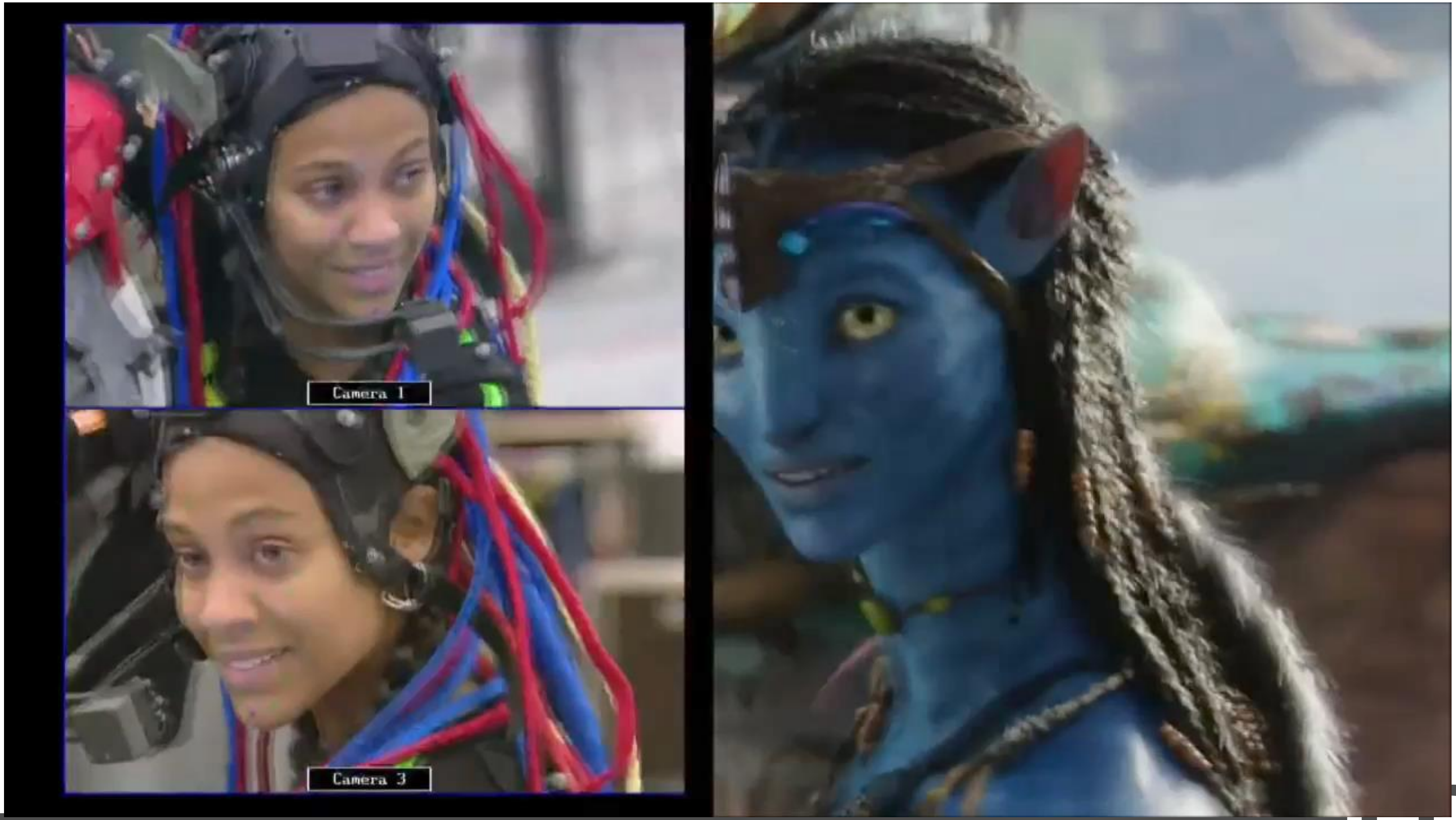
3D Scanning & Motion Capture

3D Concepts and Sensors

Prof. Matthias Nießner

Last Lecture: Motion Capture



Last Lecture: Capturing Geometry

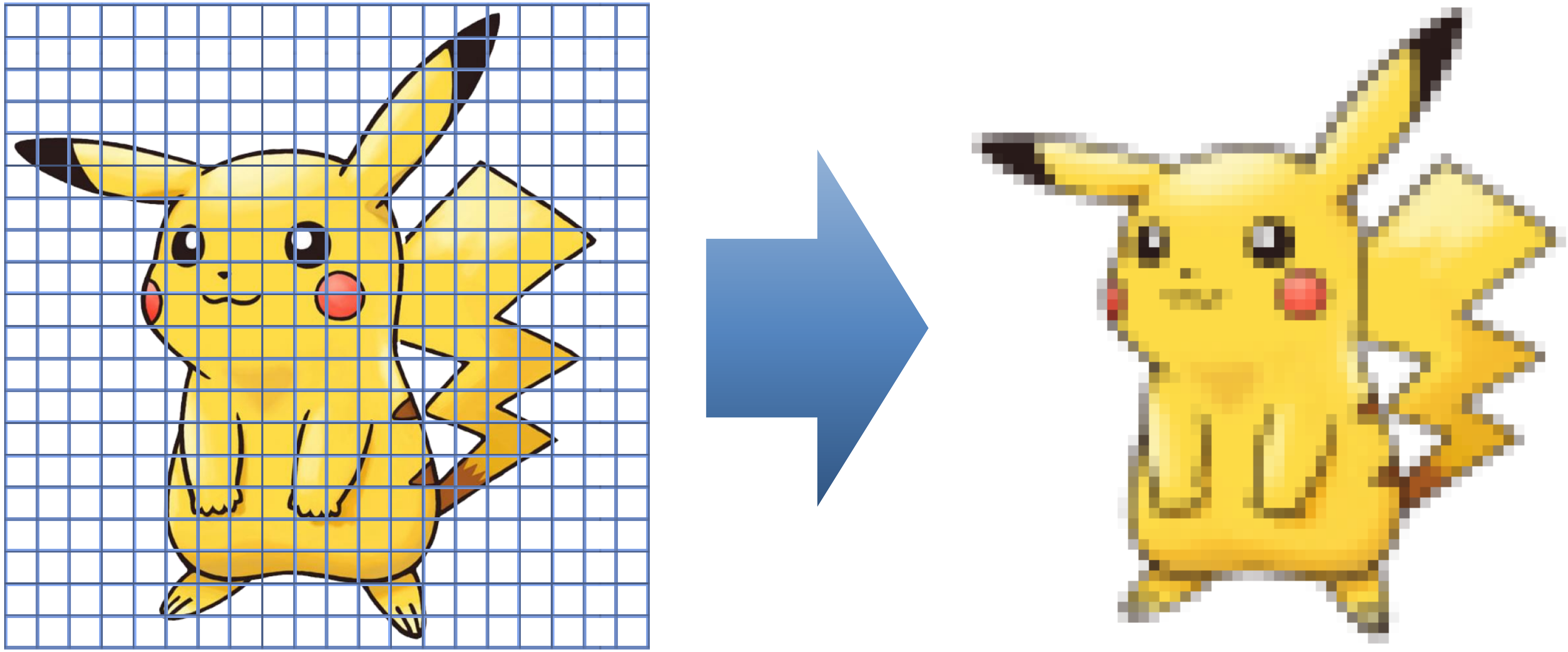
Source: matterport.com

Last Lecture: Geometry Capture

Today: What is actually “3D”?

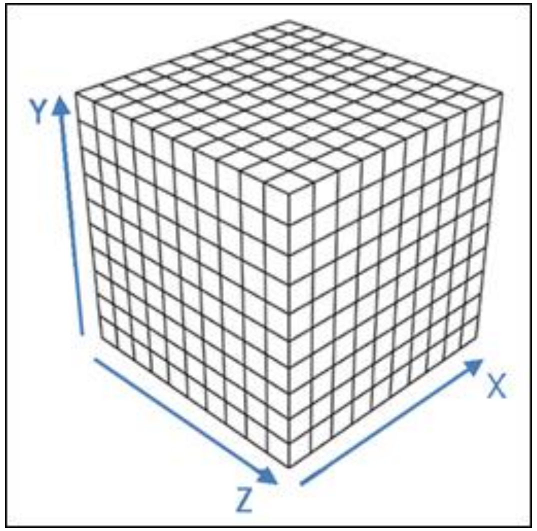
What is 2D?

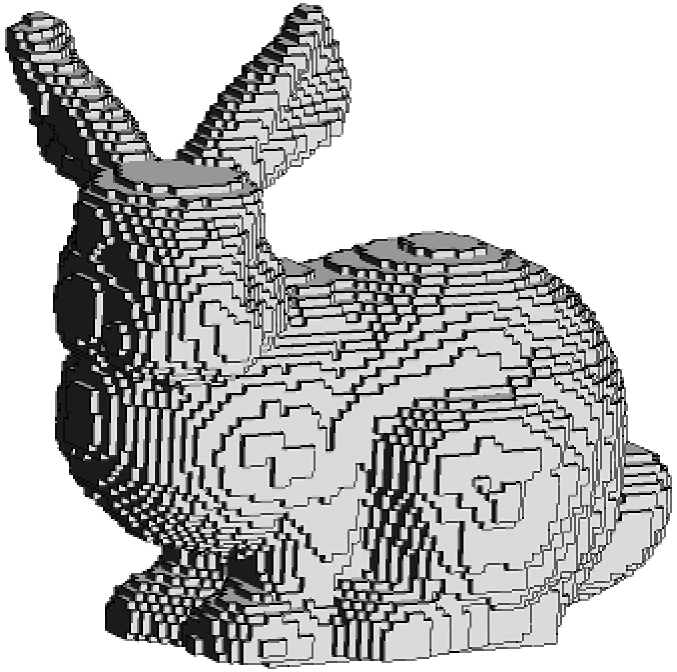
- Image: 2D array of pixels



What is 3D?

- Voxels (volumetric representations)
  - Occupancy grid in 3D: often binary; or probabilistic
  - 3D analog of pixel -> 3D array

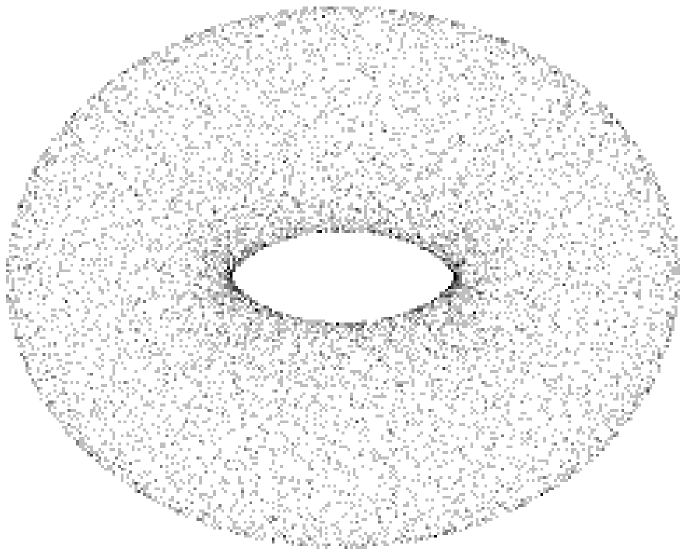




Colored voxels == Minecraft ☺

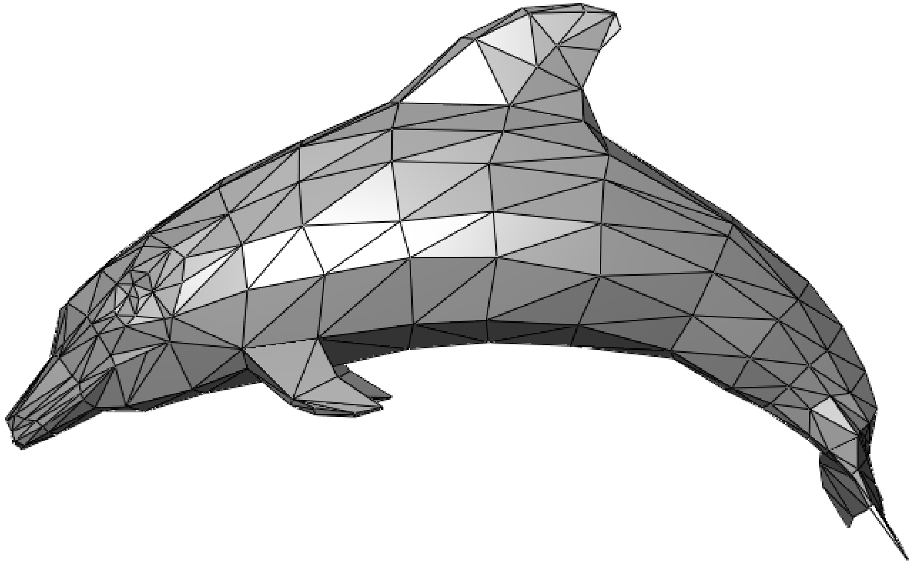
What is 3D?

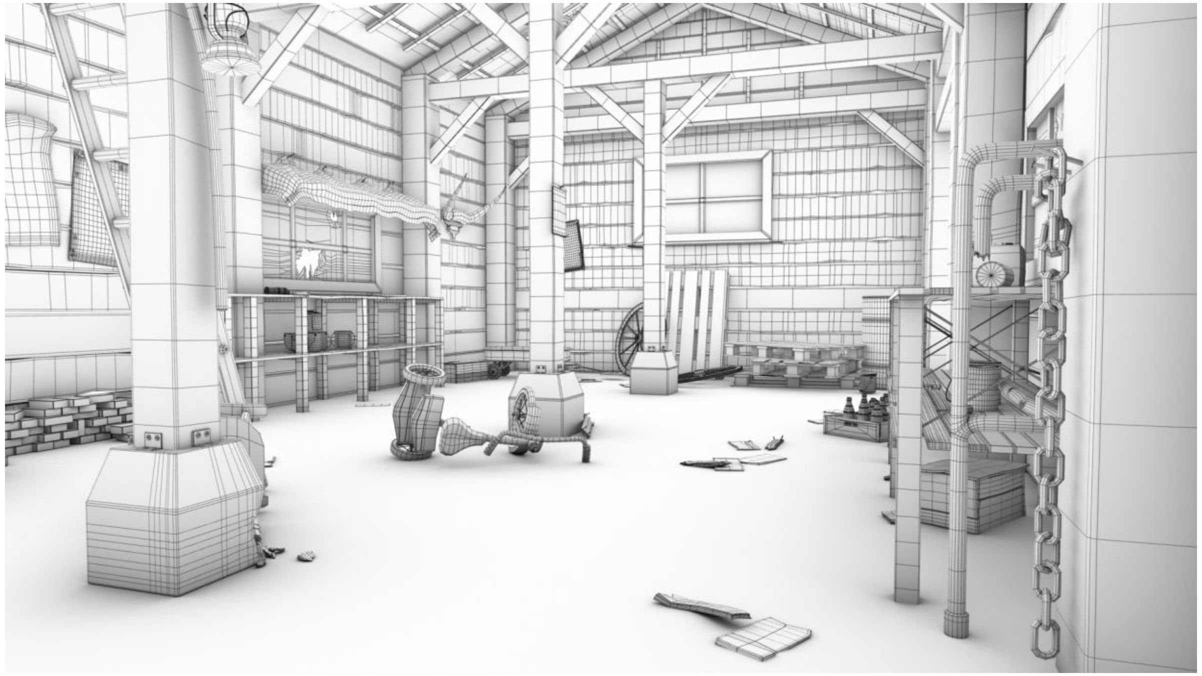
- Point Clouds
  - Set of 3D points with (x, y, z)-coordinates
  - Associated attributes
    - Color
    - Normal
  - Data are typically “raw” measurements
    - I.e., no inter- or extrapolation
    - No surface defined



What is 3D?

- (Polygonal) Meshes
  - Vertices + Faces (often triangles)
  - Piecewise linear
    - Attributes: colors, normals, textures





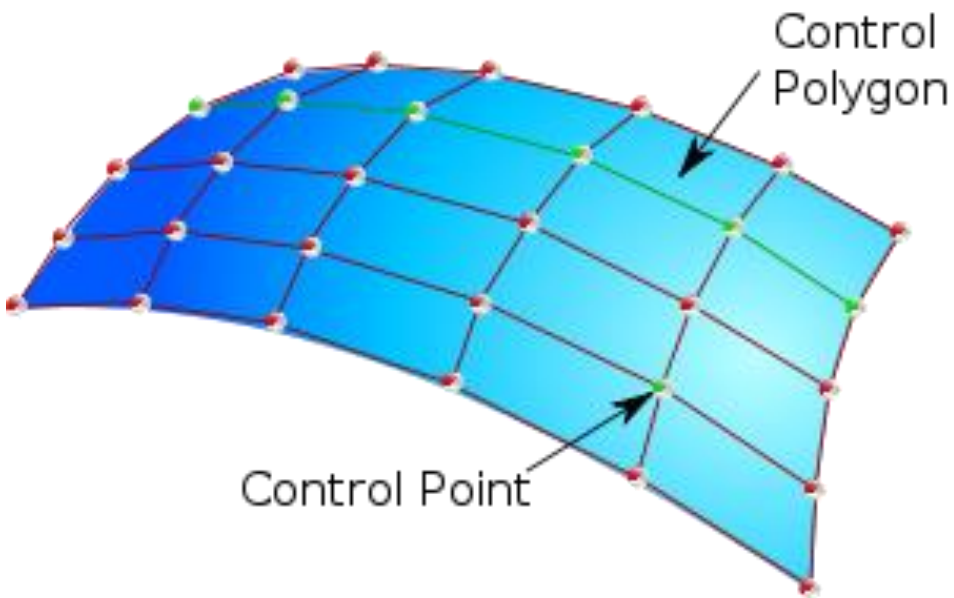
What is 3D?

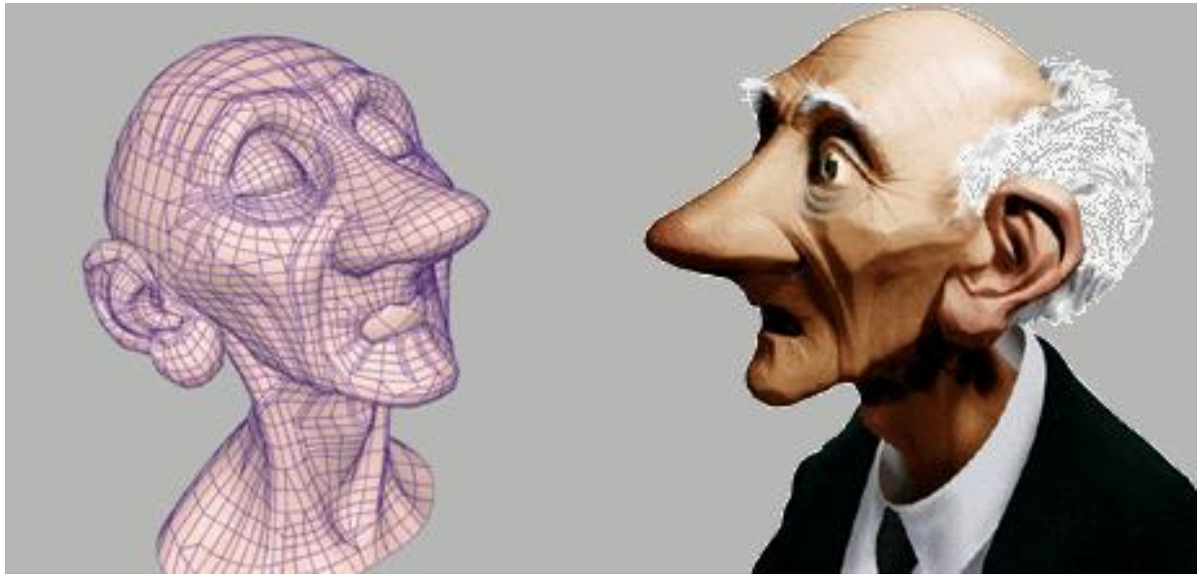
- (Polygonal) Meshes

… and with a little texturing and lighting

What is 3D?

