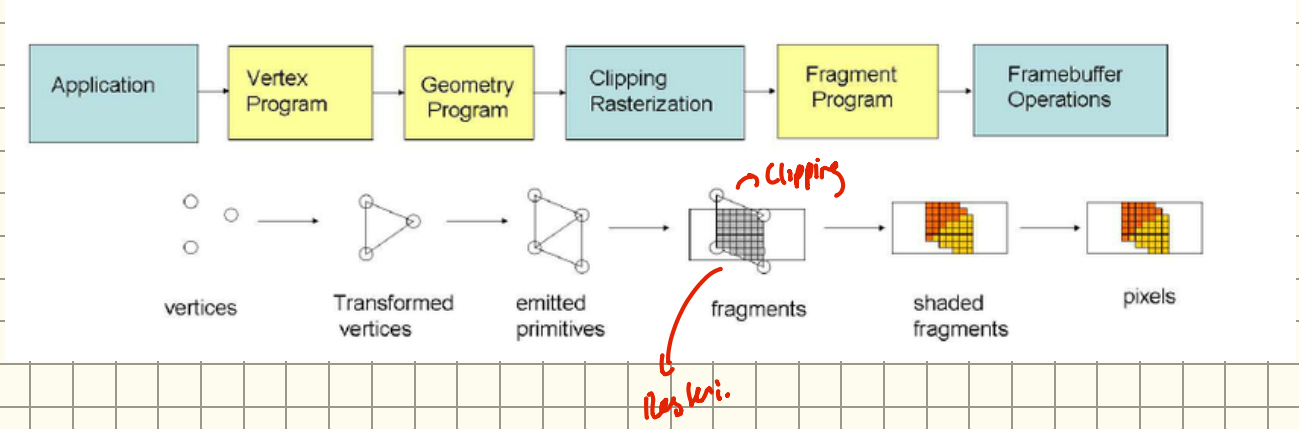
$\vec{n}$

Repetitive process:

- I have millions of triangles
  1-) Take each vertex
  2-) for each triangle
  - Calculate normal vector
  - Determine if triangle is visible (based on normal)
  - Render only visible triangles.

& There has to be some sort of a check to blocked triangles.

The Graphics Pipeline:



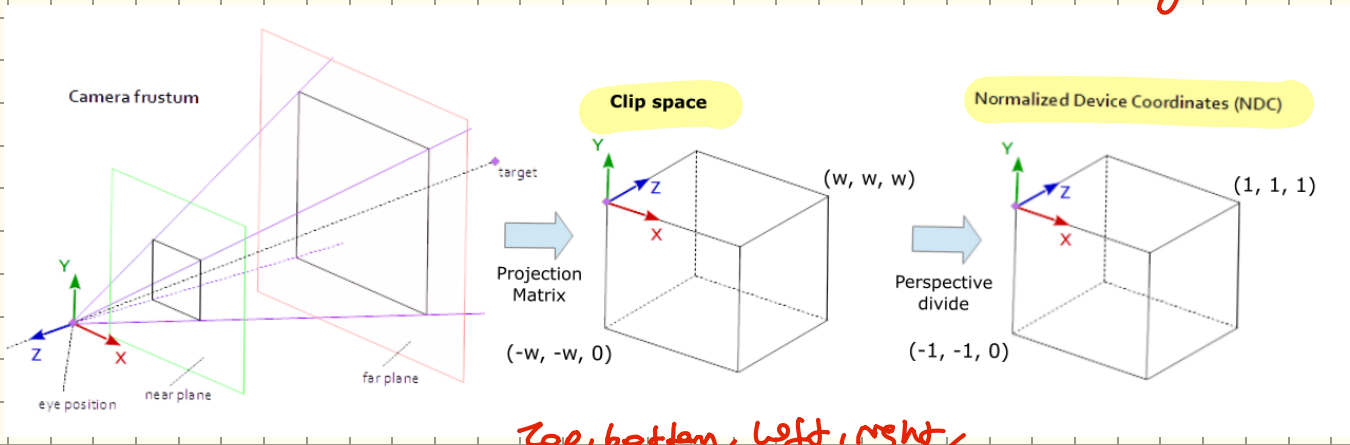
View transform -> camera, moves the world relative to camera

Projection " -> intrinsic parameters, convert 3D scene to 2D screen

Clip Space:

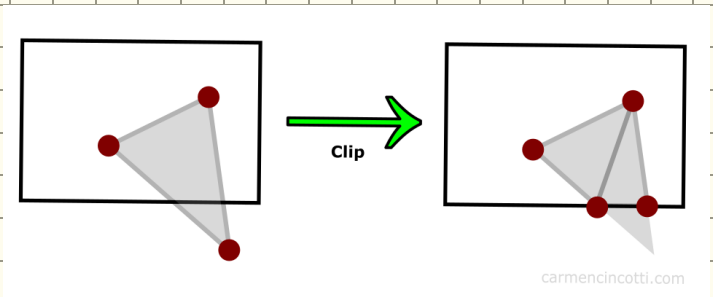
Normalized:

$-1 \leq x, y, z \leq 1$



top, bottom, left, right, near, far boundaries

- After applying projection matrix, all vertices transformed into clip space.
- Anything outside the cube is discarded
- Inside triangles send to rasterizer.



- Space is kinda orthographic
- Everything is normalized.

Clip Space

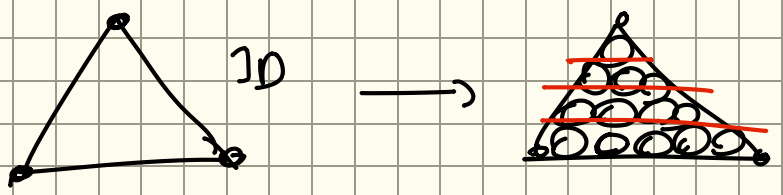
∠ After processing vertices and apply transformations to 3D object, all 3D objects are transformed into clip space [-1, 1]. (normalized space)

Triangles that extend beyond clip space boundary are clipped.

↳ After clipping, we know which triangles are visible.

Rasterization

∠ Filling triangles with pixels after its clipped and transformed into 2D screen space.
(fragment)



∠ Fill triangle from my perspective.

∠ Rasterizer have those pixel available to me.

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

↳ Include z (depth) } Only the closest triangle is visible.

0,1 unit 1 pixel

∠ During rasterization, z coor. → which pixels are visible

" " are hidden behind other objects.

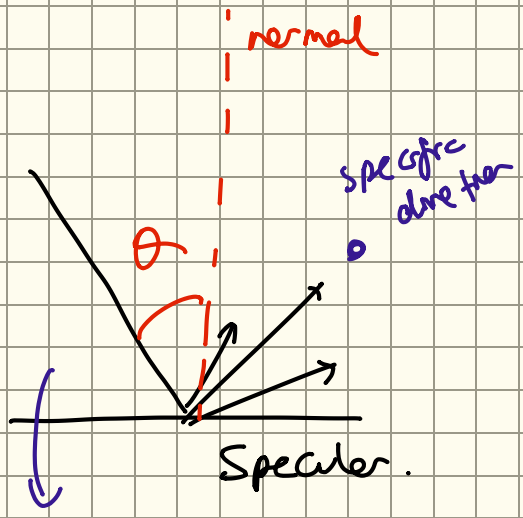
"z-buffering"

Light

→ Light travels along rays.

→ Light is generated at light sources.

Specular surface



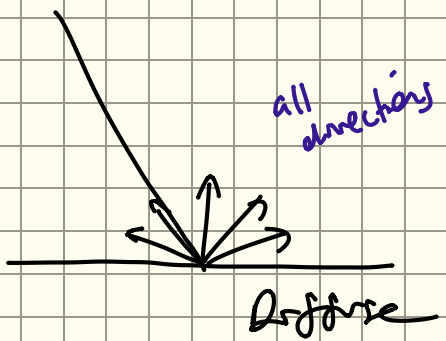
It is here

I can't see that point

bcs there is no light.

like mirror

Diffuse surface



like matte surfaces

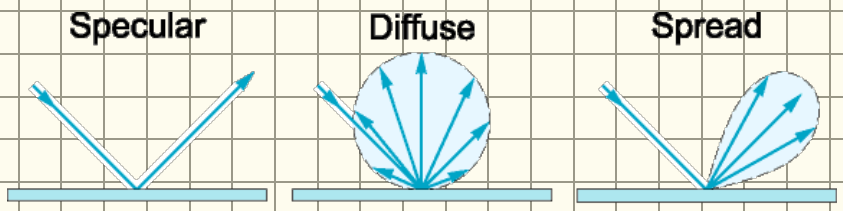


Fig. 3.2 Specular, diffuse, and spread reflection from a surface.