- Parametric surfaces
  - Splines, NURBS, subdivs.
  - Set of points that define control grid

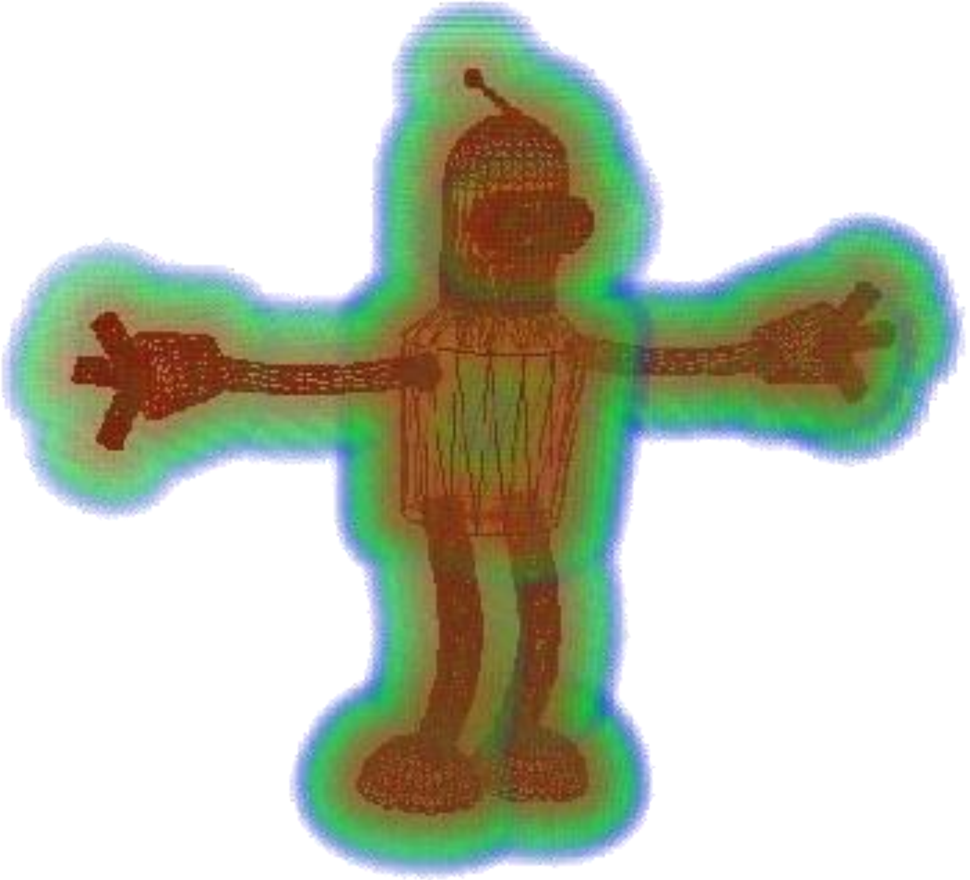


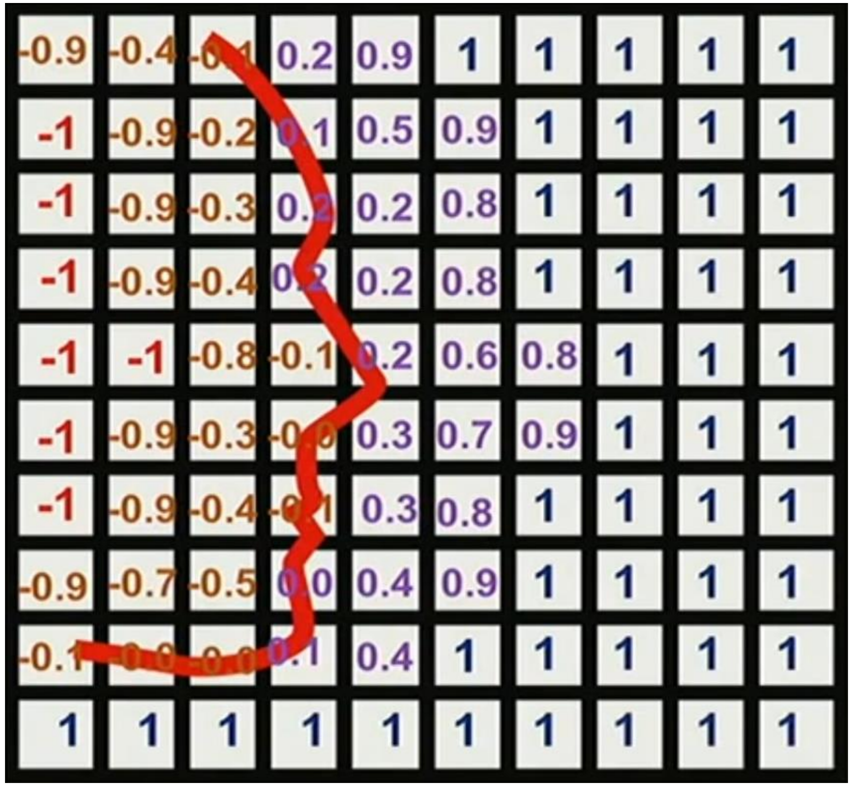


Geri's game

What is 3D?

- Implicit Surfaces: e.g., signed distance field (SDF)





Truncated Signed Distance Field (TSDF)

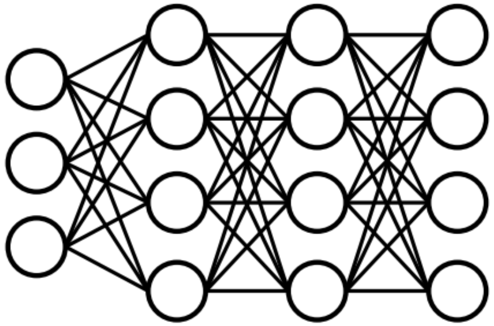
What is 3D? – Side Note

- Mathematically:
  - Explicit function: $f(x) = y$
  - Implicit function: $x^2 + y^2 - 1 = 0$

- MLP-based neural Networks are *not* implicit:

(x, y, z)

Query coordinate



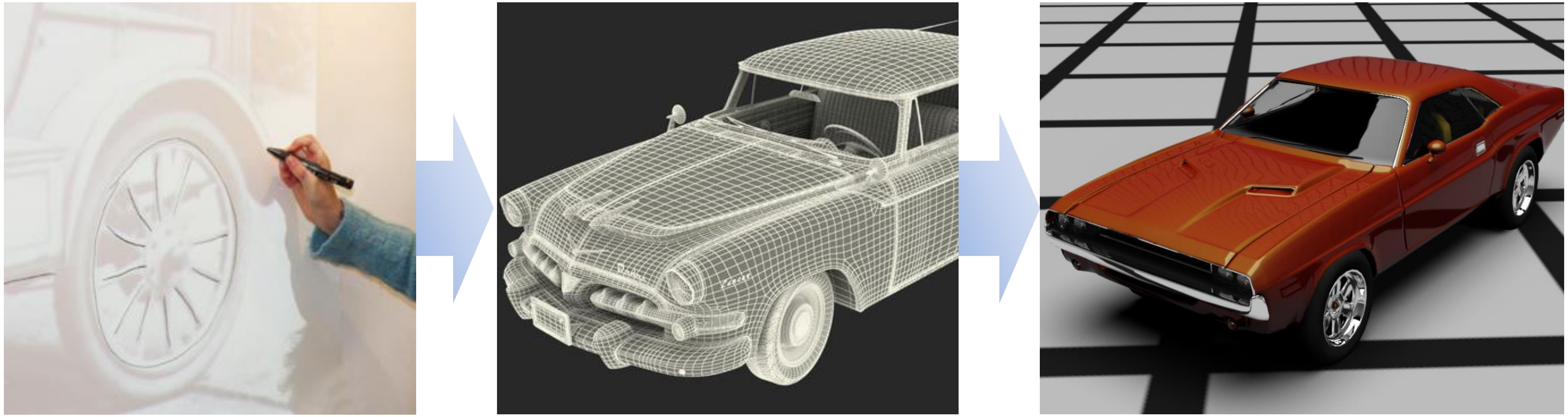
Multi-layer Perceptron (MLP)

SDF value

Literature a bit inconsistent:
e.g., Neural Fields, Neural Representation, Coordinate-based Networks, etc.

How to obtain “3D”?

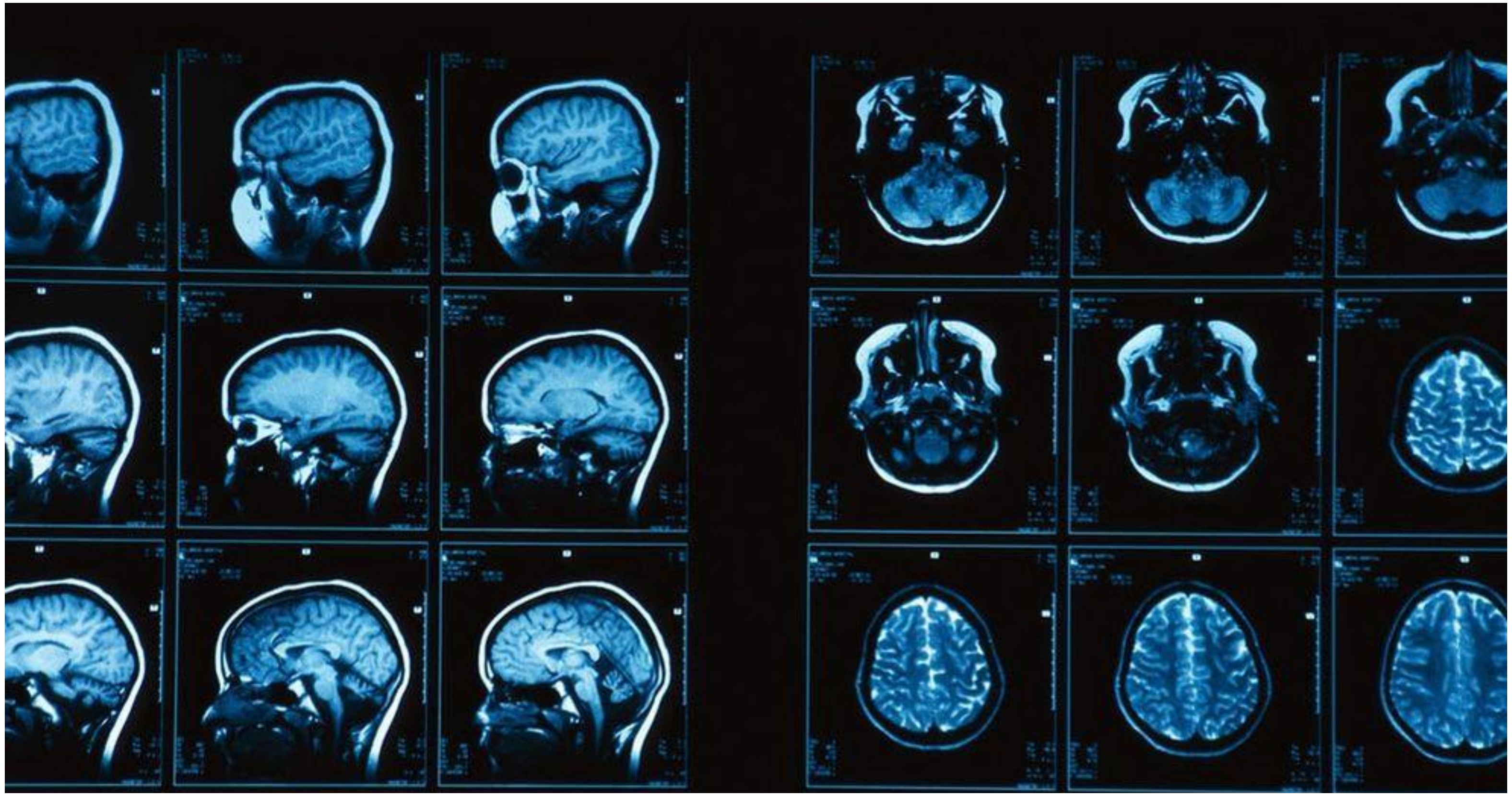
- Could be modeled by artist (video games and movies)
  - E.g., Mudbox, Maya, 3DMax, Zbrush, Houdini, SolidEdge, NX, Catia, …



Content Creation　　　Visualization / Rendering

How to obtain “3D”?

- With 3D Scanning ☺
  - MRI / CT



How to obtain "3D"?

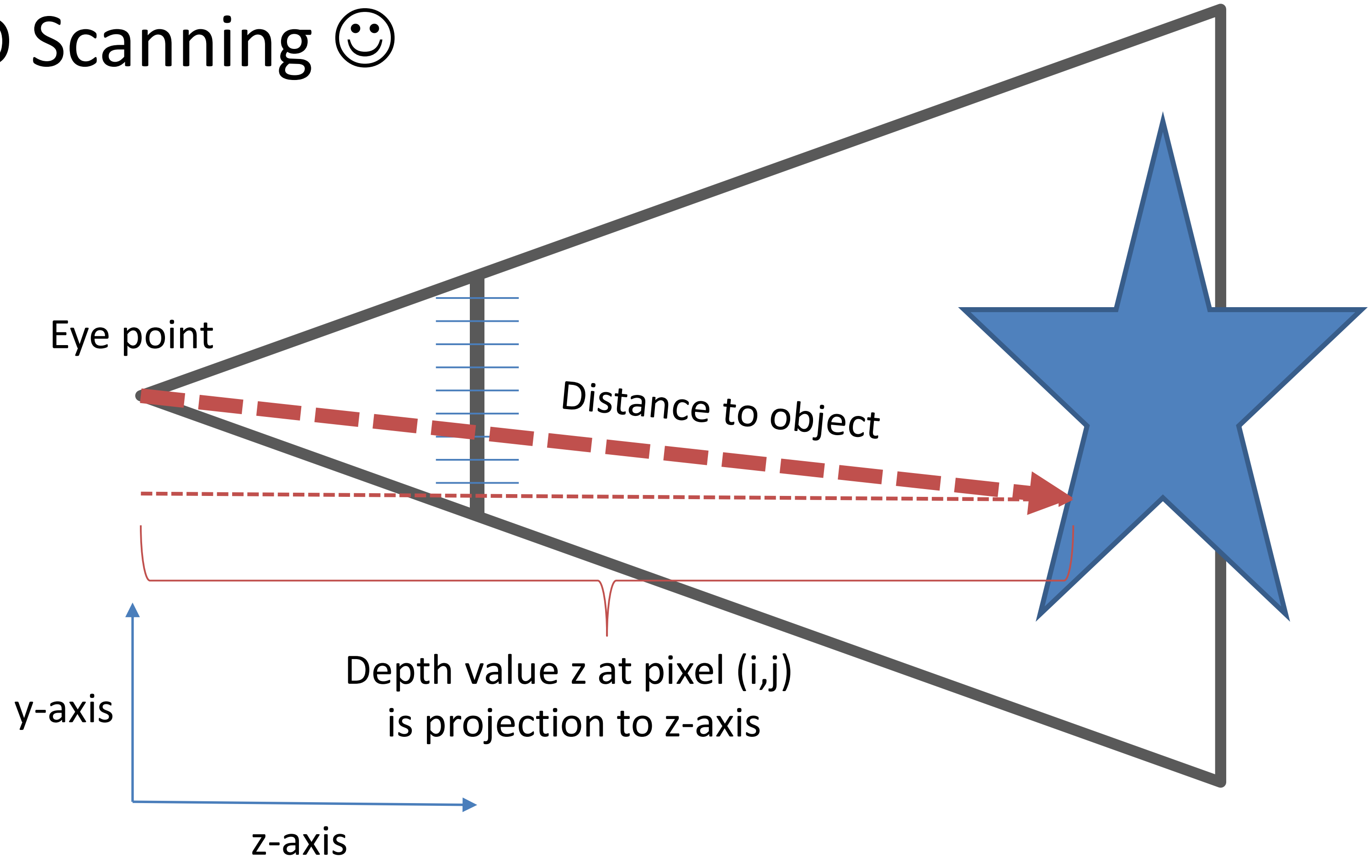
- With 3D Scanning ☺

Depth Image (D)

Color Image (RGB)

How to obtain “3D”?

- With 3D Scanning ☺



How to obtain “3D”?

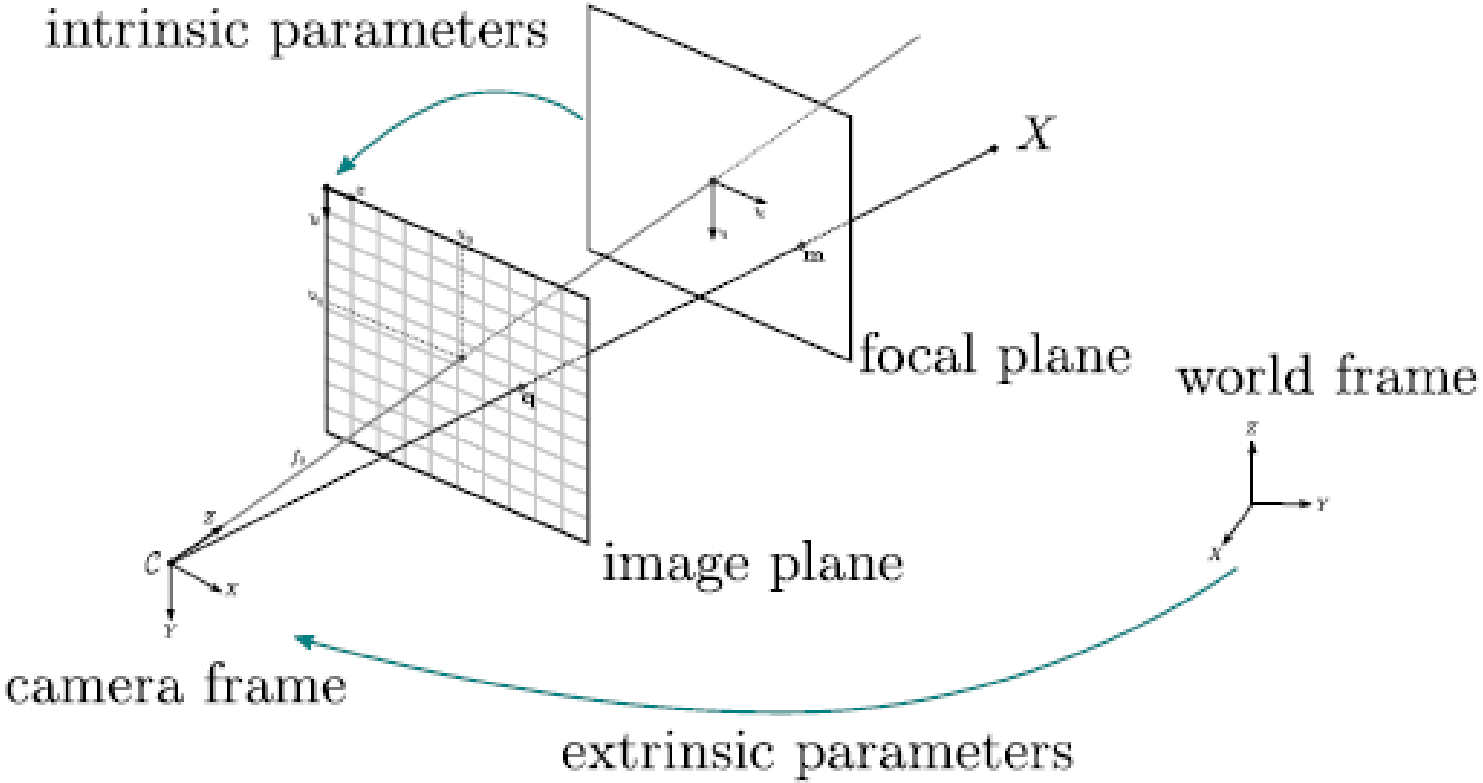
- With 3D Scanning ☺

Color Image (RGB)

Depth Image -> HSV visualization

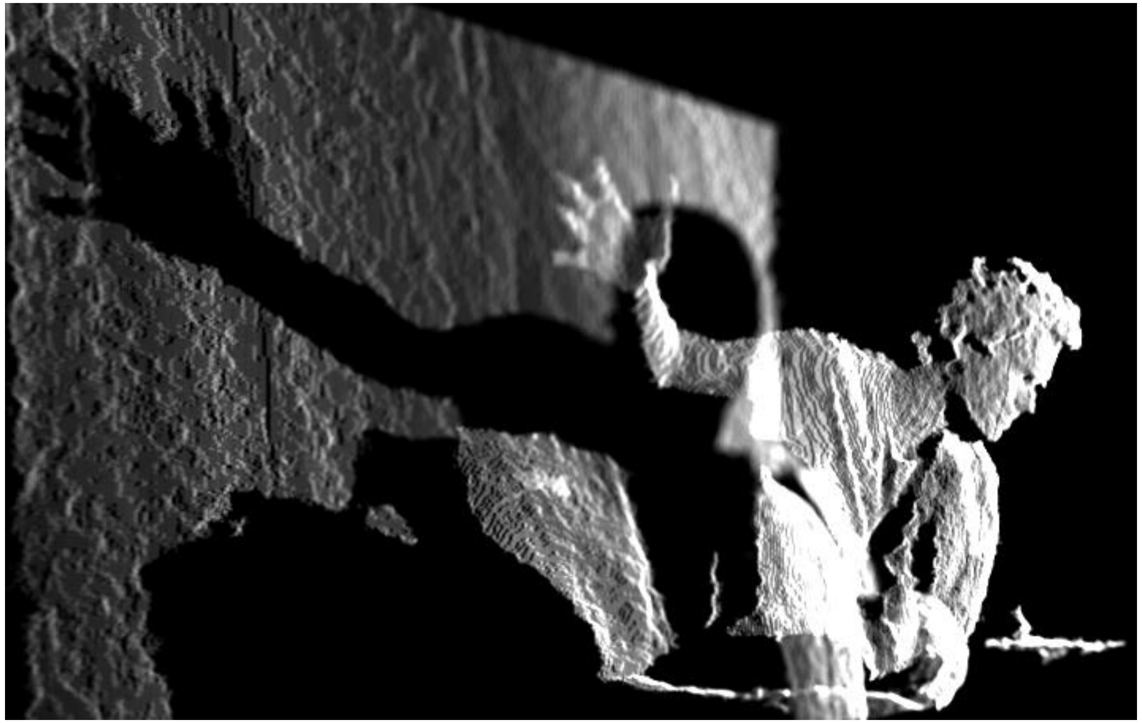
How to obtain "3D"?

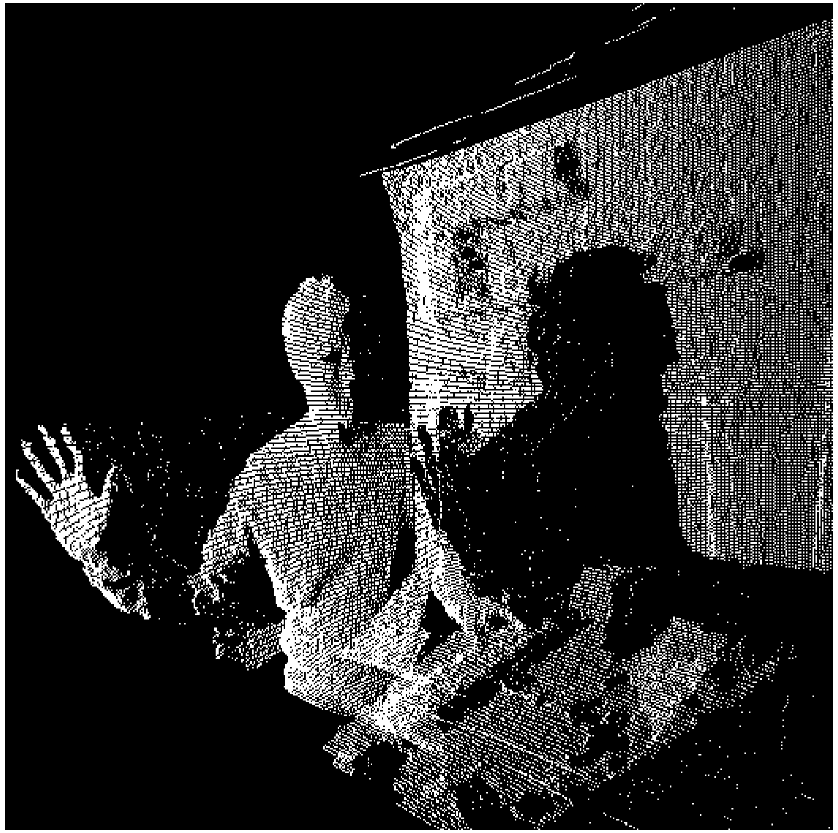
- With 3D Scanning ☺



How to obtain "3D"?

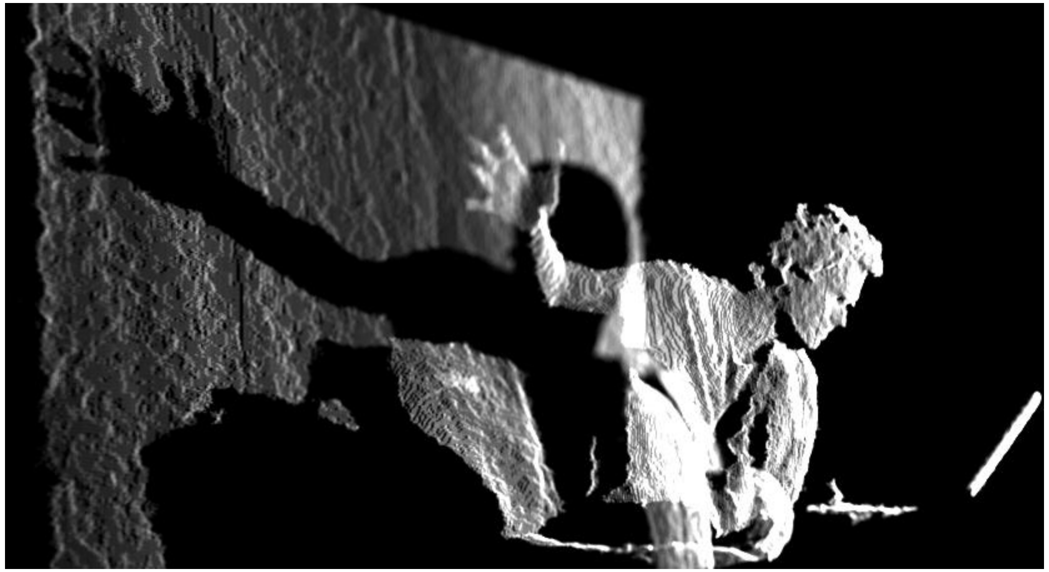
- With 3D Scanning ☺





Depth Image from the 'side'

Depth Cameras



Capture Devices

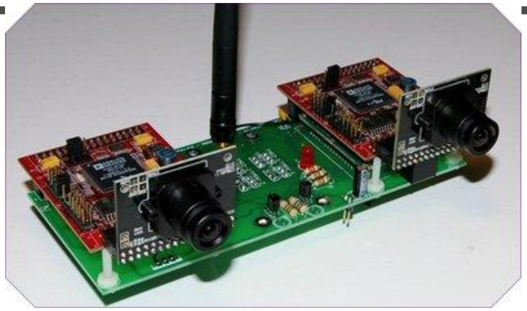
- Passive:
  - RGB
  - Stereo and Multi-view

- Active:
  - Time of Flight (ToF)
  - Structured Light
  - Laser Scanner, LIDAR

Depth Cameras

Kinect.v1: structured light

Kinect.v2: time of flight



stereo

active stereo

light detection and ranging

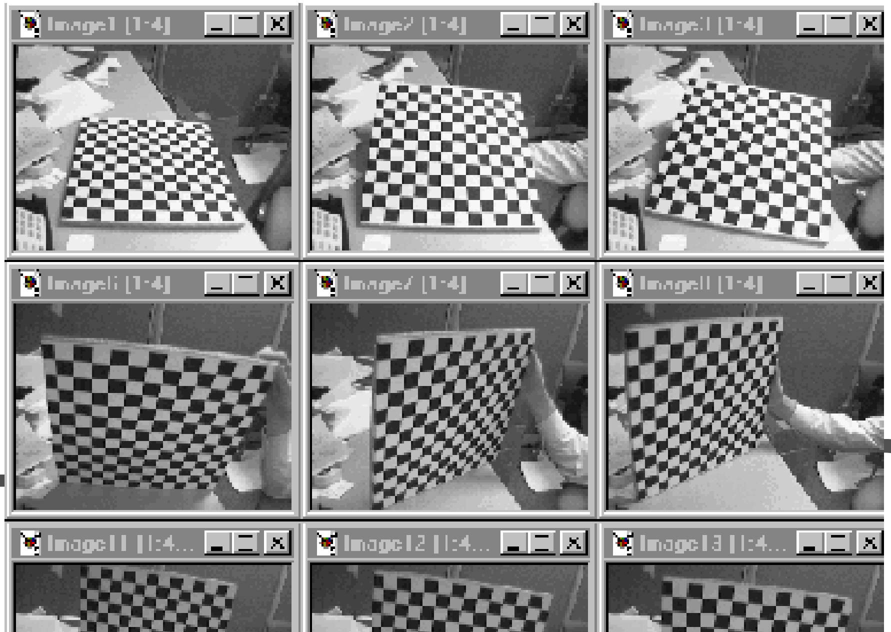
laser scanner (Cyberware)

Building a Stereo Depth Camera

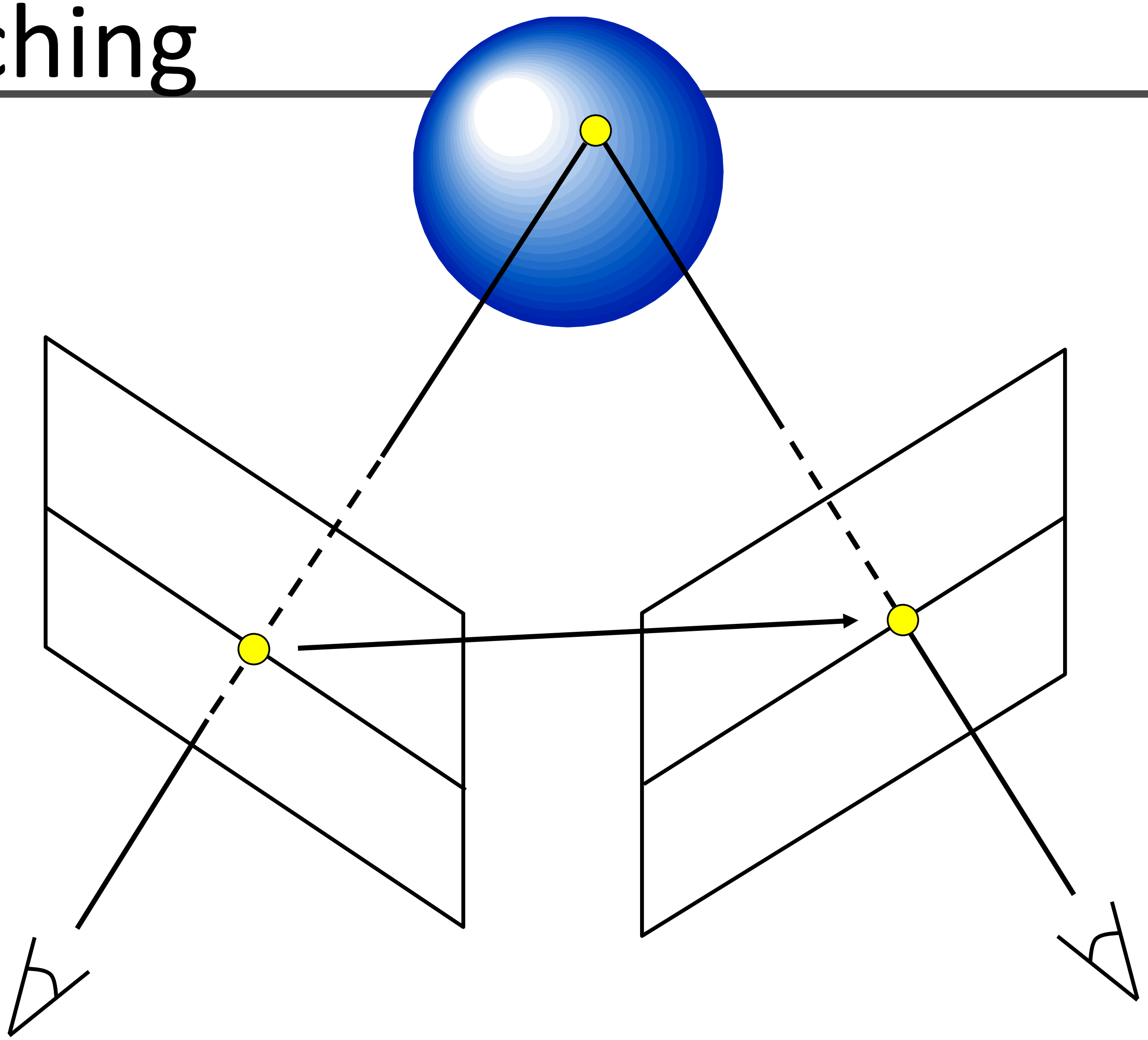
- Two RGB cameras
- Sync video capture

- Calibrate cameras

- Intrinsics (i.e., projection)
  - $\begin{bmatrix} f_x & \gamma & m_x \\ 0 & f_y & m_y \\ 0 & 0 & 1 \end{bmatrix}$
- Extrinsics (from A to B)
  - $A = [R|t]$

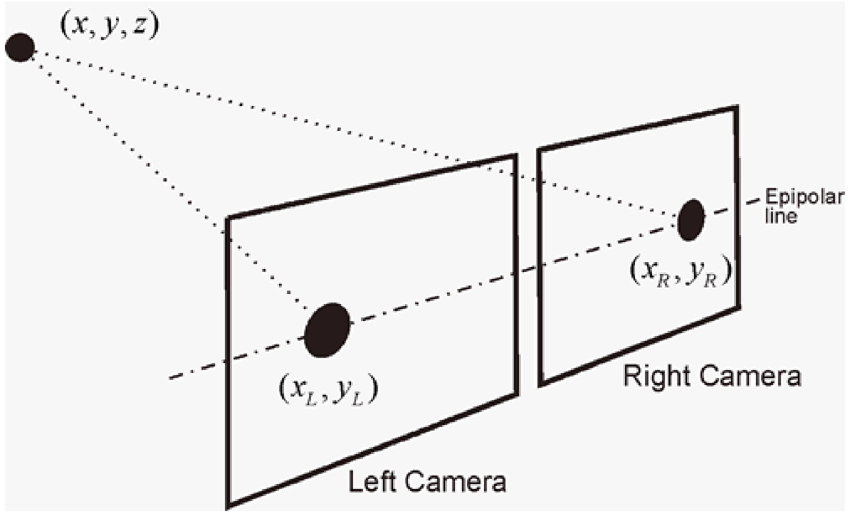


Stereo Matching



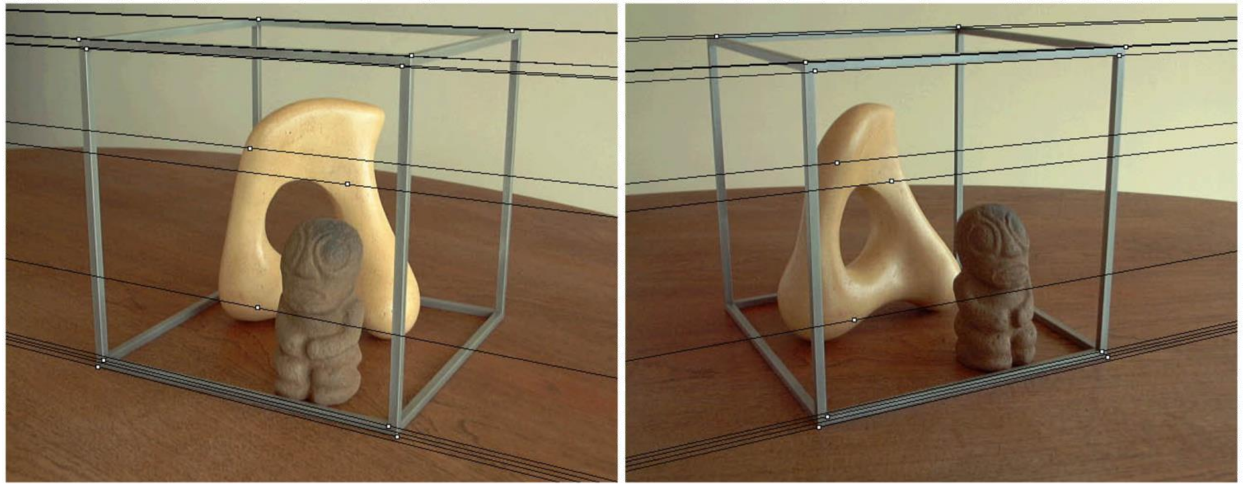
[S. Narasimhan]

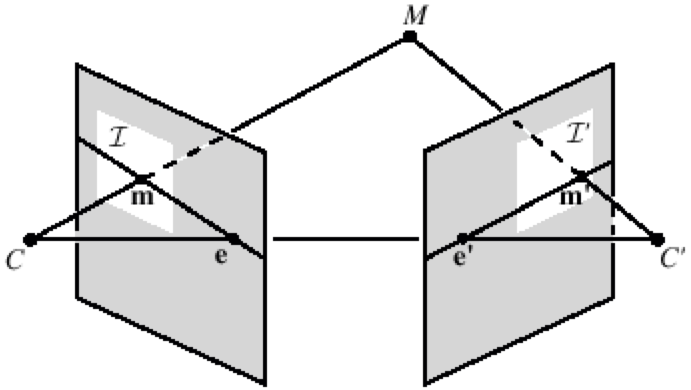
(Passive) Stereo

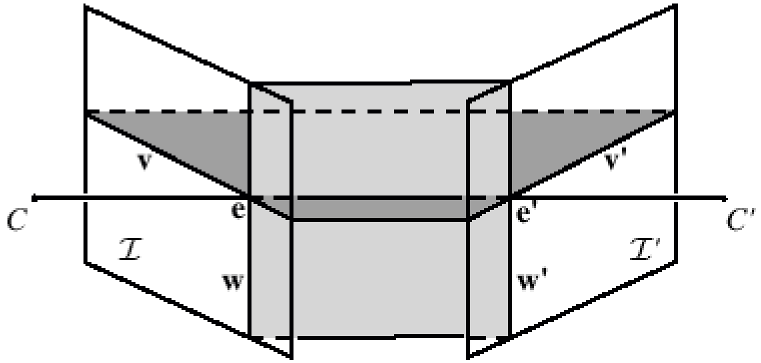


- Triangulation using epipolar geometry
- Finding correspondences is a hard problem!

Stereo Matching: Rectification

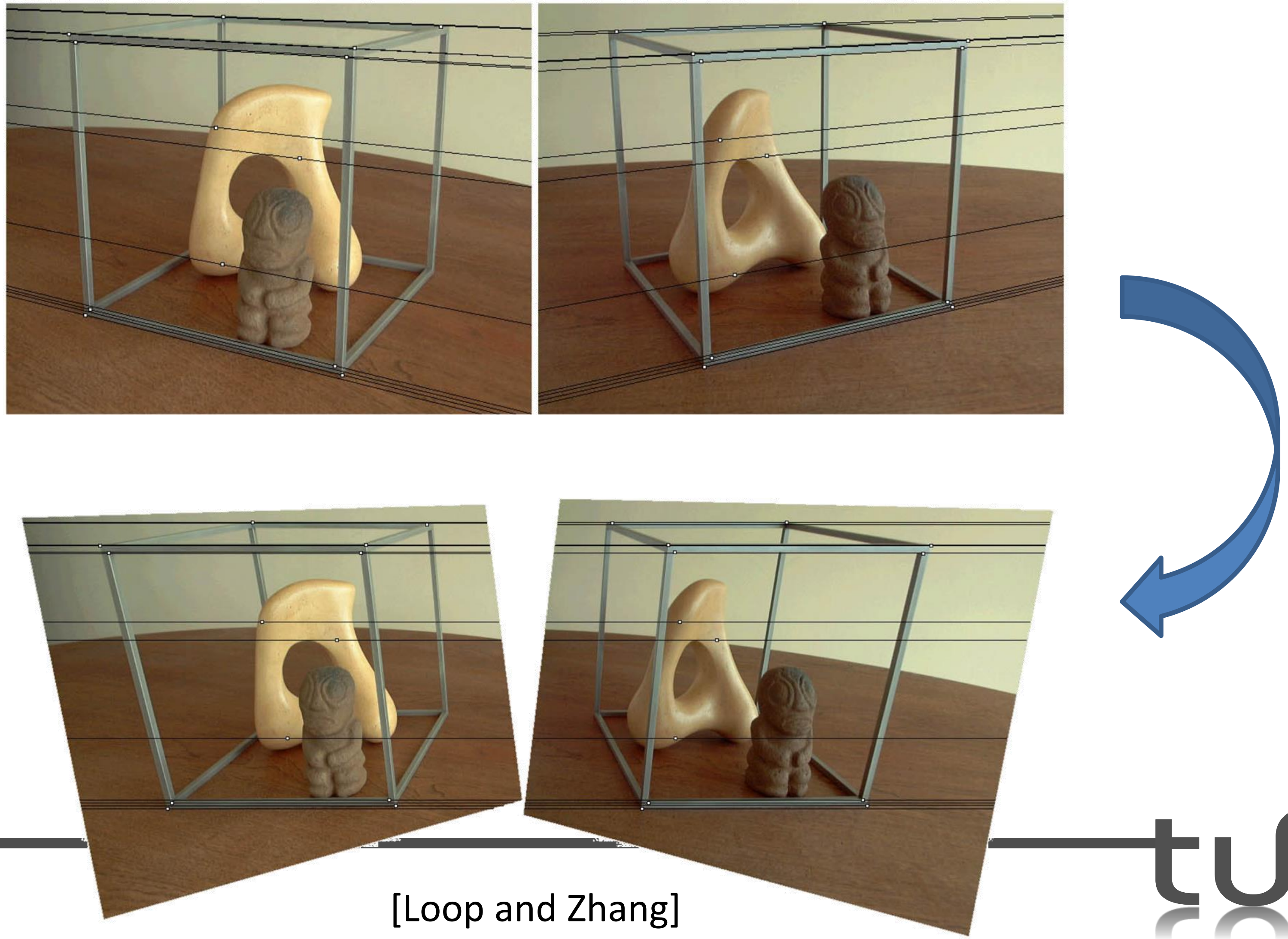






[Loop and Zhang]

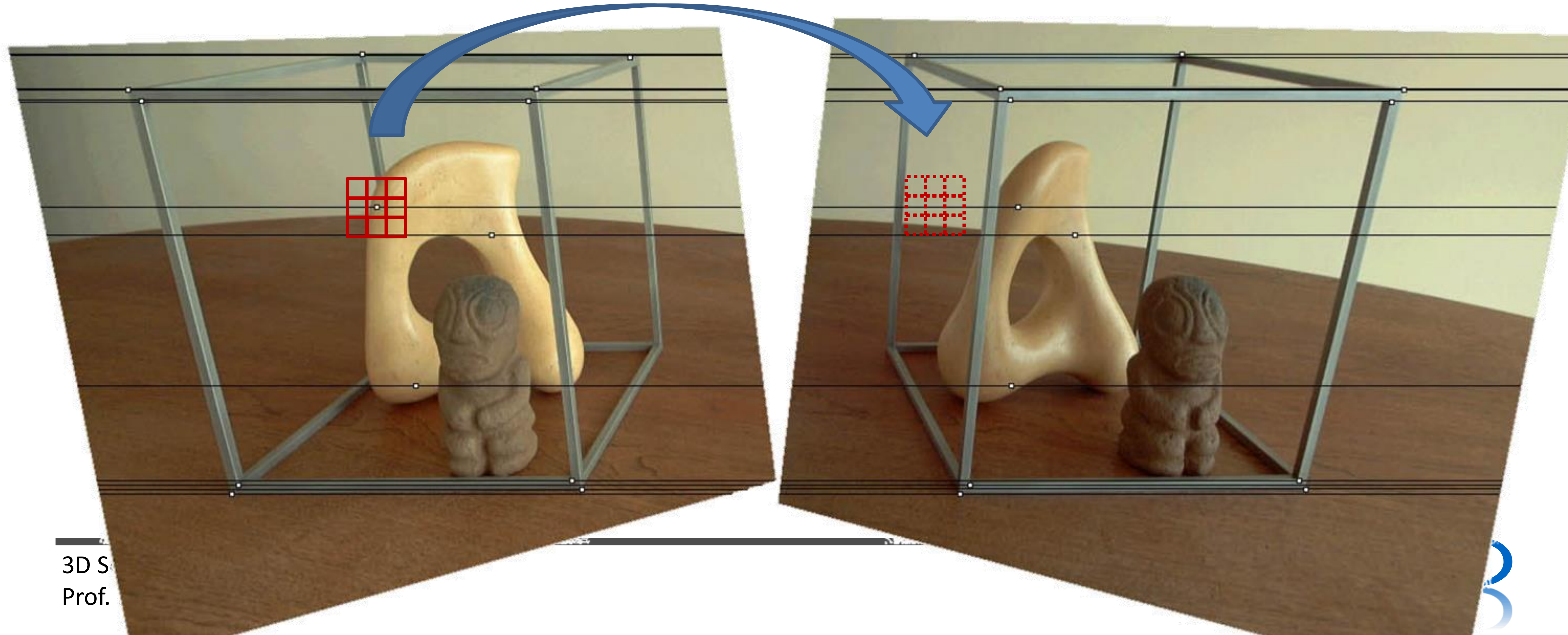
Stereo Matching: Rectification



[Loop and Zhang]