↳ Our world combination of these.

↳ Where my triangle is according to the light.

Is it diffuse, is it specular?

↓

This is complex, we are gonna do a lot of simplification

Shading

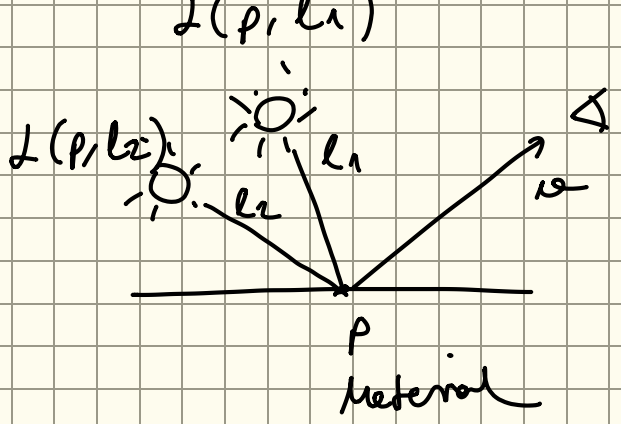
↳ Local Illumination models: simplifies lighting process.

- Ambient Lights:

  We assume that every surface get same amount of light, regardless of light sources positions.

- Point light sources:

  We assume that light comes from specific point (like sun).



$$I = f_{phong}(L^{light}, P, L, v \dots)$$

↓

Intensity of light at point P.

↳ This is still complicated

Simplify it with using different shading models.

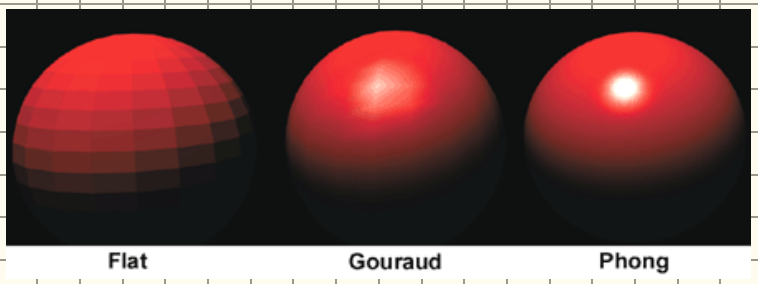
- Flat shading → Simplest among them
  Each triangle has 1 uniform color.

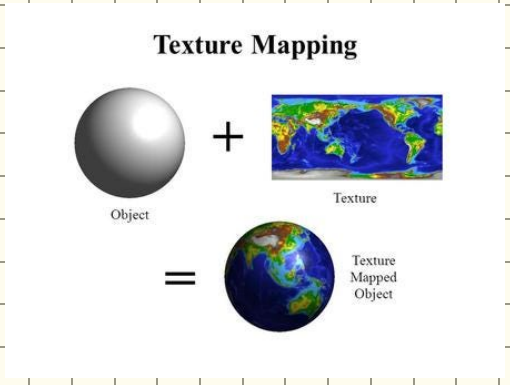
- Gouraud Shading → interpolated from each vertex color of triangle.

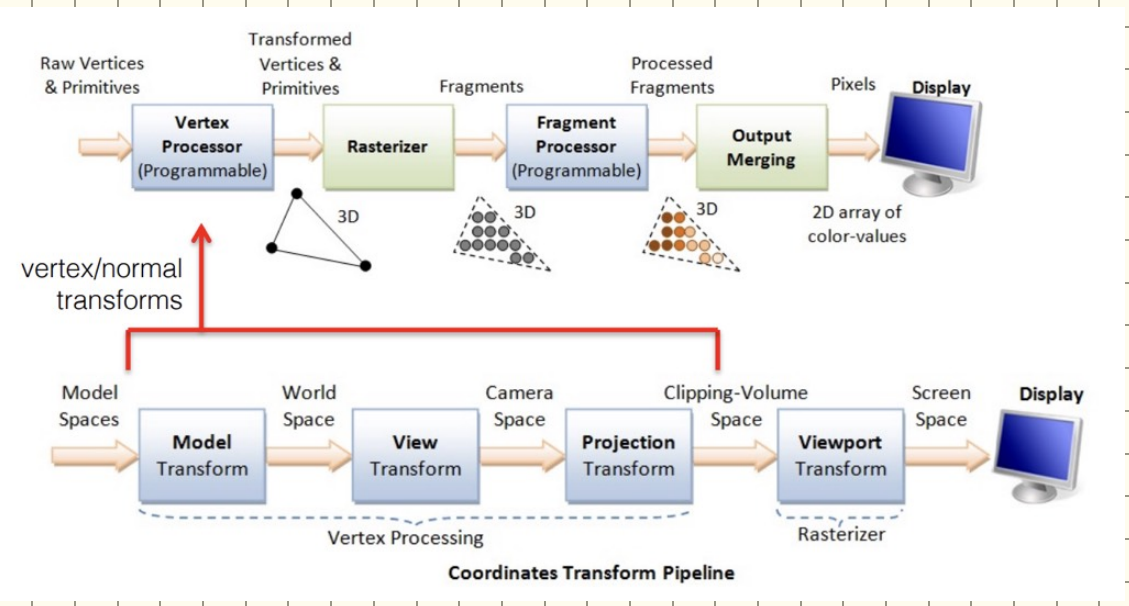
- Phong shading → Most accurate shading model.
  lighting is calculated per each pixel (fragment)

P (Material) → Texture mapping

- Repetitive texture







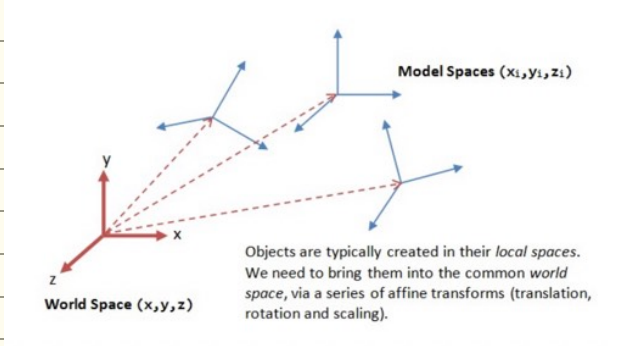
Coordinates Transform Pipeline

Stage	Space/Transformation	Description
Model Transform	Object → World Space	Positions the object in the 3D world
View Transform	World → Camera Space	Moves the camera and sets the view direction
Projection Transform	Camera → Clipping Space	Converts 3D scene to a 2D perspective view
Viewport Transform	Clipping → Screen Space	Maps the 2D image to the display screen coordinates (pixels)

"appropriate viewpoint"

"simulates camera lens"

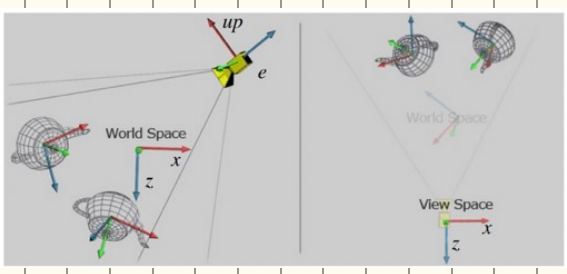
Model transform:



Projection transform:

Intrinsic parameters

View transform:



RENDERING

∠ Ray Casting : Which objects are visible from the camera's viewpoint ?

∠ Ray travels in 3D scene and checks if its intersects any triangle.

→ Triangles represent the scene.

→ Cast a ray from camera through each pixel

→ A 1000×1000 pixel screen } 1 million rays to be cast.

→ If a ray hits any triangle, that pixel gets color and lighting at the intersection point.

↳ Can we reduce 1 M ?

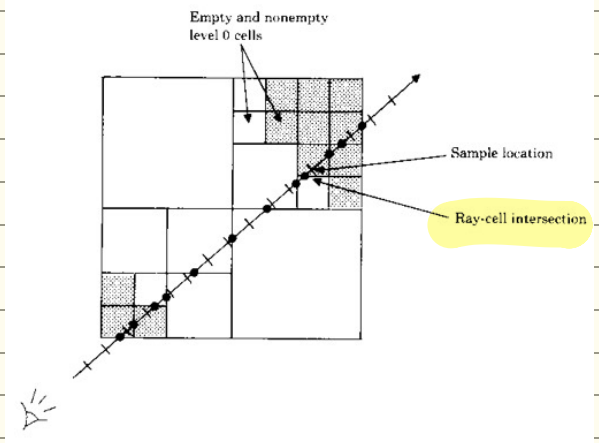
∠ Instead of going each triangle, check the hitting area.