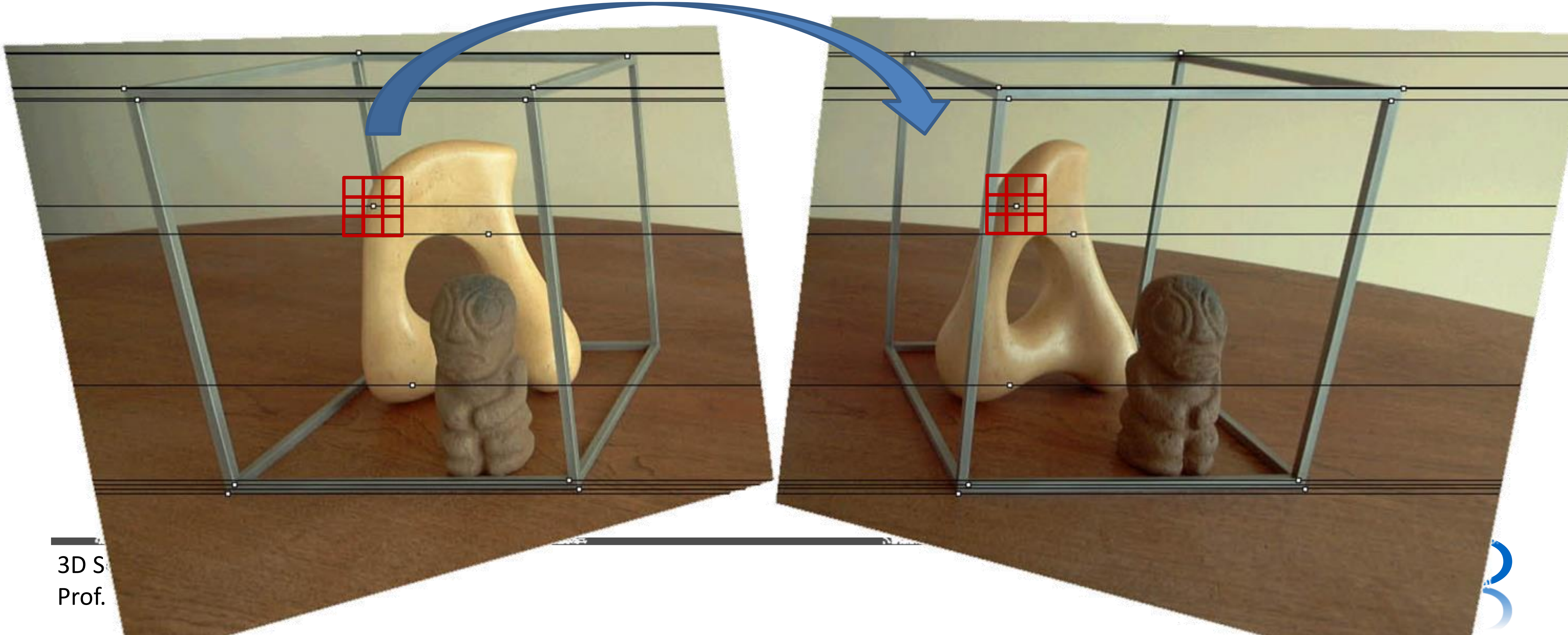
Stereo Matching: Search

for each pixel search along epipolar line to find match



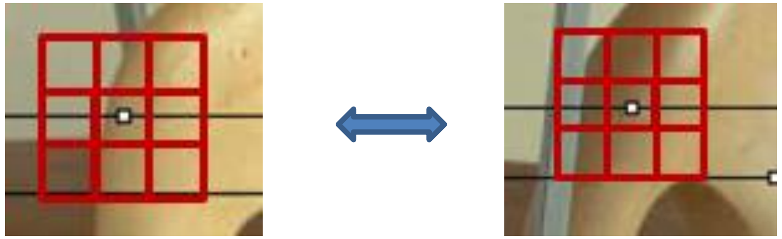
Stereo Matching: Search

use 2D feature descriptor of choice to determine best match



Stereo Matching

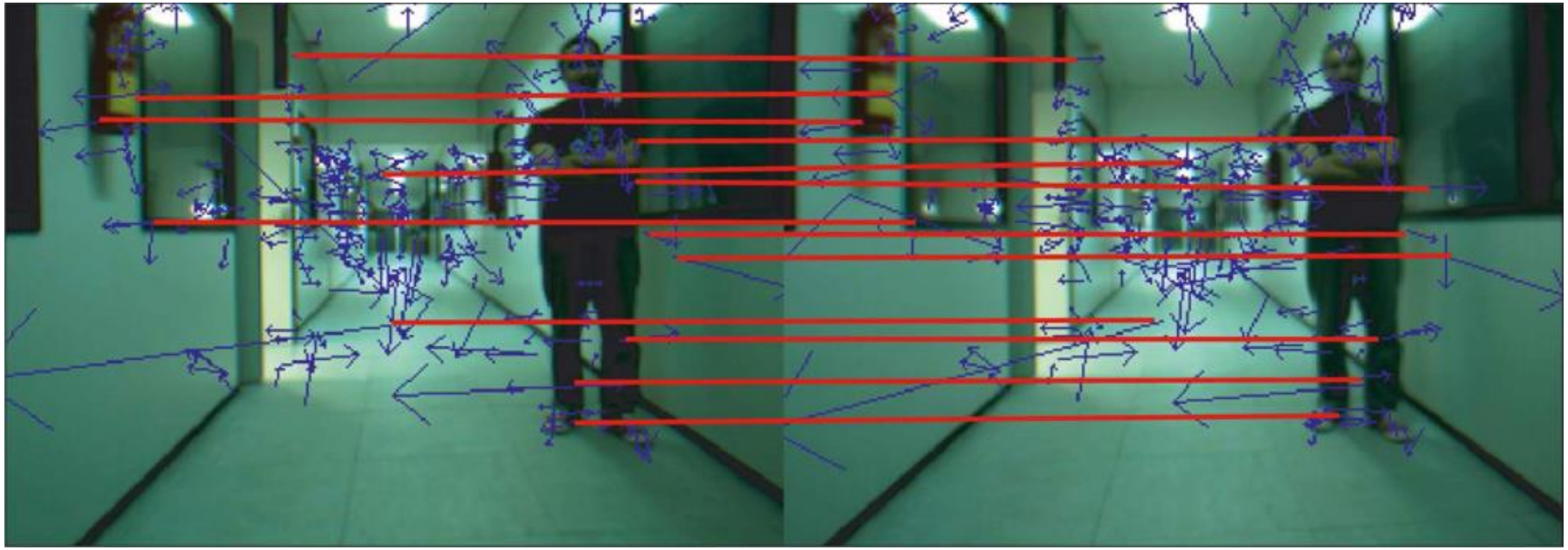
- $SSD(u,v) = \sum_{(u,v)} \left(I_{left}(i,j) - I_{right}(i,j) \right)^2$



- Sparse vs Dense Stereo Matching

Stereo Matching

- Sparse features: SIFT, SURF, ORB, …

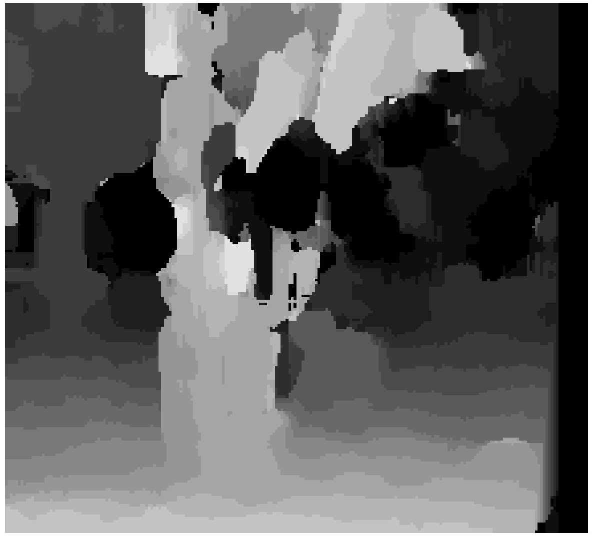


Matched SIFT features in a stereo pair (red lines indicate matches)

Stereo Matching: Search

- How big is the search window?

W = 3

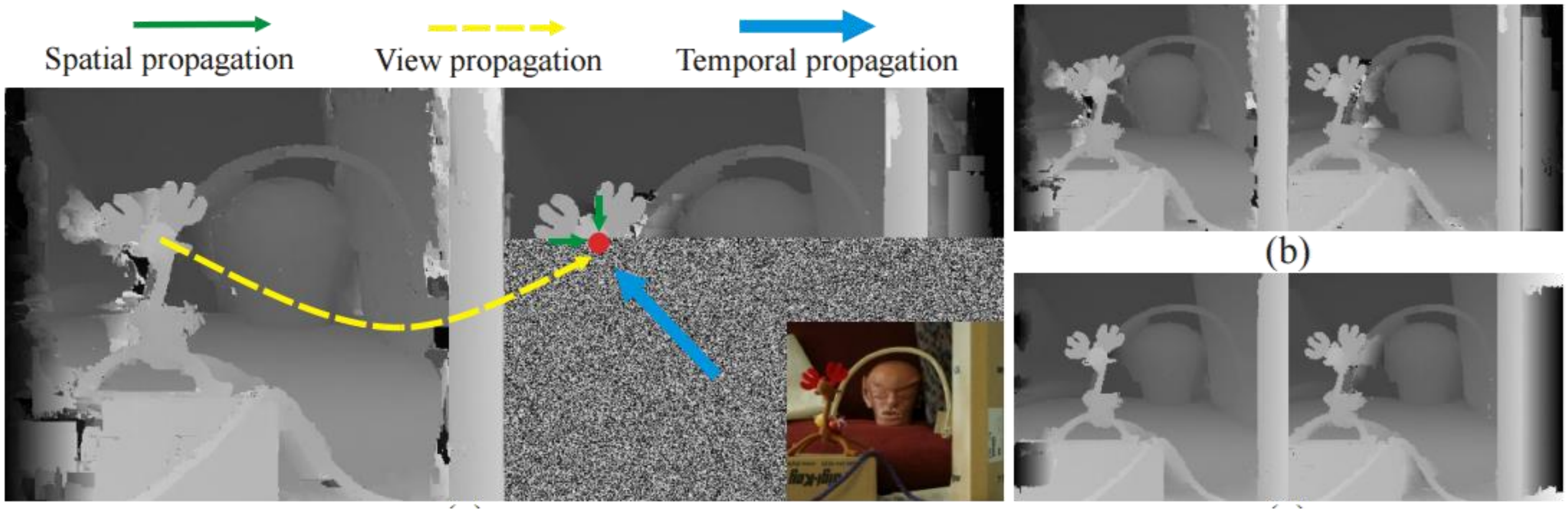


W = 20

- Smaller neighborhood -> more noise
- Larger neighborhood -> fewer details

Stereo Matching: Search

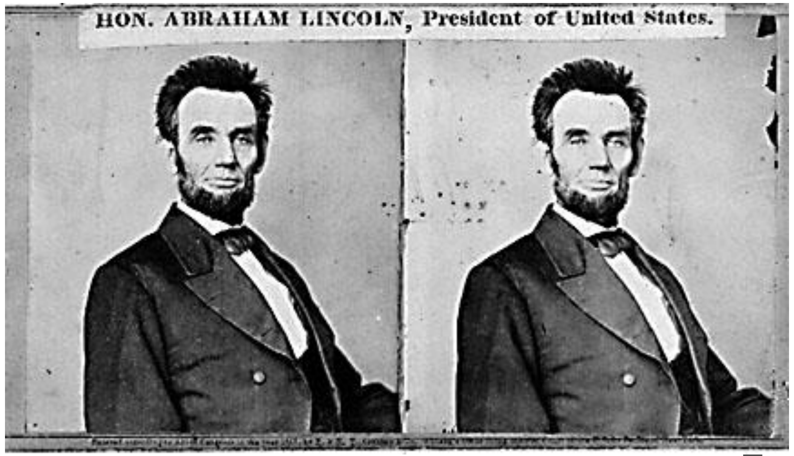
- Search time correlates w/ window size
  - PatchMatch Stereo [Barnes et al.] [Bleyer et al.]



Failures of Correspondence Search

Non-Lambertian surfaces, specularities

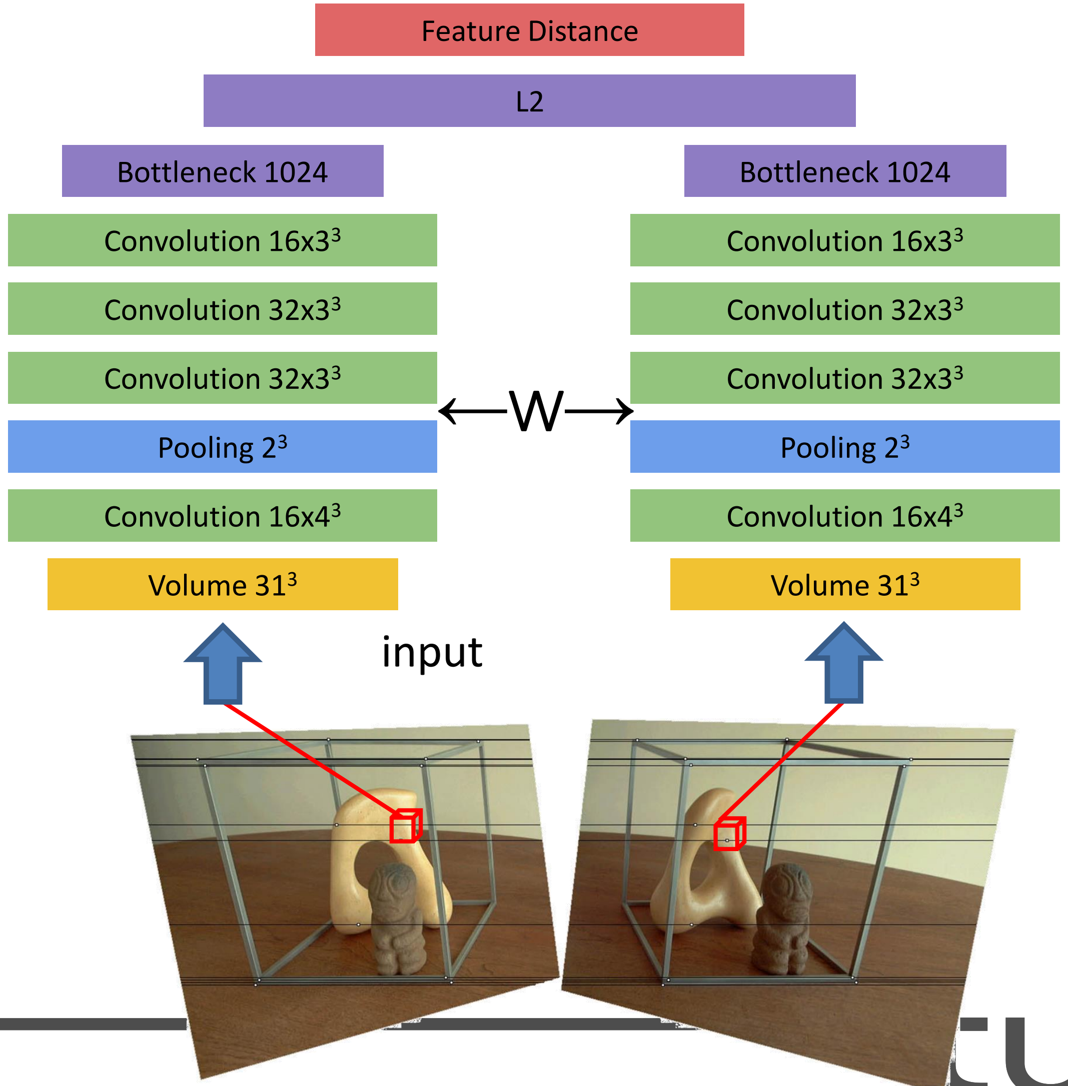
Occlusions, repetition



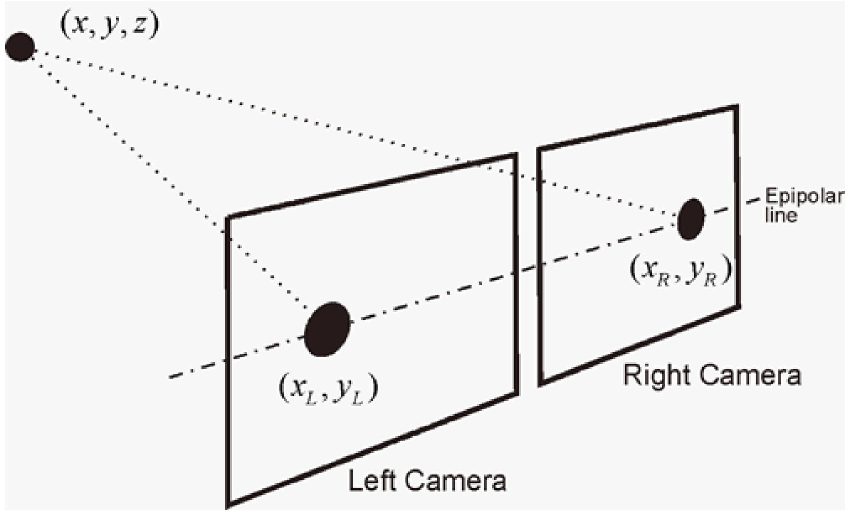
Textureless surfaces

Stereo Matching with Deep Learning

- Feature matching is ideal task for CNN
  - For instance, use Siamese Networks
  - Many papers recently!



(Passive) Stereo



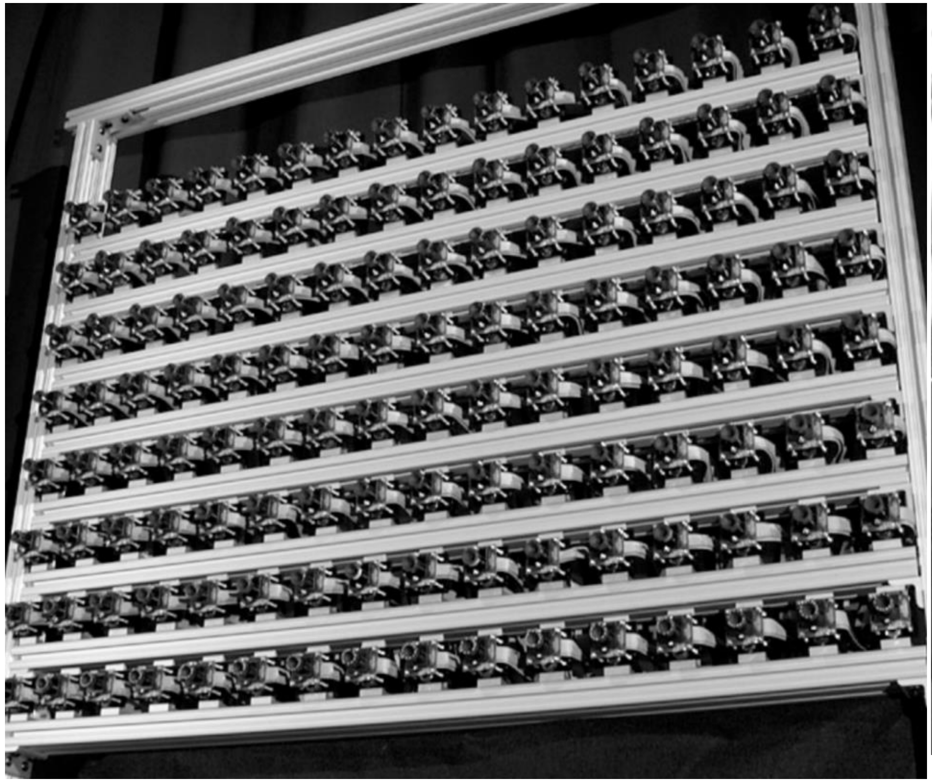
- Triangulation using epipolar geometry
- $z = \frac{fT_x}{d}$

Stereo Matching: Baseline

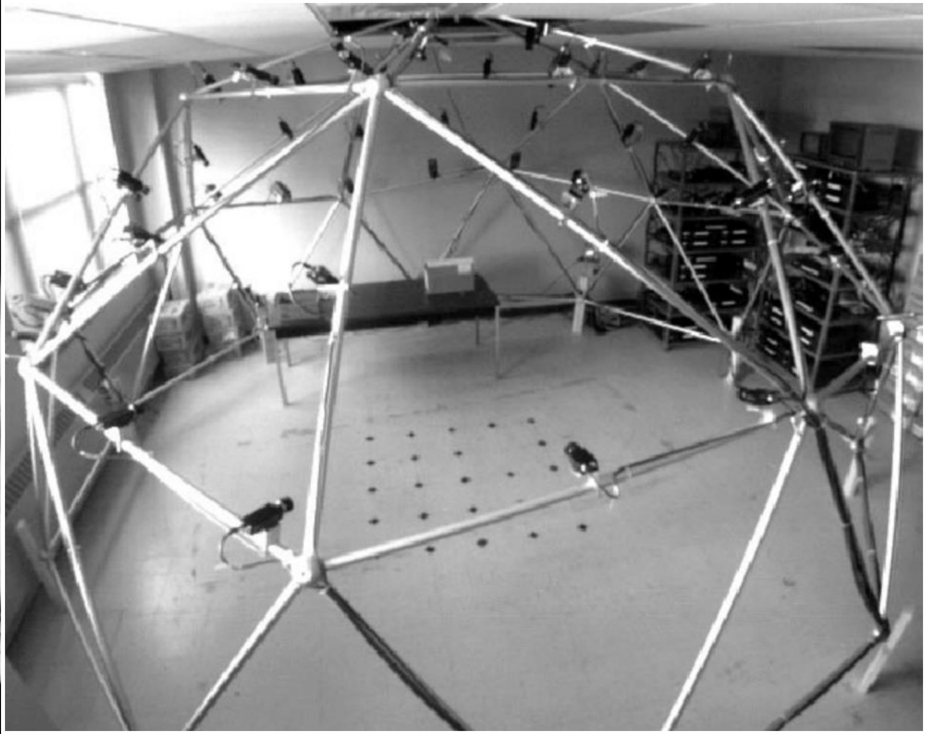
- Wide baseline -> harder to find matches
  - More distortion
  - More occlusions

- Small baseline -> matches are less accurate
  - Small disparity error results in large distance
  - Disparity needs to be sub-pixel accurate

Multi-view Stereo



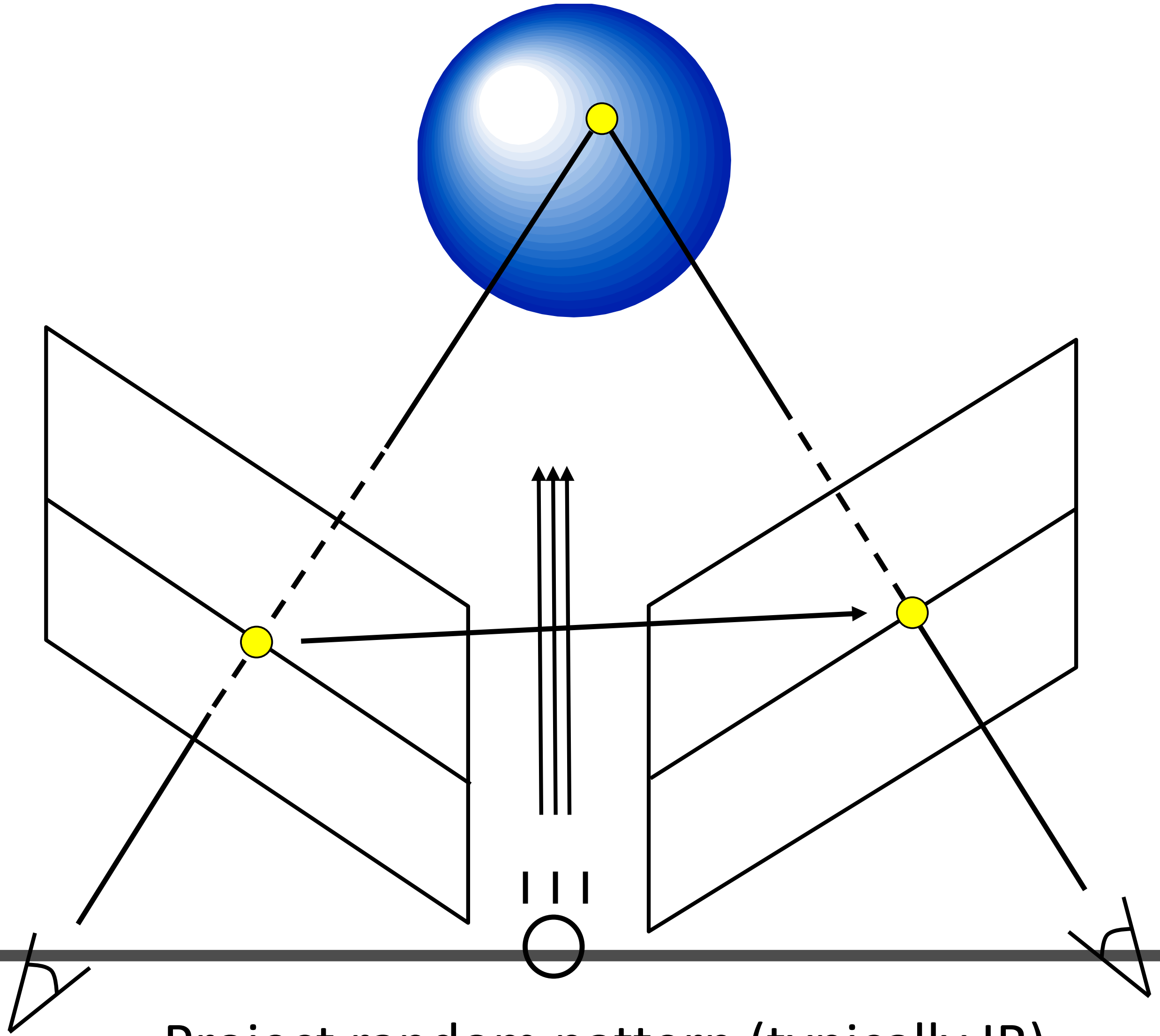
Stanford multi-camera array



CMU multi-camera stereo

(Active) Stereo

(Active) Stereo

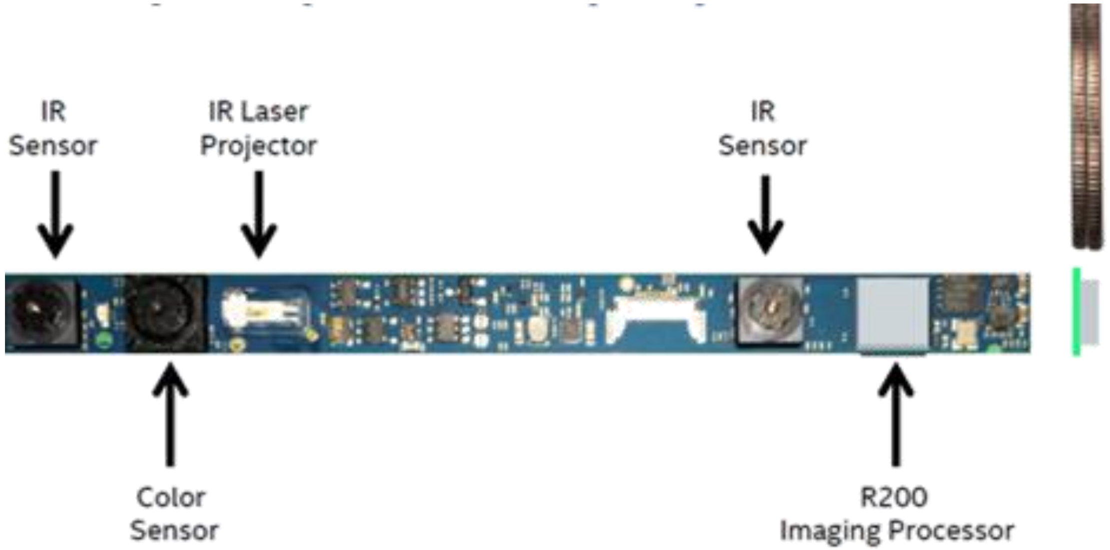


Project random pattern (typically IR)

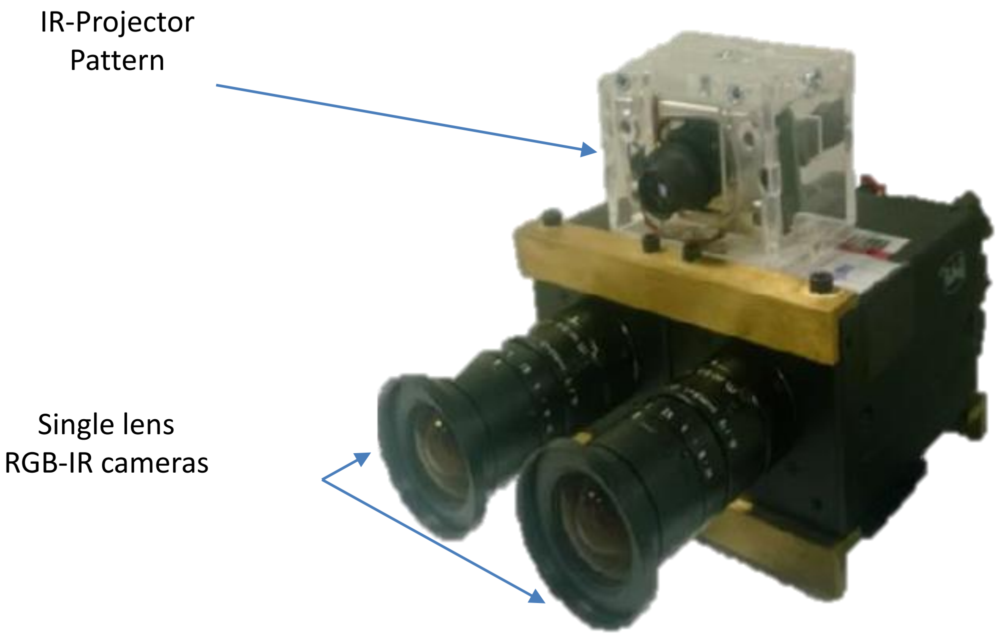
[S. Narasimhan]

Stereo Camera: Real Sense

- Active mode (close range, indoor)
- Passive mode (outdoor)



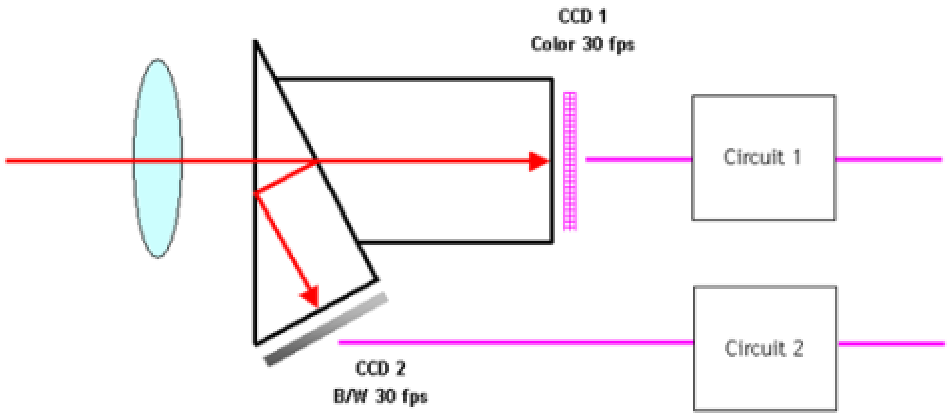
(Active) Stereo



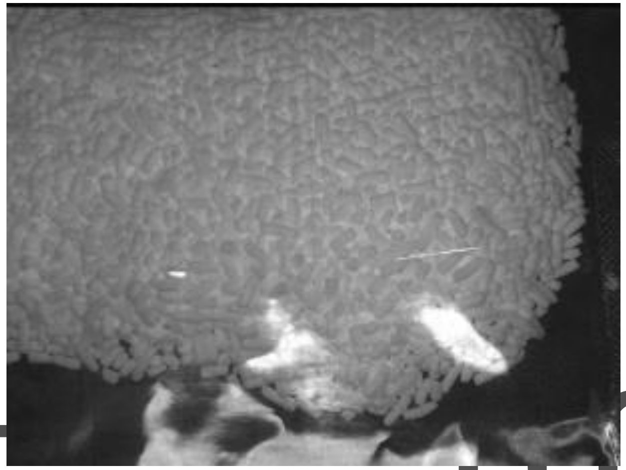
Patchmatch Stereo [Bleyer11]

(Active) Stereo

- Beam splitter



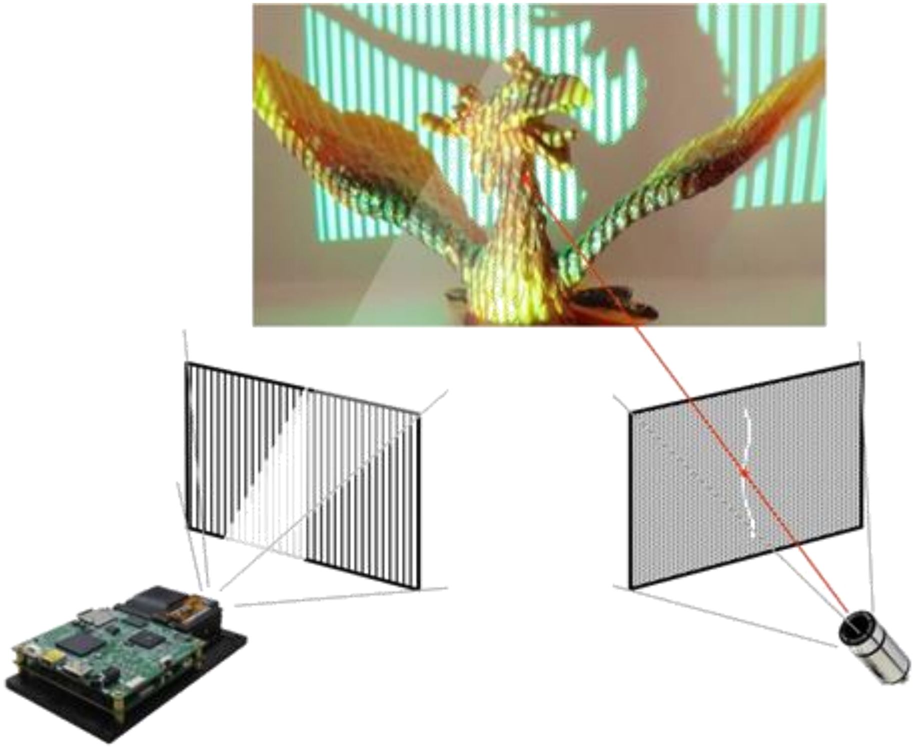
visible light



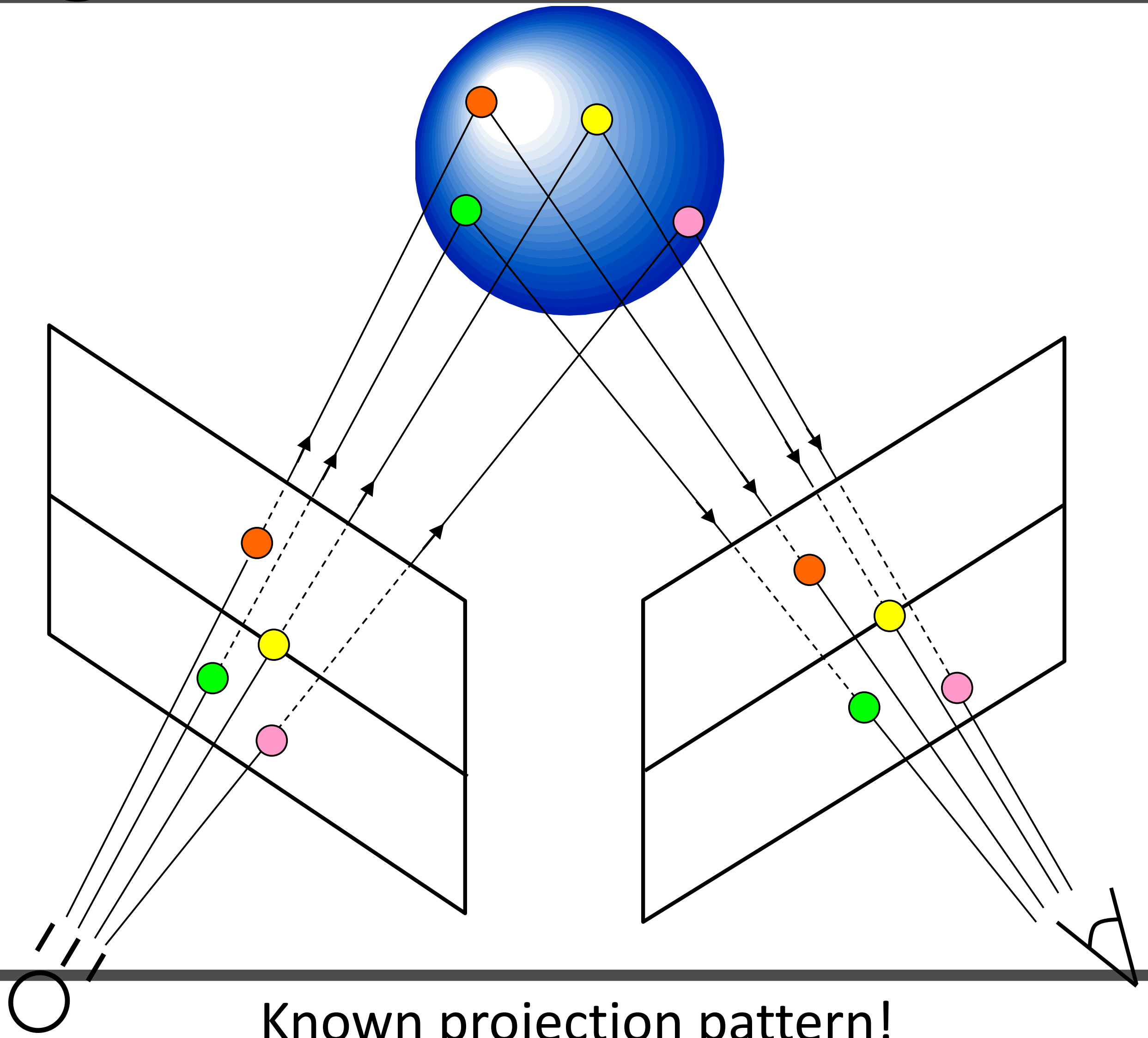
infrared

Structured Light

Structured Light



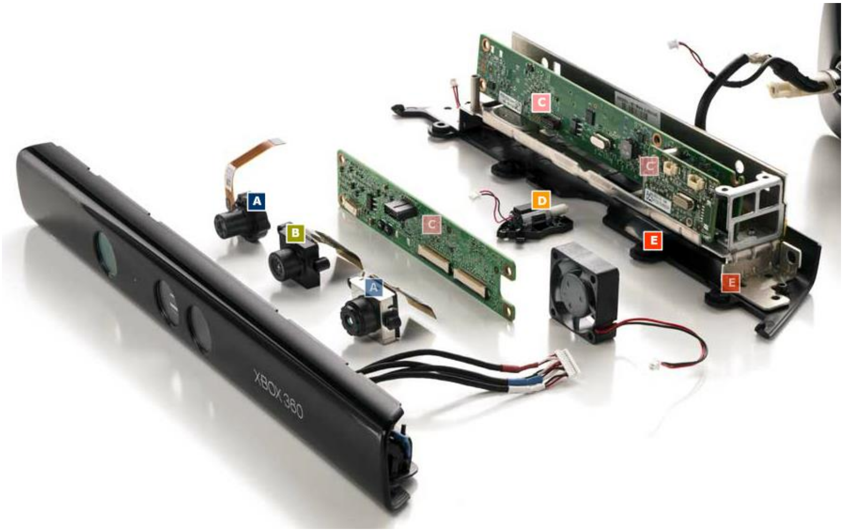
Structured Light

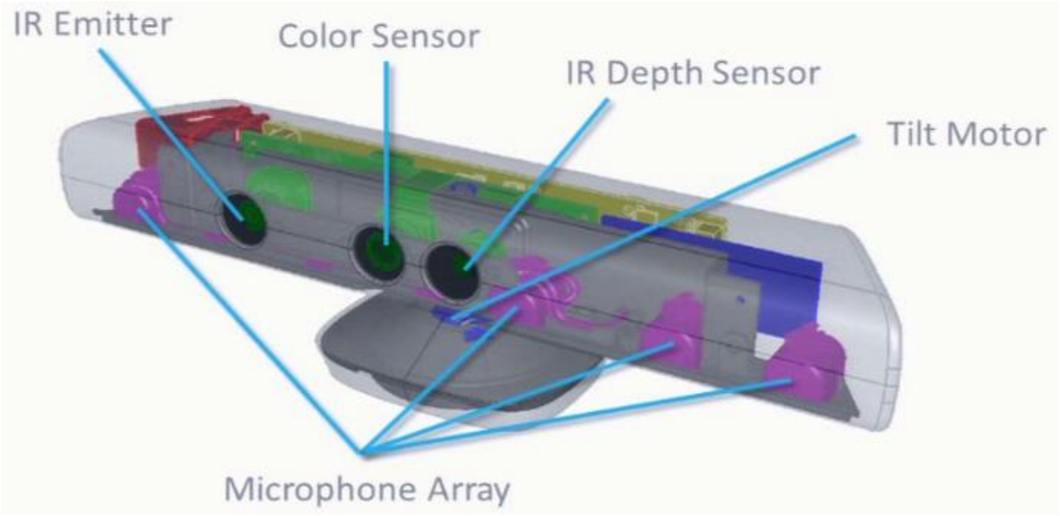


Known projection pattern!