Hierarchical representation;

1-) Split space on quadrants

→ Only look for the rays in relevant region.

KD tree to divide space.



2-) Even if I find the triangle visible for me, I need to repeat this for every ray.

This process is quite expensive, I can't do it in real time, like if I move...

"It has to be done in real time"

↳ Rendering can be done once you have camera, lighting...

↳ Combination of rendering and ray casting?

↳ Modern rendering engines ✓

- **Ray Casting** is used to calculate **accurate shadows, reflections, and global illumination.**
- The **rest of the scene** is rendered using **traditional methods** (like rasterization).

↳ Real-time rendering?

Here's what happens in a **real-time rendering pipeline**:

1. **Vertex Processing:**
Transform **3D vertices** into **2D screen coordinates.**
(We talked about this earlier with **View Transform** and **Projection Transform.**)
2. **Clipping & Rasterization:**
Convert **3D triangles** into **fragments (potential pixels).**
3. **Shading & Lighting (Fragment Processing):**
Apply **textures, colors, lighting, and shadows** to each pixel.
4. **Depth Testing (Z-buffer):**
Ensure **only visible pixels** are drawn.
5. **Final Image Output:**
Send the final image to the screen.

Rendering → Vertex processing → Clipping → Rasterization → Fragment Shader → Depth testing → Output

Ray casting → Ray Generation → Ray Triangle Intersection Test → Shading, lighting → Color output each pixel

Steps in Raycasting:

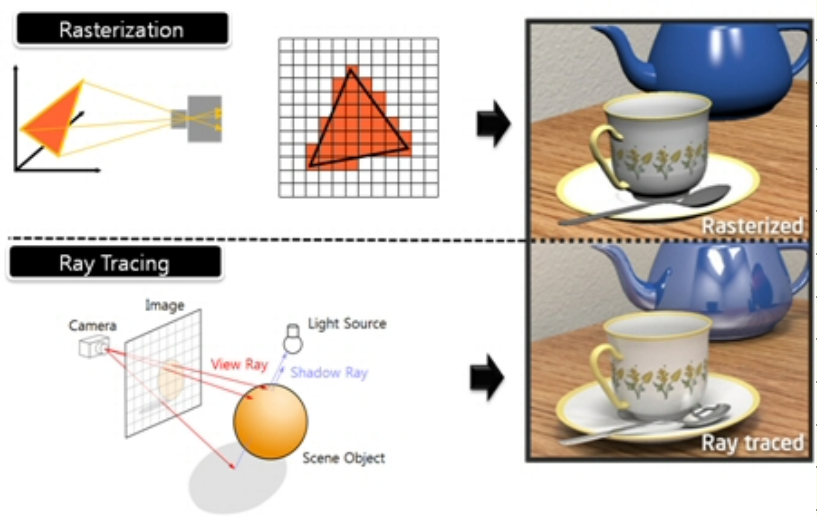
1. For each pixel on the screen:
    - Cast a ray from the camera through the pixel.
    - Check for intersections with objects in the scene.
    - Determine the closest object hit by the ray.
    - Calculate the pixel color based on the material properties, lighting, etc.

It simulates how light rays behave in real world.

Steps in Rasterization:

1. **Vertex Processing:** Transform 3D vertices to screen space.
2. **Clipping:** Discard any parts of objects outside the camera's view.
3. **Rasterization:** Convert triangles into fragments (pixels).
4. **Fragment Processing:** Calculate the color of each fragment.
5. **Output Merging:** Combine fragments into a final image.

lighting effects is hard



10 Usability Heuristics

- Score must be visible } User should be informed whats happening (notifications...)
- System should be alive
- Match between system and real world. → Tesla car
  ↳ User language, familiar concepts
- User control and freedom. } free to undo, redo
- Consistency and standards → Mouse moving left/right
  Screen button
- Error prevention
  ↳ Should prevent error before it occurs.