[S. Narasimhan]

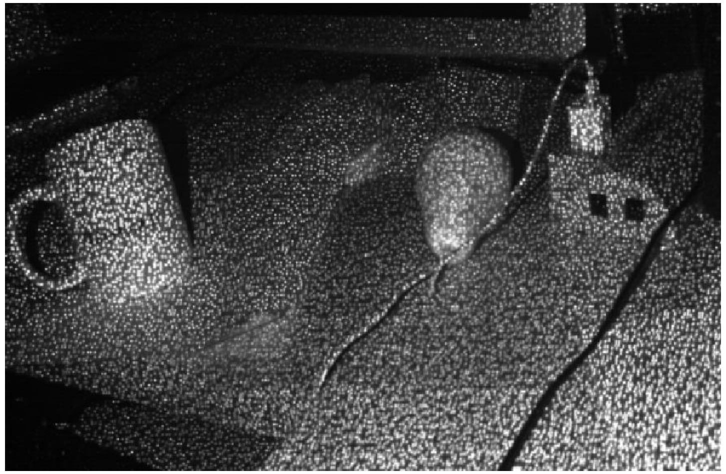
Structured Light: Kinect.V1



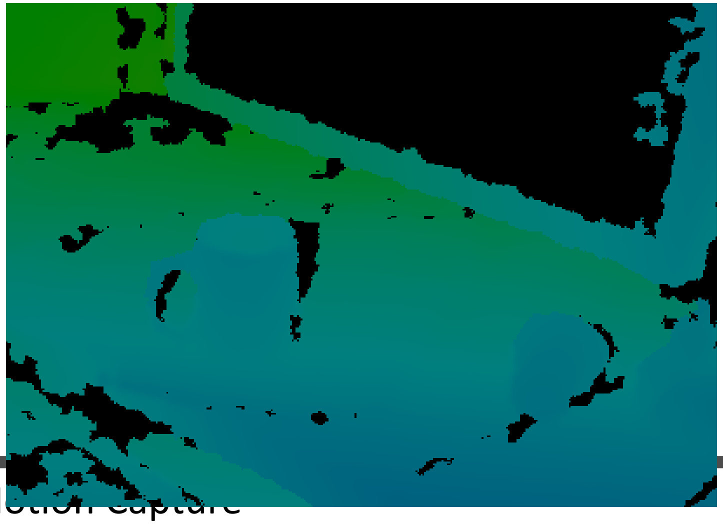


Structured Light: Kinect.V1

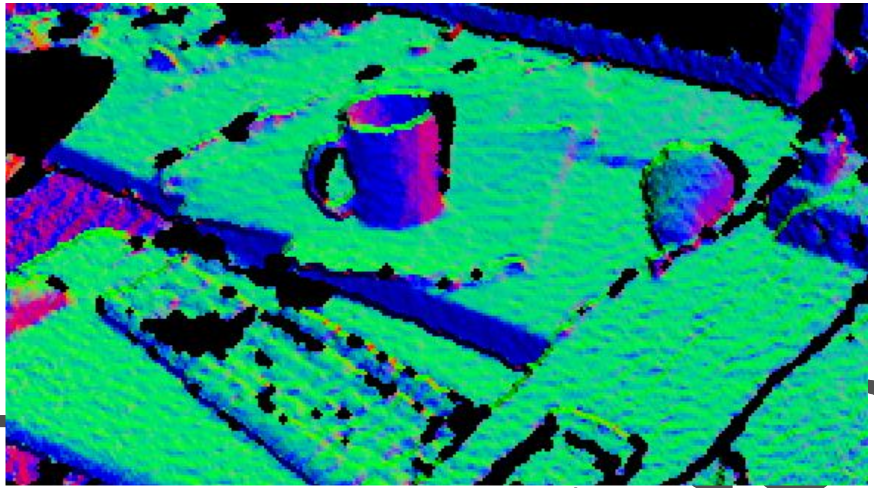
Calibrated IR pattern



Input RGB

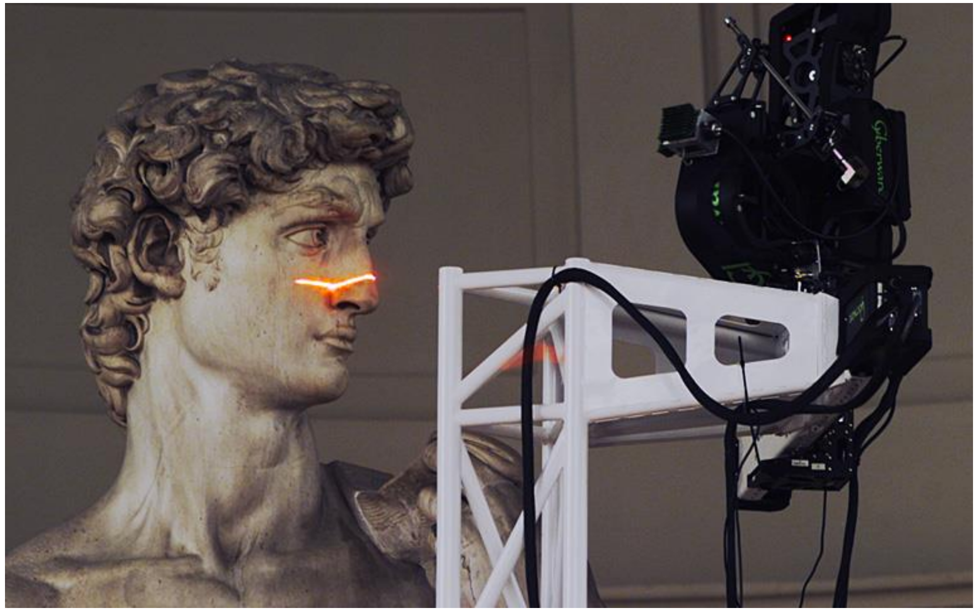


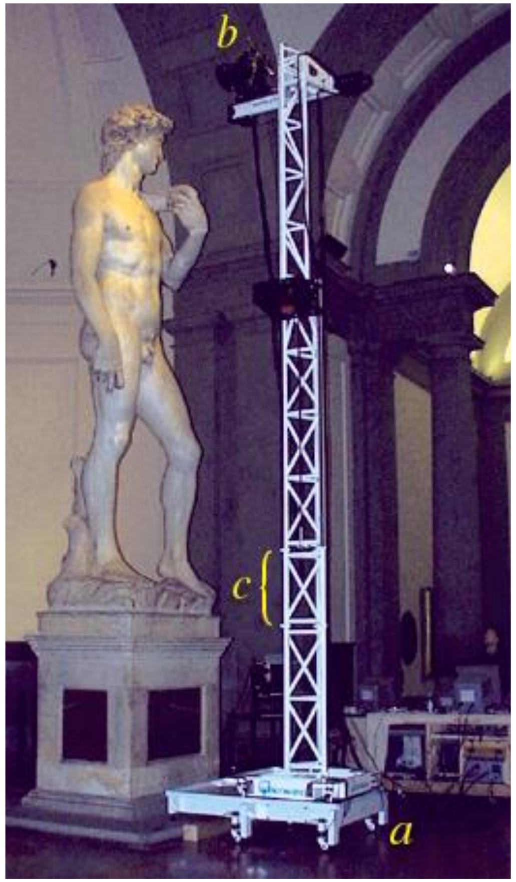
Input depth



Input normals

Laser Scanning

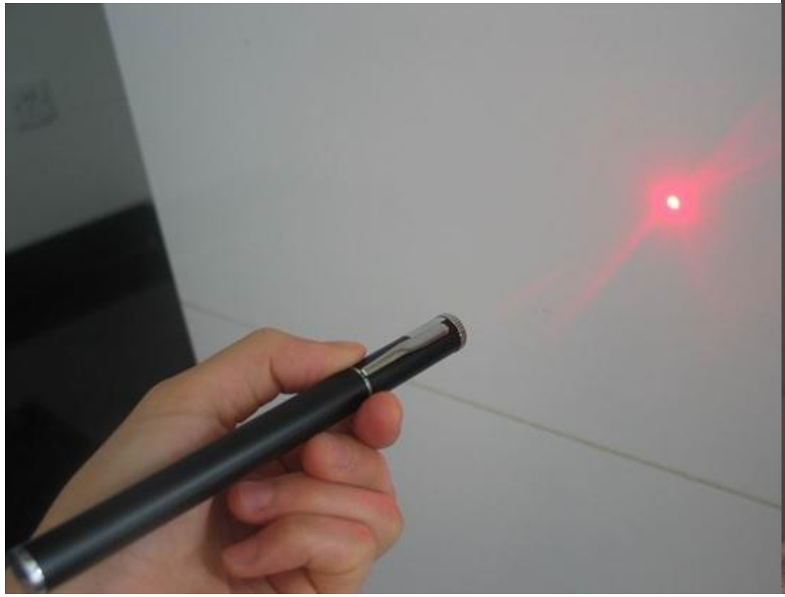


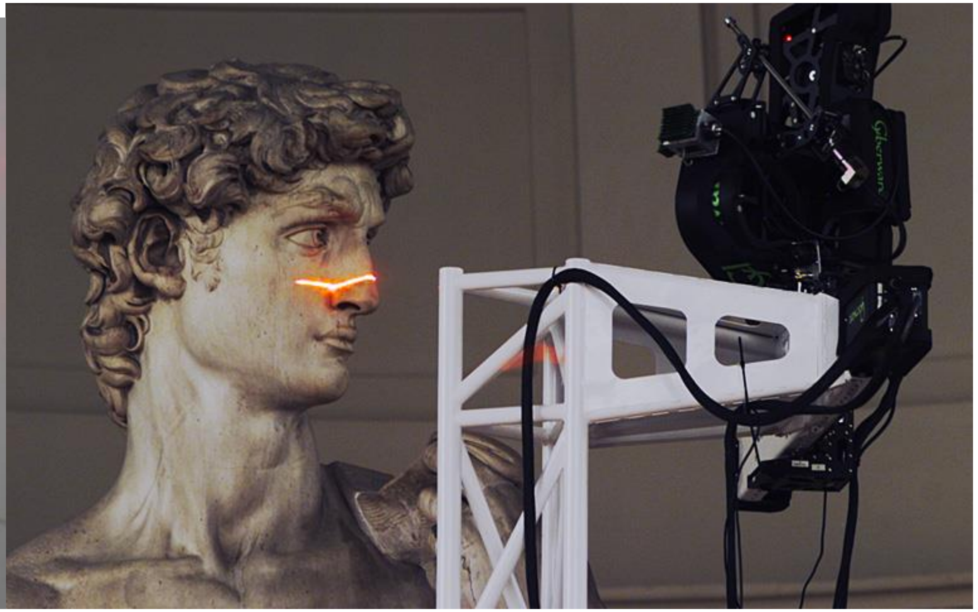


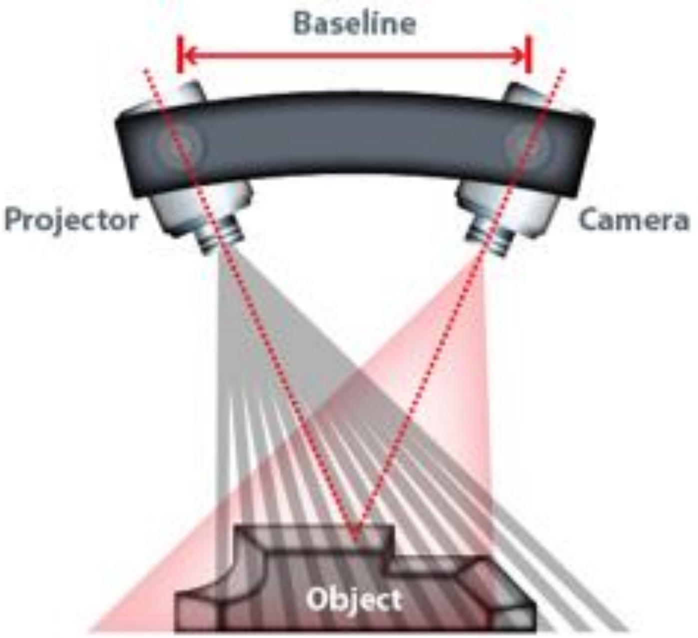
The Digital Michelangelo Project [Levoy et al.]

Laser Scanning

- Simple version: calibrate -> laser pointer + camera
- Common: Scanline-by-scanline







Laser Scanning

High-end laser scan (point cloud)

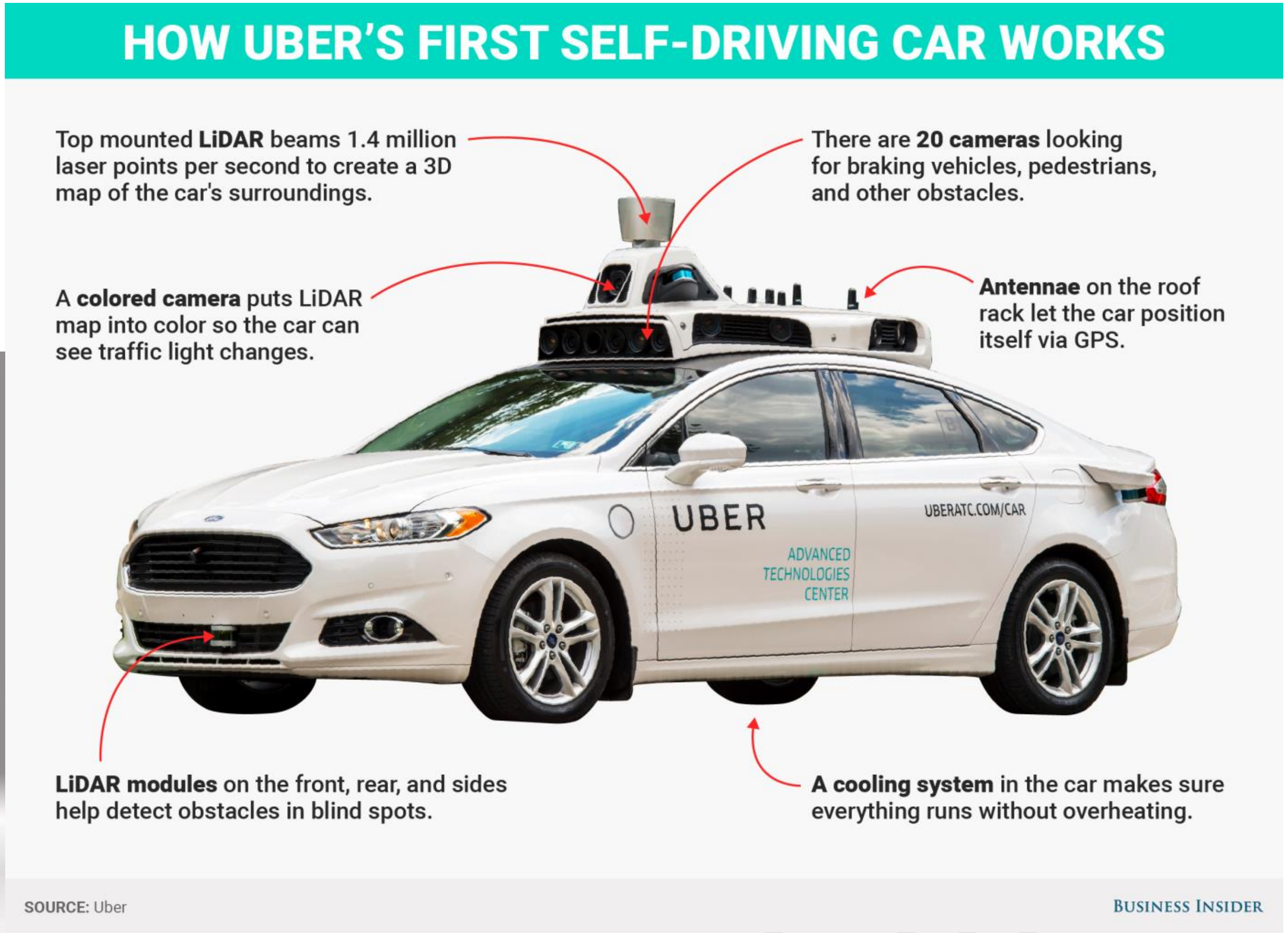
LIDAR

- *Light Imaging, Detection, And Ranging (LIDAR)*

From street view to autonomous cars

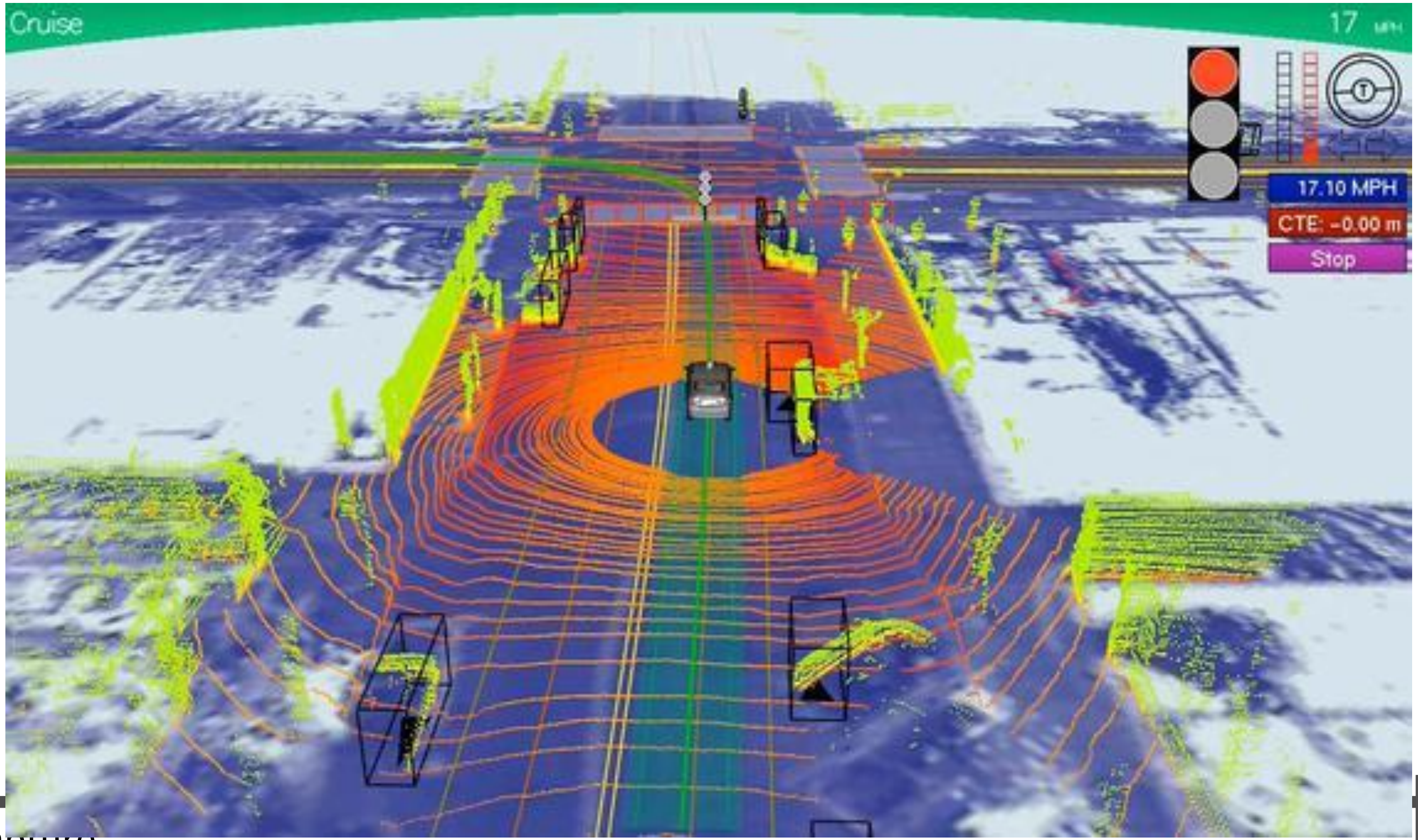
LIDAR


WAYMO

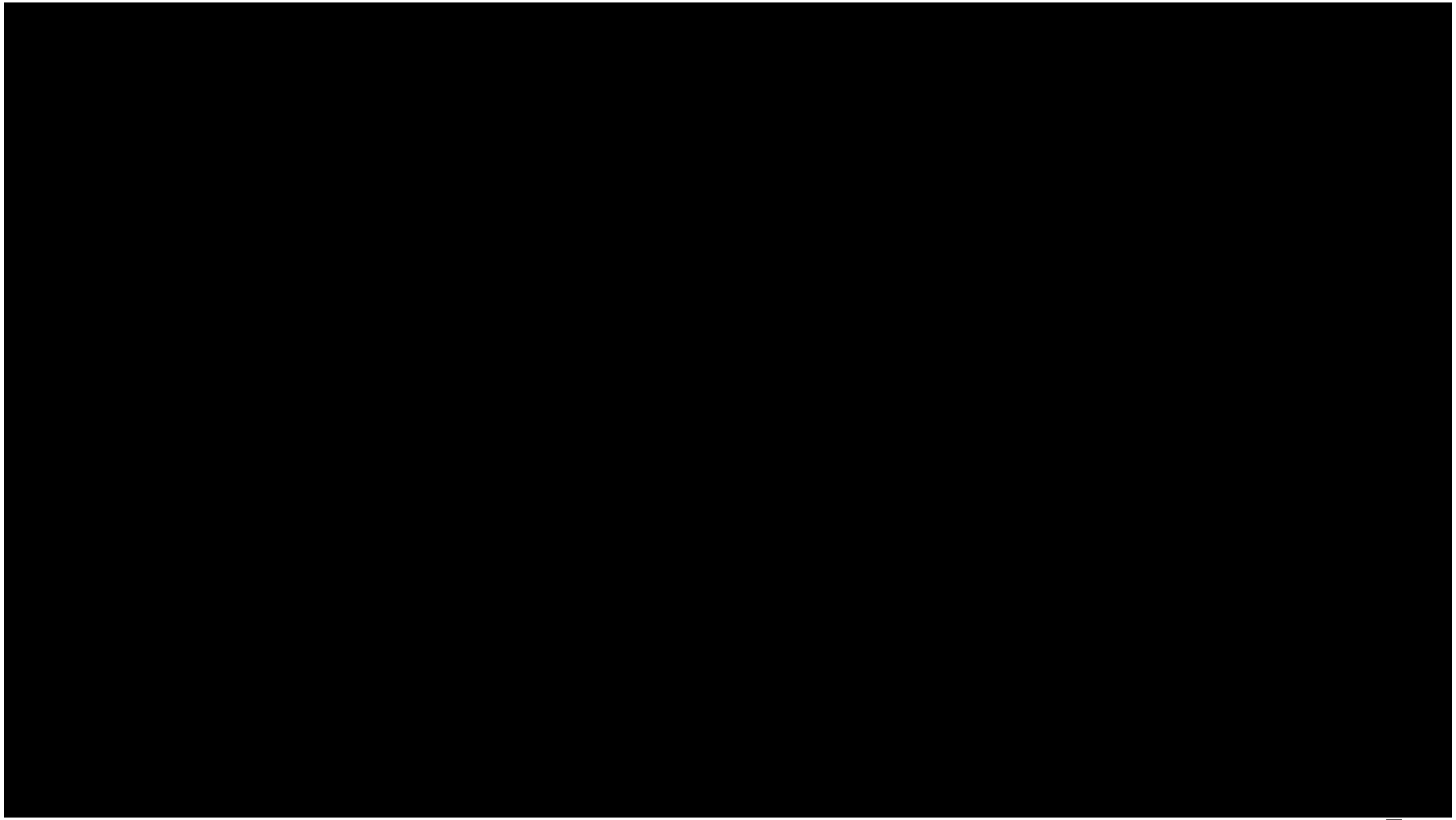

HOW UBER'S FIRST SELF-DRIVING CAR WORKS
Top mounted LiDAR beams 1.4 million laser points per second to create a 3D map of the car's surroundings.
There are 20 cameras looking for braking vehicles, pedestrians, and other obstacles.
A colored camera puts LiDAR map into color so the car can see traffic light changes.
Antennae on the roof rack let the car position itself via GPS.
UBER
ADVANCED TECHNOLOGIES CENTER
UBERATC.COM/CAR
LiDAR modules on the front, rear, and sides help detect obstacles in blind spots.
A cooling system in the car makes sure everything runs without overheating.
SOURCE: Uber
BUSINESS INSIDER

Laser Scanning: Mapping the Environment

- Lidar in self-driving cars



Laser Scanning for 3D Printing



Depth Cameras

- Stereo and Multi-view
  - Passive: work indoors and outdoors
  - Rely on features from the environment
  - Computationally expensive due to feature matching step
- Time of Flight (ToF)
  - Active: can map featureless regions
  - Often in IR spectrum -> fails outdoors
  - Sensitive to scattering, indirect lighting, etc.
- Structured Light
  - Active: can map featureless regions
  - Often in IR spectrum -> fails outdoors
  - Need precise calibration between projector and sensor
- Laser Scanner, Lidar
  - Only a small scanline -> slow (if faster, only sparse points; e.g., LIDAR)
  - Very precise though because feature matching is trivial