- Recognition rather then recall

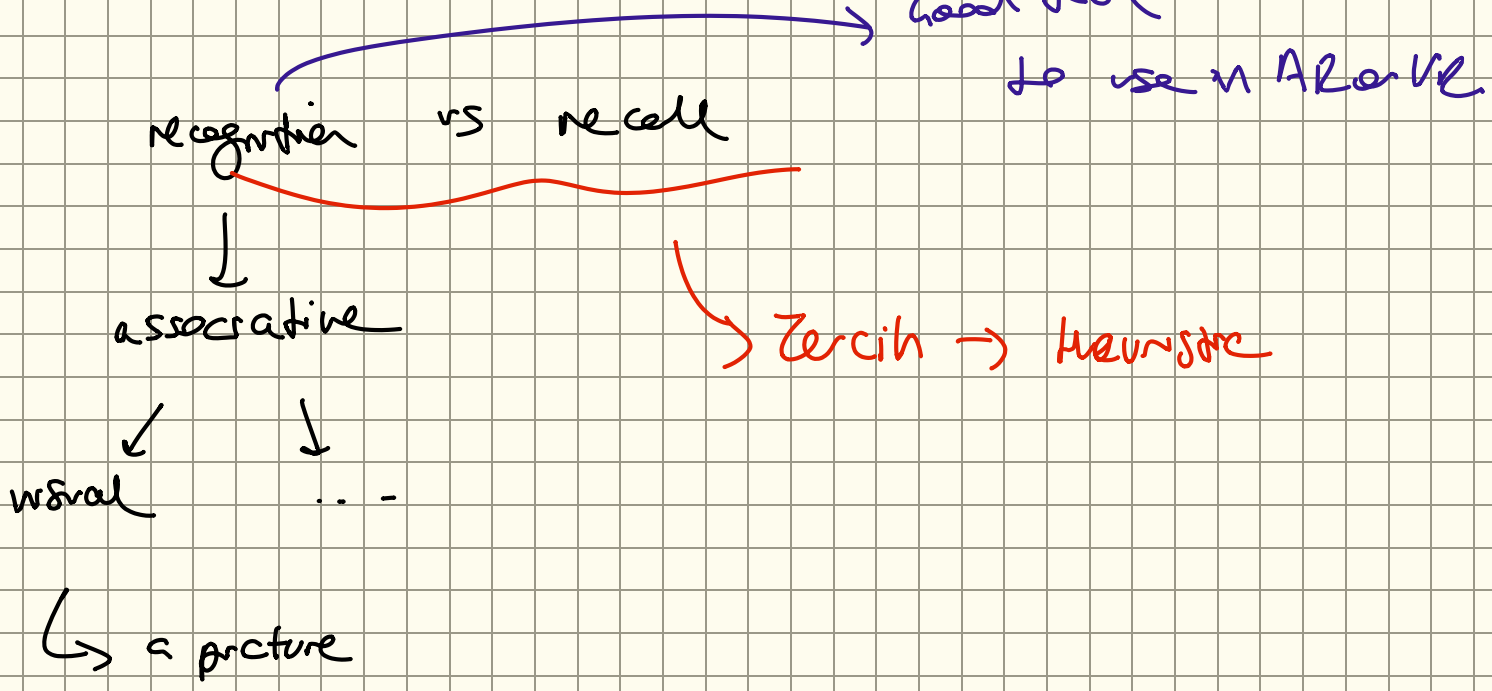
Try both remembering!

10 step task → now at 7 step → Recall: keep it in memory

↳ Recognition: there are some info. in env.
to recognize the step.

↳ AR gives you more things to recognize more than recall.

"associative memory" } System gives reminders (picture, showing buttons)
Easier for user as they dont need to recall details.



↳ a picture

↳ 2D design difficult!

↳ 3D " is more difficult!

↳ User should deal with spatial dimensions and new interactions.

- Aesthetic vs minimalism

- Recover from errors } "Tracking is bad, you need to recover"

↳ No notification → Restart → Bad
↳ Scan again

- Help and documentation

Nielsen Norman Group

Jakob's Ten Usability Heuristics

1 Visibility *of* System Status

Designs should *keep users informed* about what is going on, through appropriate, timely feedback.

Interactive mall maps have to show people where they currently are, to help them understand where to go next.

2 Match between System and the Real World

The design should speak the users' language. Use words, phrases, and concepts *familiar to the user,* rather than internal jargon.

Users can quickly understand which stovetop control maps to each heating element.

3 User Control *and* Freedom

Users often perform actions by mistake. They *need a clearly marked "emergency exit"* to leave the unwanted action.

Just like physical spaces, digital spaces need quick "emergency" exits too.

4 Consistency *and* Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. *Follow platform conventions.*

Check-in counters are usually located at the front of hotels, which meets expectations.

5 Error Prevention

Good error messages are important, but the best designs carefully *prevent problems* from occurring in the first place.

Guard rails on curvy mountain roads prevent drivers from falling off cliffs.

6 Recognition Rather Than Recall

***Minimize the user's memory load* by making elements, actions, and options visible. Avoid making users remember information.**

People are likely to correctly answer "Is Lisbon the capital of Portugal?".

7 Flexibility *and* Efficiency of Use

Shortcuts — hidden from novice users — may *speed up the interaction* for the expert user.

Regular routes are listed on maps, but locals with more knowledge of the area can take shortcuts.

8 Aesthetic *and* Minimalist Design

Interfaces should not contain information which is irrelevant. Every extra unit of information in an interface *competes* with the relevant units of information.

A minimalist three-legged stool is still a place to sit.

9 Recognize, Diagnose, *and* Recover from Errors

Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

Wrong-way signs on the road remind drivers that they are heading in the wrong direction.

10 Help *and* Documentation

It's best if the design *doesn't need* any additional explanation. However, it may be necessary to provide documentation to help users complete their tasks.

Information kiosks at airports are easily recognizable and solve customers' problems in context and immediately.

NN/g

www.nngroup.com/articles/ten-usability-heuristics/

Feedback - Senses

∝ I am acting

↳ with generic sensors } vision, touch, hearing ...

↳ with virtual sensors } VR glasses, vibration ...

∝ Inconsistency between my 2 sensors

while in roller coaster

↓ generic → standing still

↓ virtual → in motion

} "Motion sickness"

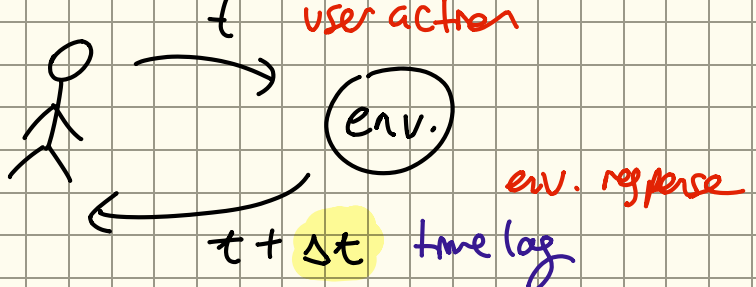
- Spatial domain

- In VR system its difficult to match virtual and balance match.

↳ Spatial mismatches:

Even video seethrough with 2 cameras have some mismatching (eye side) // delays.

- Temporal Delays:



- Hitting the ball in AR → Generating sound, noise, vibrate...

↳ "Virtual synesthesia"

Spatial Continuum: Seamless interaction between real and virtual world.

"Virtual env. should feel like natural extension of real world without any delays"

Ex: A virtual car should stay still when I move should cast realistic shadow and reflection

Explain how to go from task domain to spatial axes?

- **Task domain**: "Move this object from Point A to Point B."
- **Spatial axes**: This task needs to be broken down into **movements along the X, Y, and Z axes** in 3D space.

Step	Description
1. Define Task	Identify the task to be performed (e.g., move an object).
2. Break into Movements	Identify movements required (e.g., move left, right, up).
3. Map to Spatial Axes	Translate movements to the **X, Y, Z axes**.
4. Use Coordinates	Use a **coordinate system** to place or move objects.

How would you define space in 3D UI?

x, y, z

Aspect	Explanation
Coordinate System	Space is defined by **X, Y, Z** axes.
Egocentric Space	User-centered space, objects move relative to the user.
Allocentric Space	World-centered space, objects have fixed positions.
Position, Orientation, Scale	Core attributes of any object in 3D space.
Interaction Space	Area where users can interact with virtual objects.

AR vs VR vs MR vs DR

AR → Adds virtual elements to real world, doesn't replace it.

VR → Replaces the real world with fully virtuality

MR → Comb. of AR + VR where virtual obj. interact with real world.

DR → Removes object from reality, rather than adding.

Scenario	AR	VR	MR	DR
Gaming	Pokemon Go	Beat Saber	HoloLens interactive games	Removing enemies in real-world games
Interior Design	Visualize furniture in a room	Full virtual room walkthrough	Place furniture + interact	Remove unwanted objects from a room