Applications & Research Directions

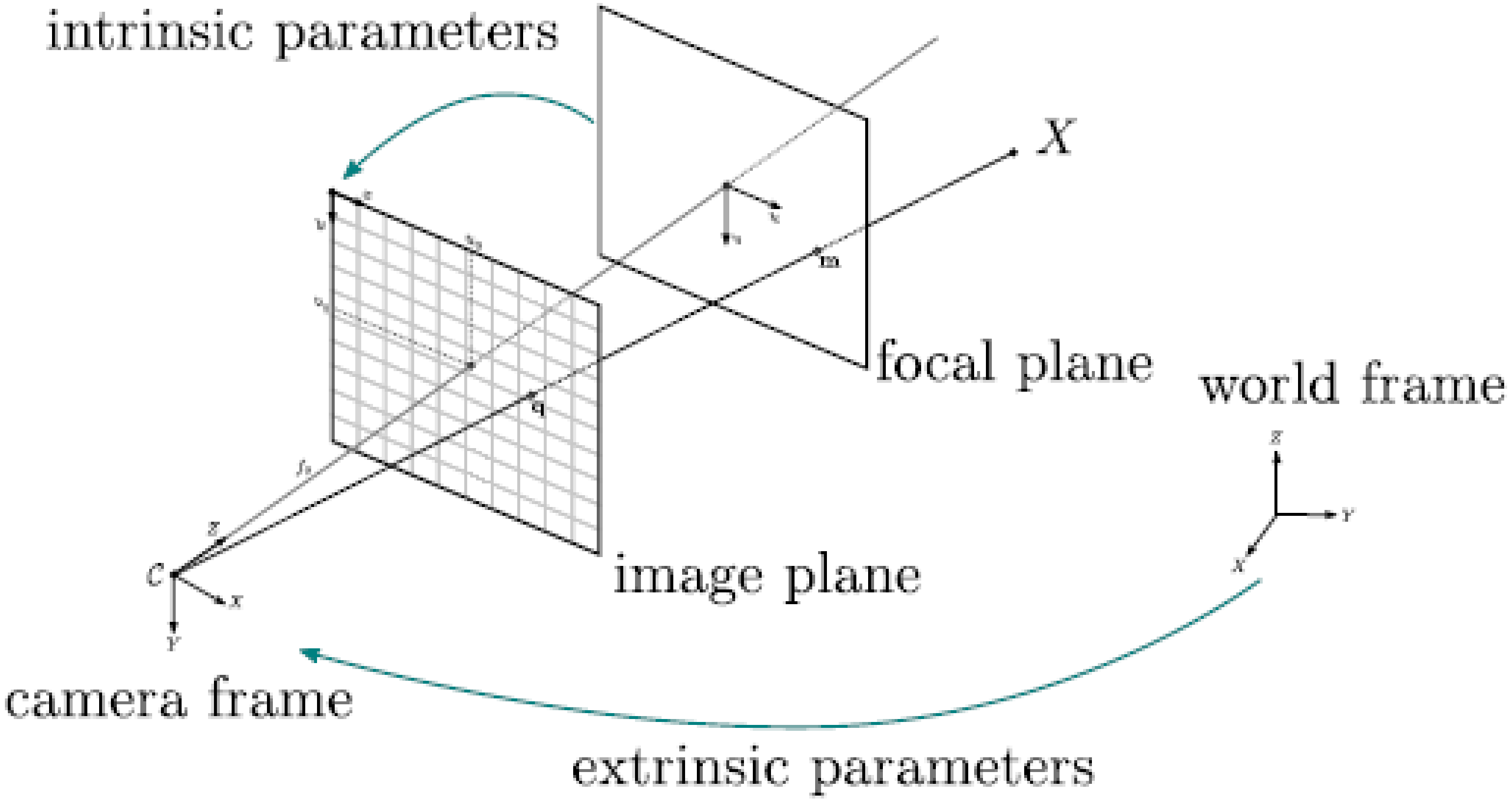
- Body tracking
- Gesture control
- Face tracking
- 3D reconstruction
- Localization
- Scene understanding
- Virtual & augmented reality
- Fabrication
- Neural Rendering
- And many more ☺

self-driving cars

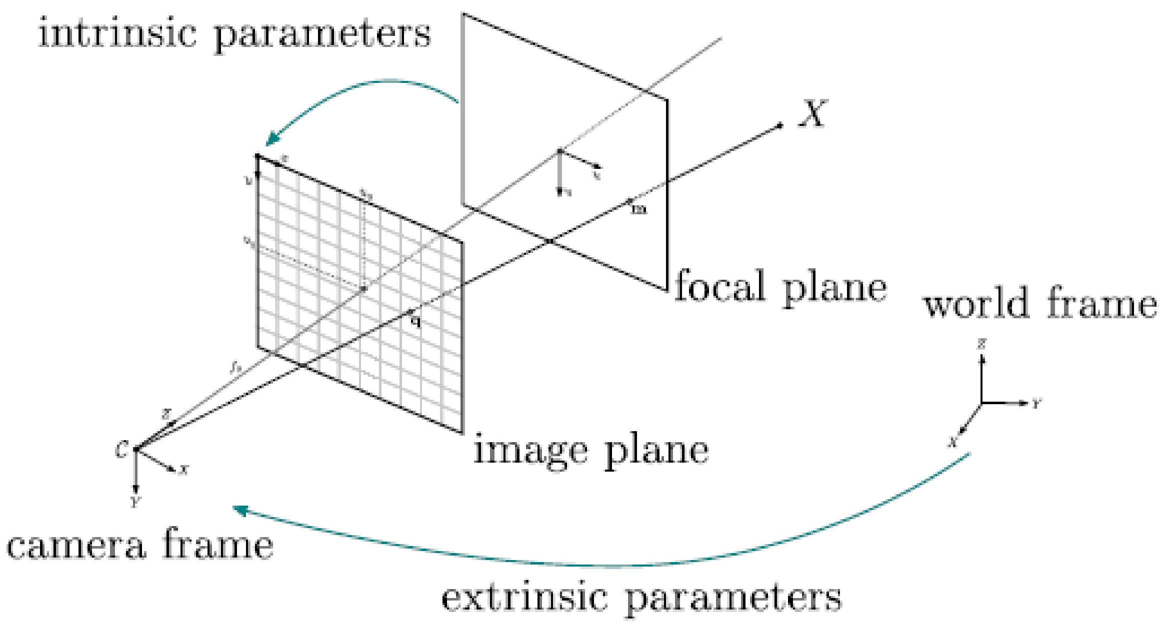
Camera Concepts

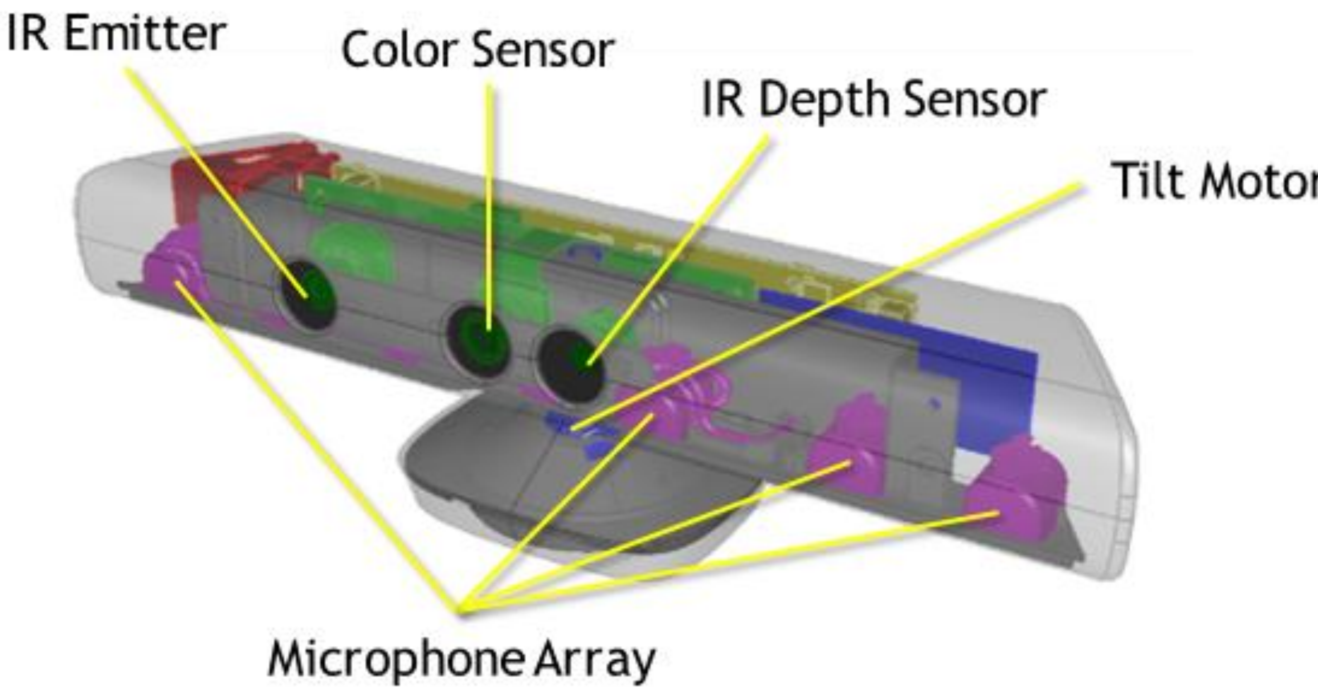
Camera Concepts

- Intrinsic Parameters
  - Focal length
  - Principal point
  - (Skew)
  - (Distortion params.)
- Extrinsic Parameters
  - 6 degrees of freedom (DoF); aka ‘pose’ Rotation + translation



Camera Concepts





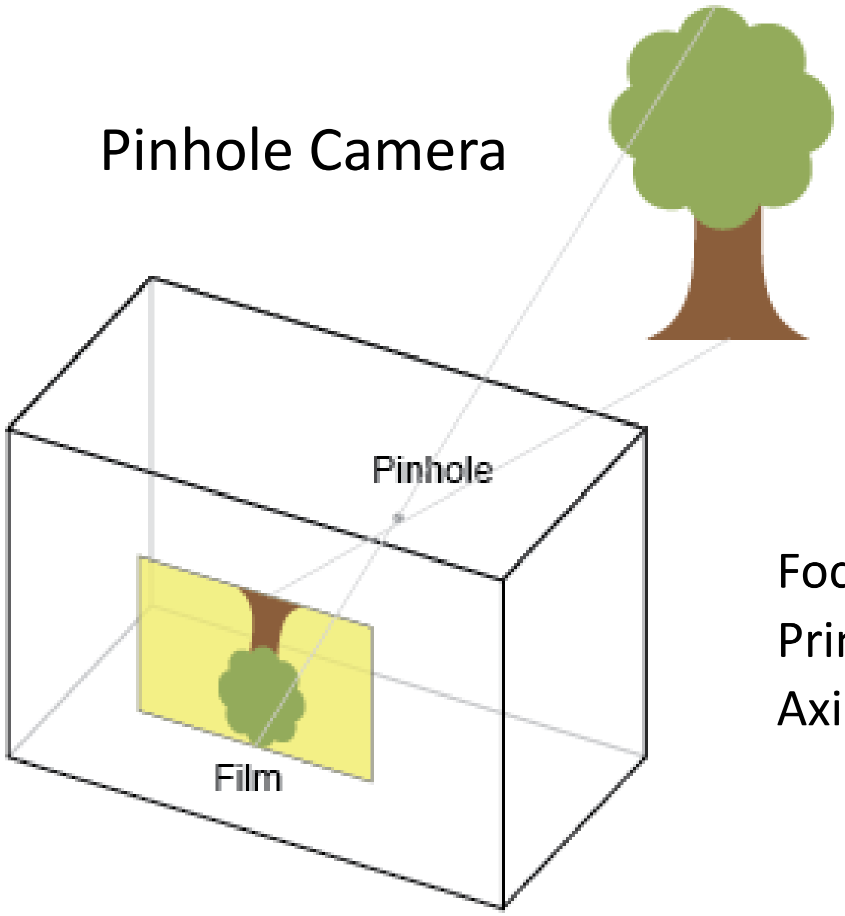
Often we have mapping between depth and color (also extrinsics)

Camera Concepts

- *World space*:
  - global coordinate system
- *Camera space*:
  - coordinate system within current frame
- *Screen / Pixel coordinates:*
  - Pixel position (x, y) + depth value (z)
- *Normalized Device coordinates (NDC):*
  - OpenGL $[-1; 1]^3$
  - DirectX $[-1; 1]^2 \times [0; 1]$
- *Model space:*
  - Local coordinate frame for a given model (within a scene)

Camera Intrinsics

Pinhole Camera



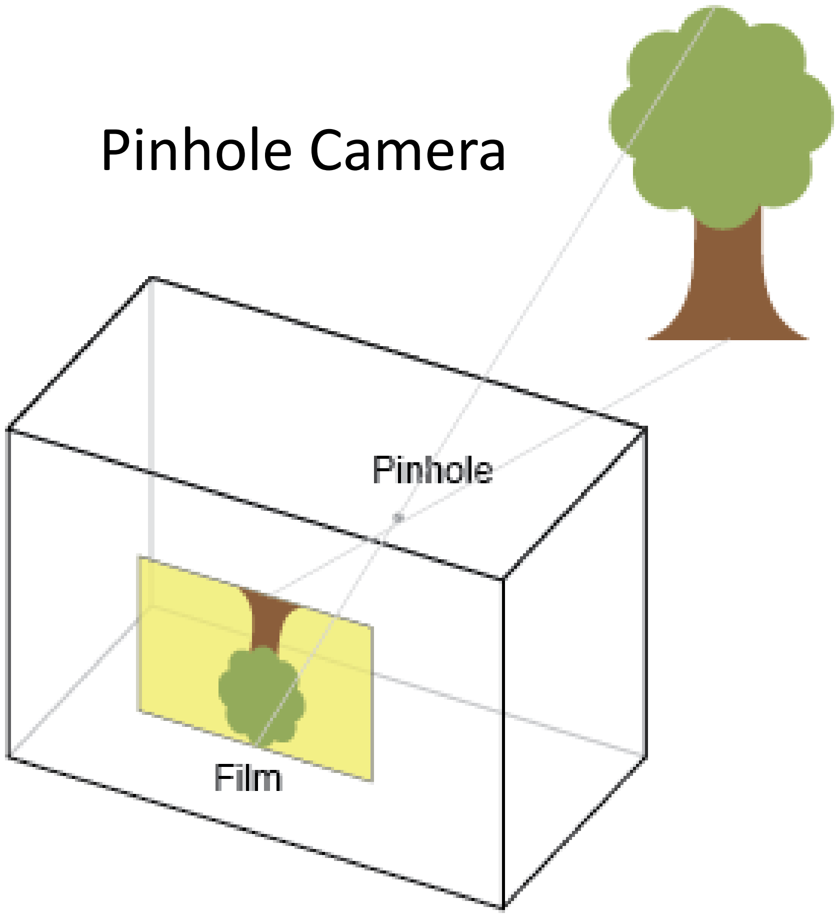
Intrinsic Matrix: $K = \begin{pmatrix} f_x & \gamma & m_x \\ 0 & f_y & m_y \\ 0 & 0 & 1 \end{pmatrix}$

Focal length: f_x and f_y	-> distance between pinhole and image plane
Principal point m_x, m_y	-> proj. center (often width/2, height/2)
Axis skew: γ	-> shear distortion (it's "mostly" zero)

http://ksimek.github.io/2013/08/13/intrinsic/

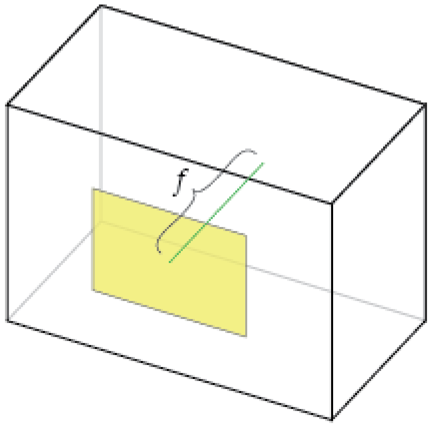
Camera Intrinsics

Pinhole Camera



Intrinsic Matrix: $K = \begin{pmatrix} f_x & \gamma & m_x \\ 0 & f_y & m_y \\ 0 & 0 & 1 \end{pmatrix}$

Focal length f_x, f_y

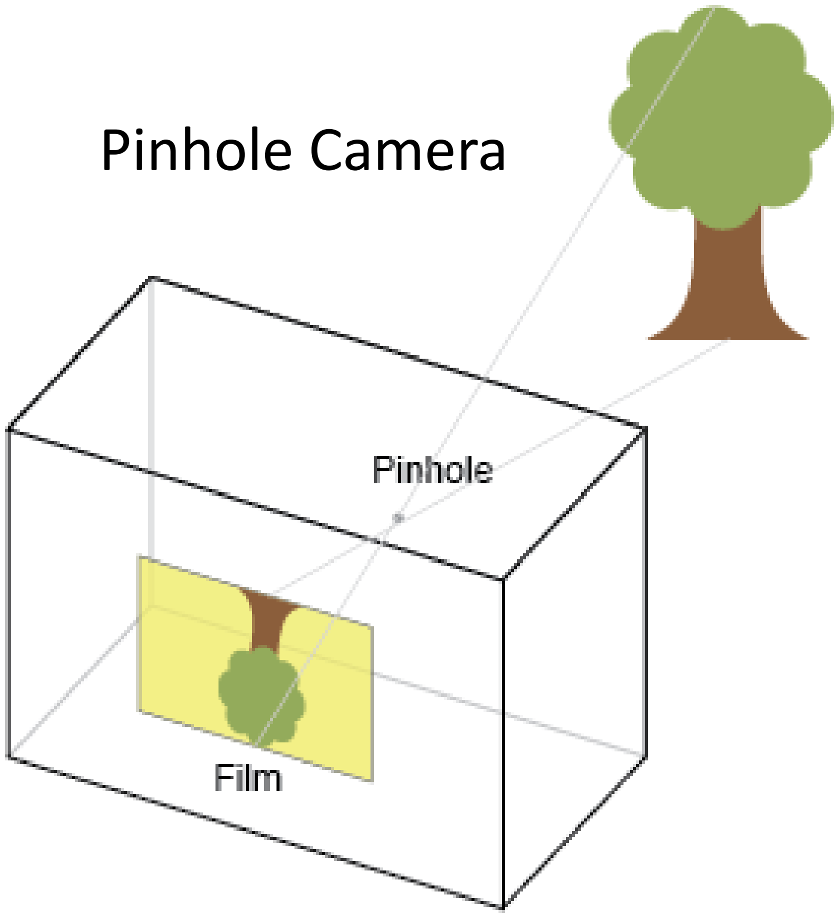


- Focal length is distance between pinhole and film (image plane).
- Focal length is measured in pixels.
- Ratio f_x and f_y defines aspect ratio.

http://ksimek.github.io/2013/08/13/intrinsic/

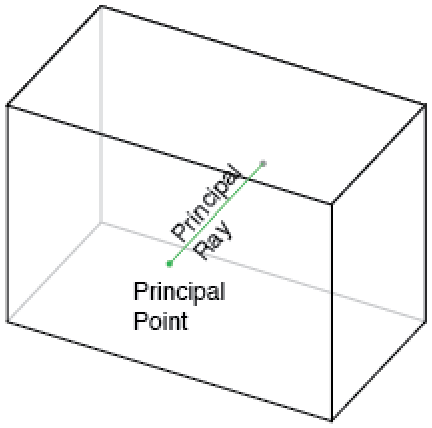
Camera Intrinsics

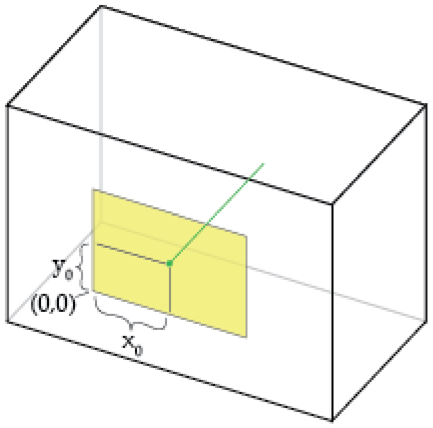
Pinhole Camera



Intrinsic Matrix: $K = \begin{pmatrix} f_x & \gamma & m_x \\ 0 & f_y & m_y \\ 0 & 0 & 1 \end{pmatrix}$

Principal point m_x, m_y (or x_0, y_0); in pixels





http://ksimek.github.io/2013/08/13/intrinsic/

Camera Intrinsics

- Projection:

$$\begin{pmatrix} f_x & \gamma & m_x \\ 0 & f_y & m_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

$$K \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Need division by z_c to obtain pixel coordinates

Camera Intrinsics

- Screen space projection (from camera to screen)

```
// pos.xyz in meters; output in pixels/meters
static inline float3 cameraToScreen(const float3& pos)
{
		return make_float3(
				pos.x*c_depthCameraParams.fx/pos.z + c_depthCameraParams.mx,
				pos.y*c_depthCameraParams.fy/pos.z + c_depthCameraParams.my,
				pos.z);
}
```

Camera Intrinsics

- Inverse Projection (from screen to camera space)

```
// ux, uy in pixels; depth in meters
static inline float3 screenToCamera(uint ux, uint uy, float depth)
{
		const float x = ((float)ux-c_depthCameraParams.mx) / c_depthCameraParams.fx;
		const float y = ((float)uy-c_depthCameraParams.my) / c_depthCameraParams.fy;
		return make_float3(depth*x, depth*y, depth);
}
```

Camera Extrinsics

- 6 Degrees of Freedom (DoF)
  - 3 for rotation
  - 3 for translation

$$[R,T] = \begin{pmatrix} R_{00} & R_{01} & R_{02} & t_x \\ R_{10} & R_{11} & R_{12} & t_y \\ R_{20} & R_{21} & R_{22} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- 4x4 or 3x4 matrix

http://ksimek.github.io/2012/08/22/extrinsic/

Camera Extrinsics

$$R = R_z(\gamma) \cdot R_y(\beta) \cdot R_x(\alpha)$$

$$R^T = R^{-1}$$

$$|\det R| = 1$$

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

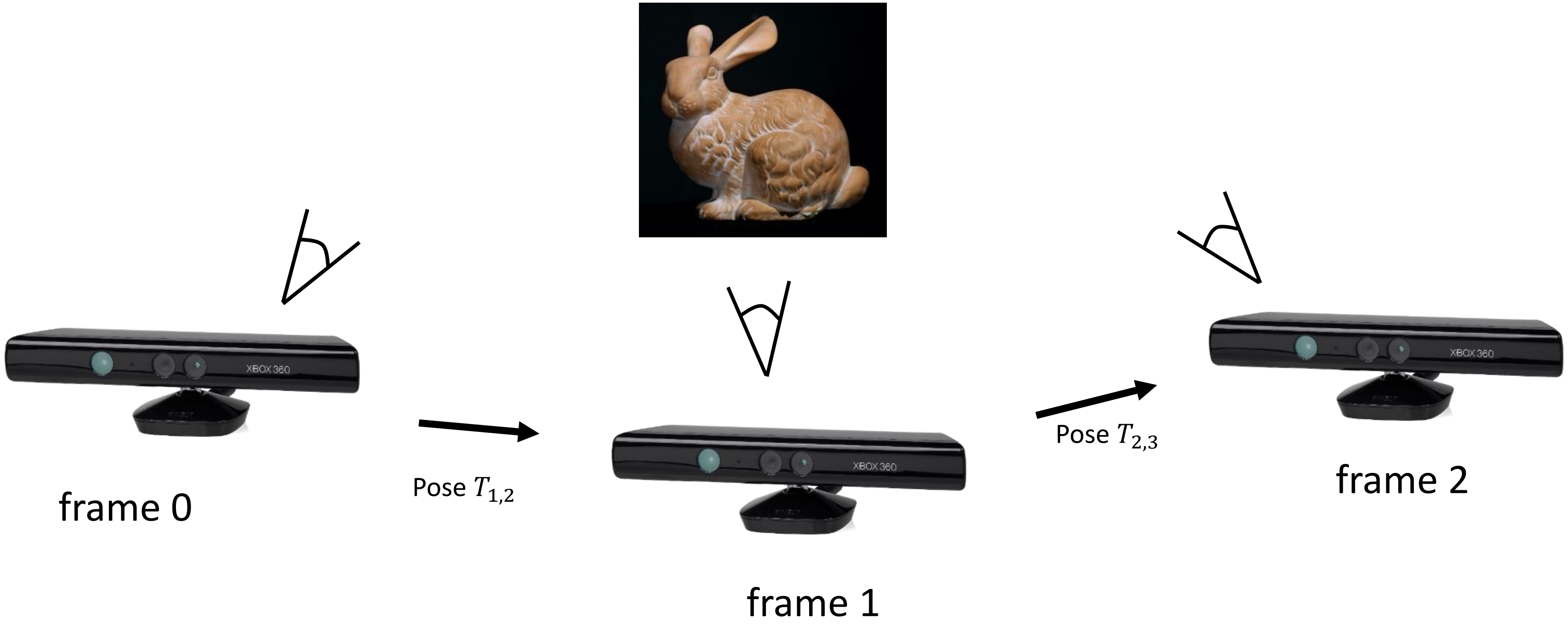
$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

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

http://ksimek.github.io/2012/08/22/extrinsic/

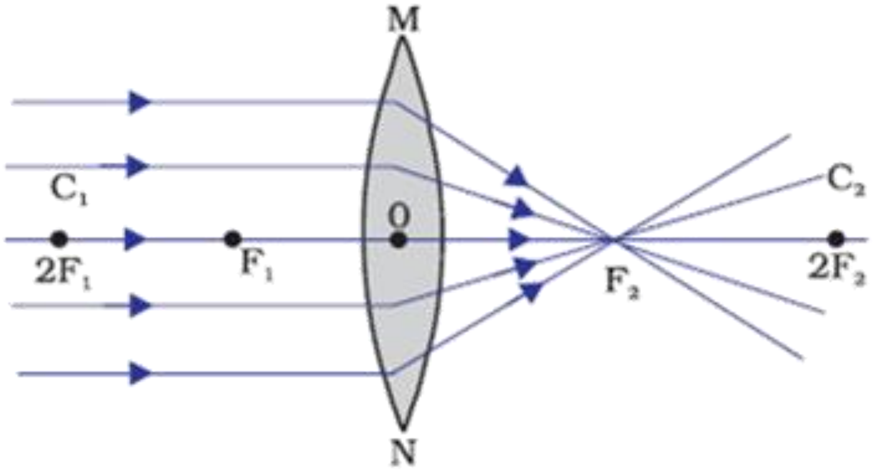
Camera Extrinsics

- It's all relative
  - Does the object move or the camera?



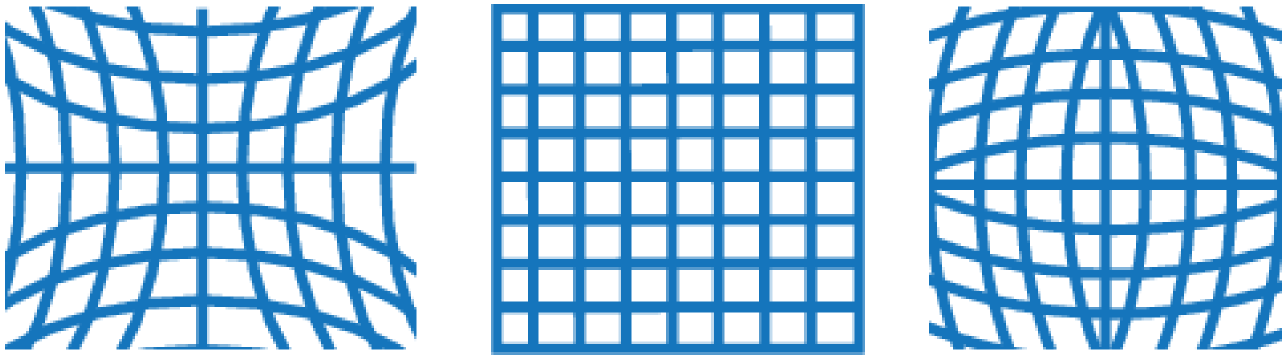
Camera Calibration

- Pinhole Camera vs Real Camera



Camera Calibration

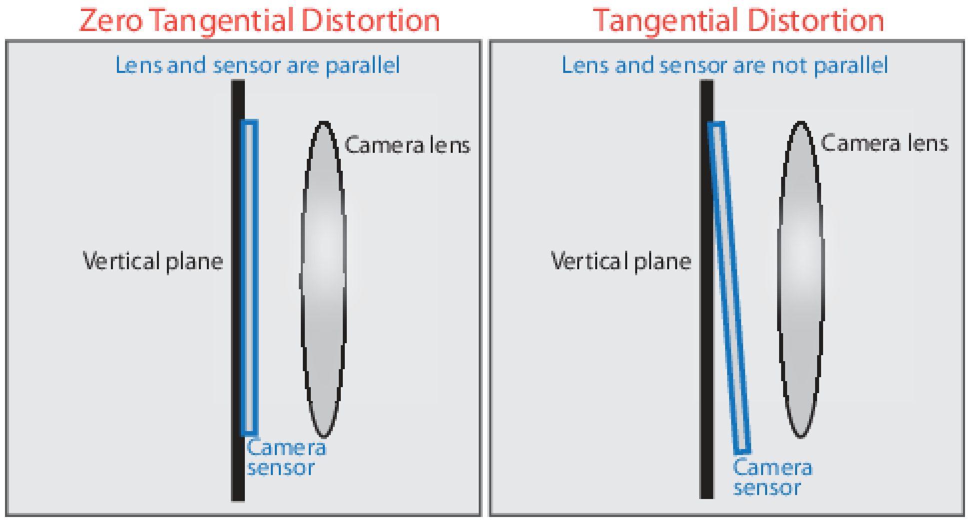
- Pinhole Camera vs Real Camera
  - Many approximations
- Radial distortion and tangential distortion models



Negative radial distortion "pincushion"

No distortion

Positive radial distortion "barrel"



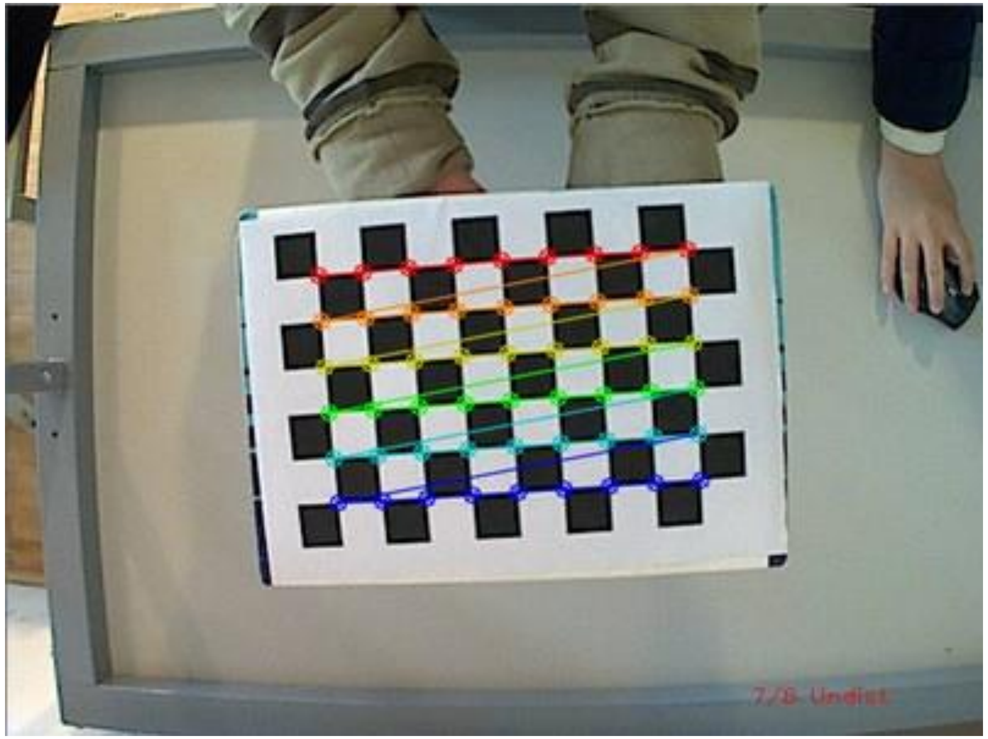
https://de.mathworks.com/help/vision/ug/camera-calibration.html

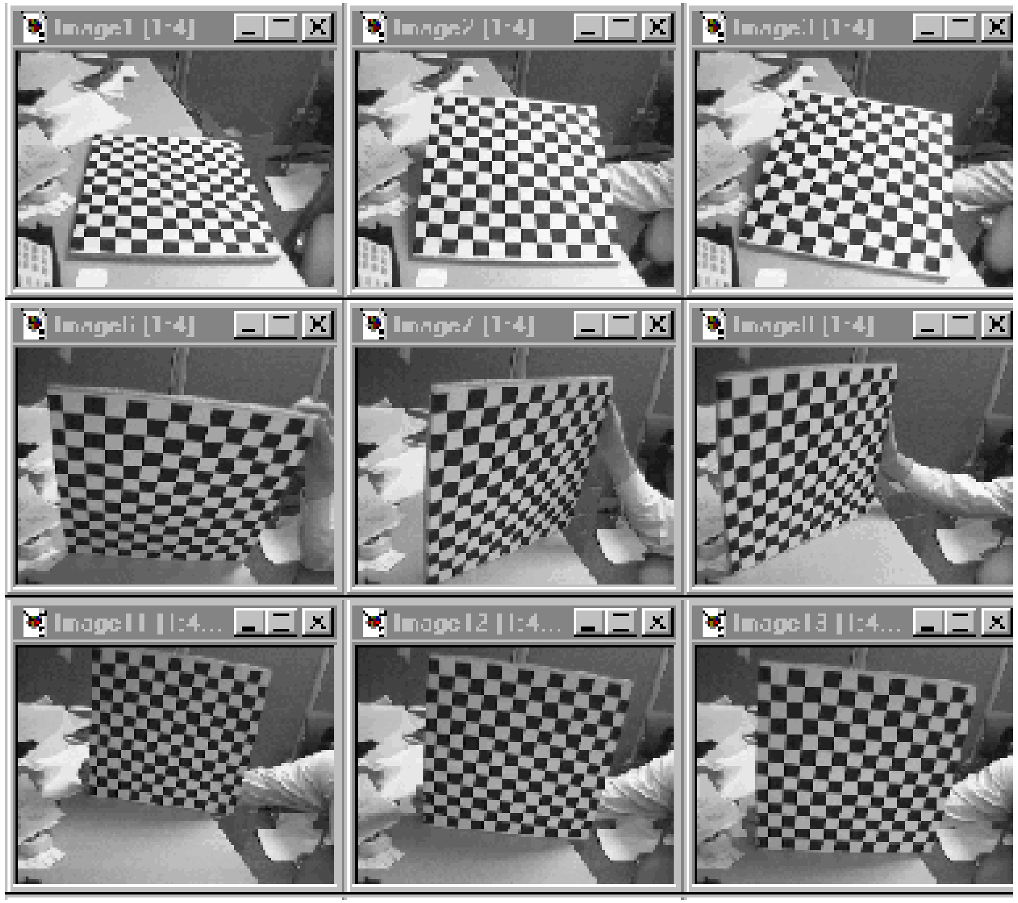
Camera Calibration

- Distortion coefficients k_1, k_2, p_1, p_2, k_3
- For radial distortion:
  - Occurs due to “barrel” / “fish-eye” effect
  - $x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$
  - $y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$
- For tangential distortion:
  - Occurs if image plane and lens are not parallel
  - $x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$
  - $y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy)]$

https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

Camera Calibration





https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

Example: 3D Reconstruction

3D Scanner

Intrinsics to obtain point cloud
for current frame

[S. Narasimhan]

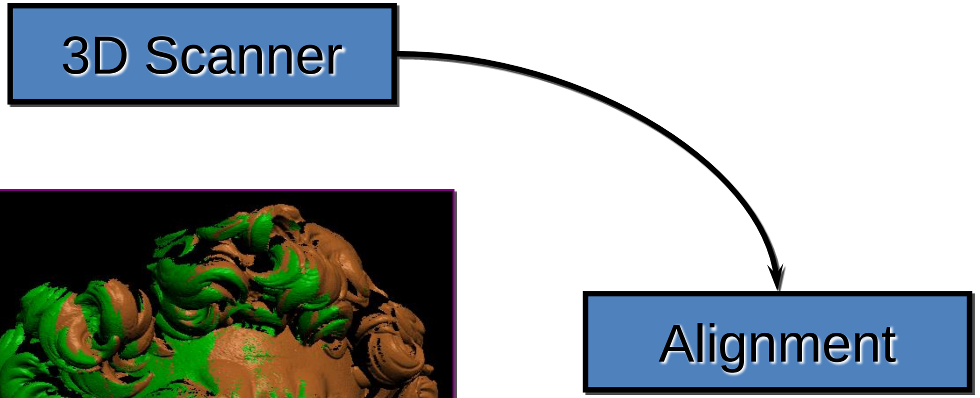
Example: 3D Reconstruction

3D Scanner

Move camera (or object) to obtain next frame (i.e., next view)

[S. Narasimhan]

Example: 3D Reconstruction

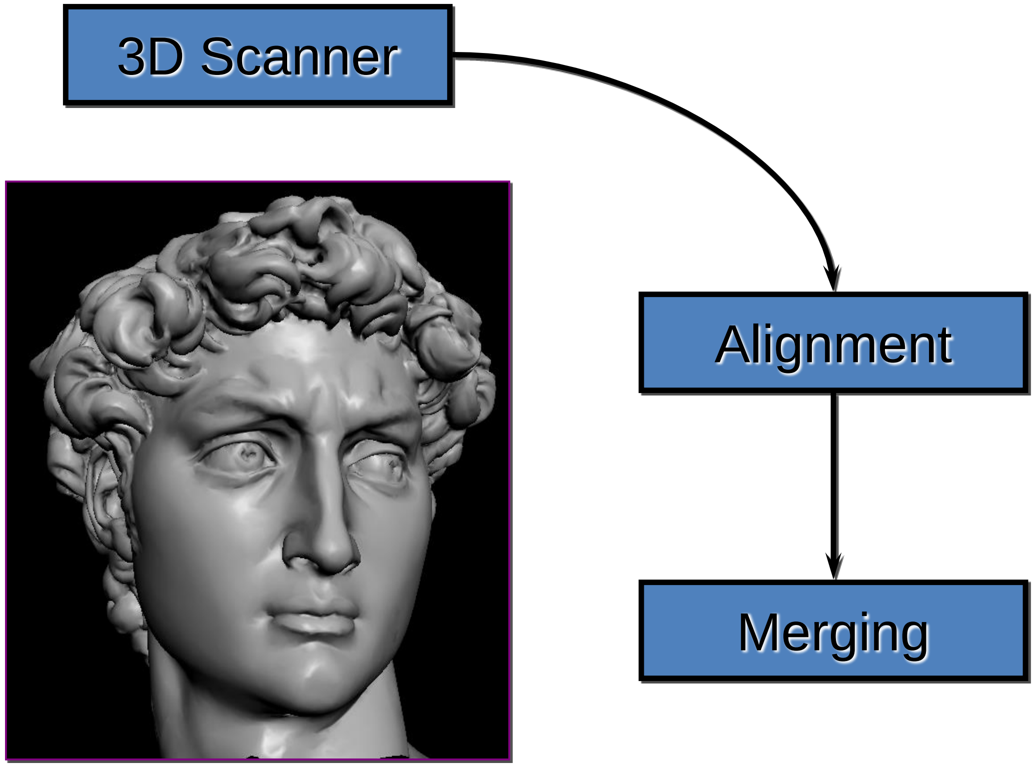


Compute alignment between the capture frames; i.e., compute the extrinsics (6DoF)

[S. Narasimhan]

Example: 3D Reconstruction

Merge point clouds +
surface reconstruction



[S. Narasimhan]

Example: 3D Reconstruction

Can we do with many views

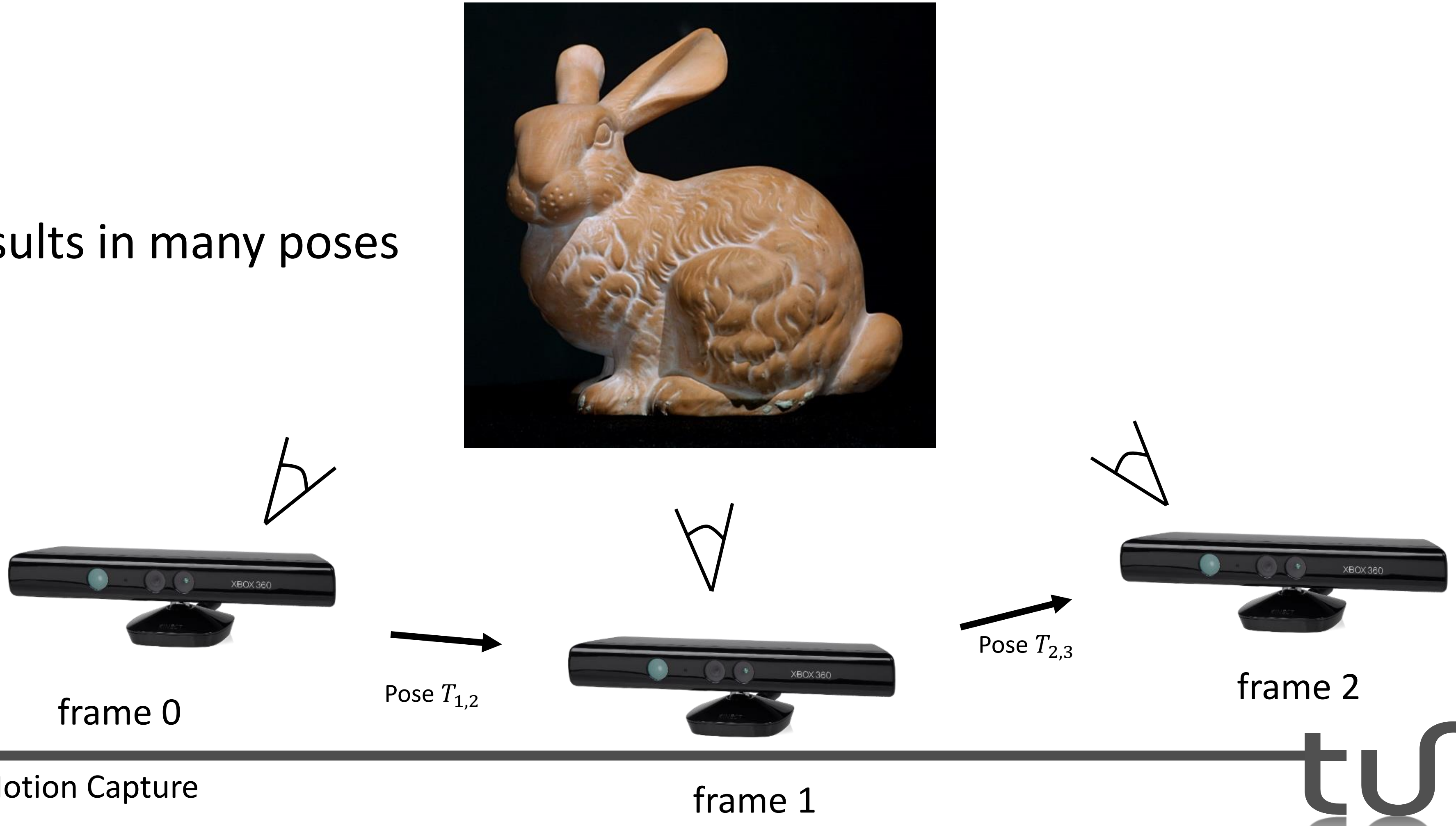
frame 0

frame 1

frame 2

Example: 3D Reconstruction

Which results in many poses



Administrative

- First Tutorial:
  – This week ☺
  – Release of first assignment
  – Introduction of first assignment
  – Groups of two during exercises
  – Check out Moodle

Administrative

See you next week!