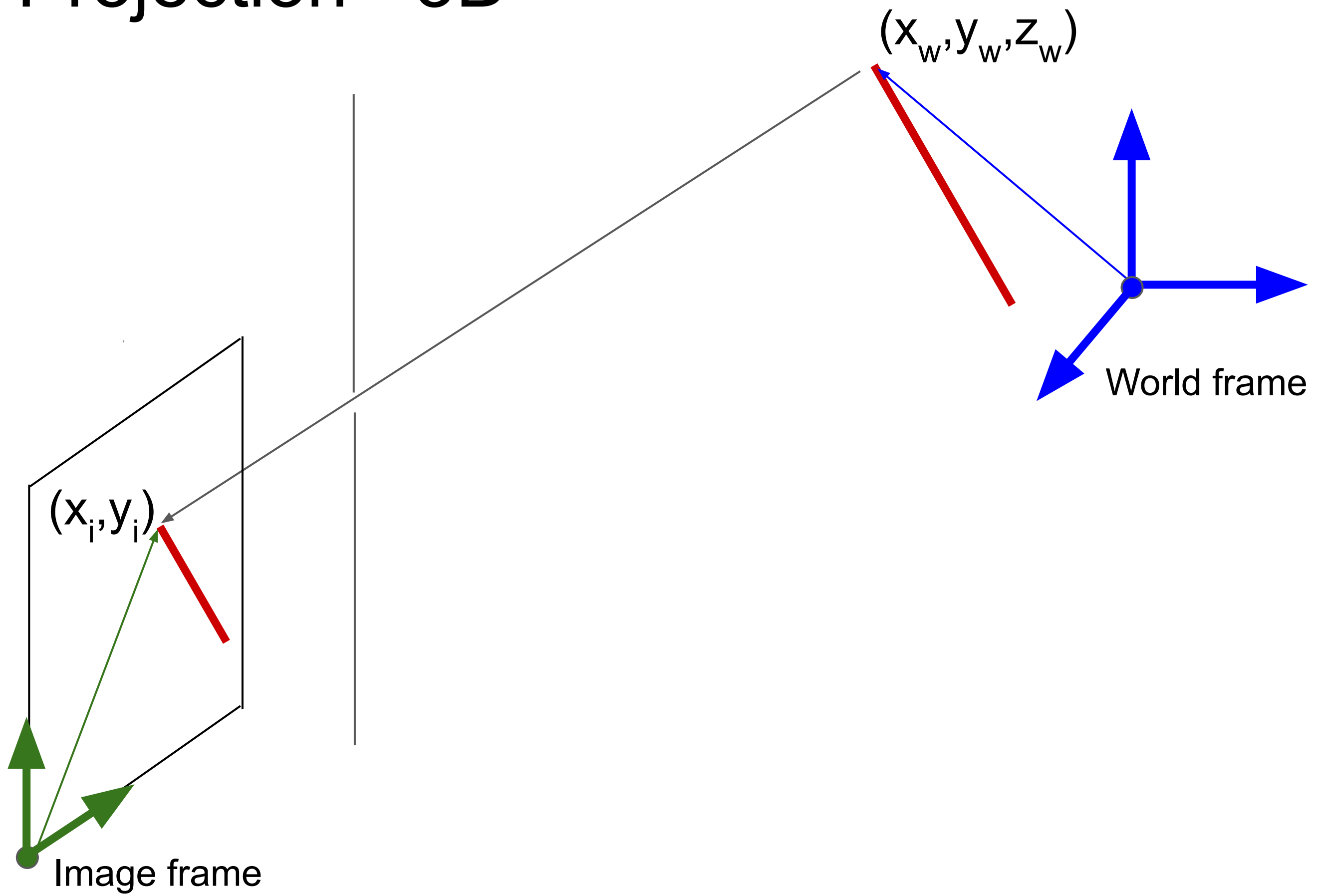
MECS 6616

3D Computer Vision

Spring 2020
Matei Ciocarlie

Perspective Projection - 3D



Camera Matrix

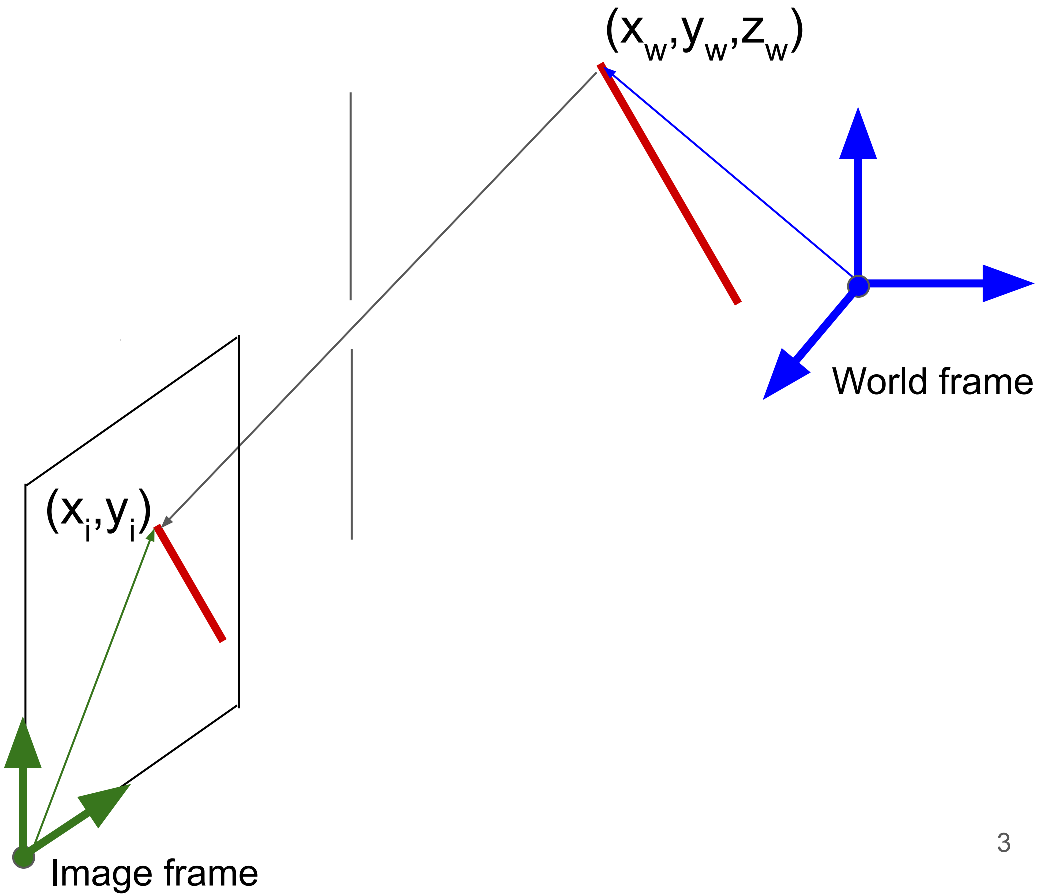
World point $\mathbf{x}_w = [x_w, y_w, z_w, 1]^T$

Image point $\mathbf{x}_c = [x_i, y_i, 1]^T$

Camera matrix $\mathbf{P} \subset \Re^{3x4}$:

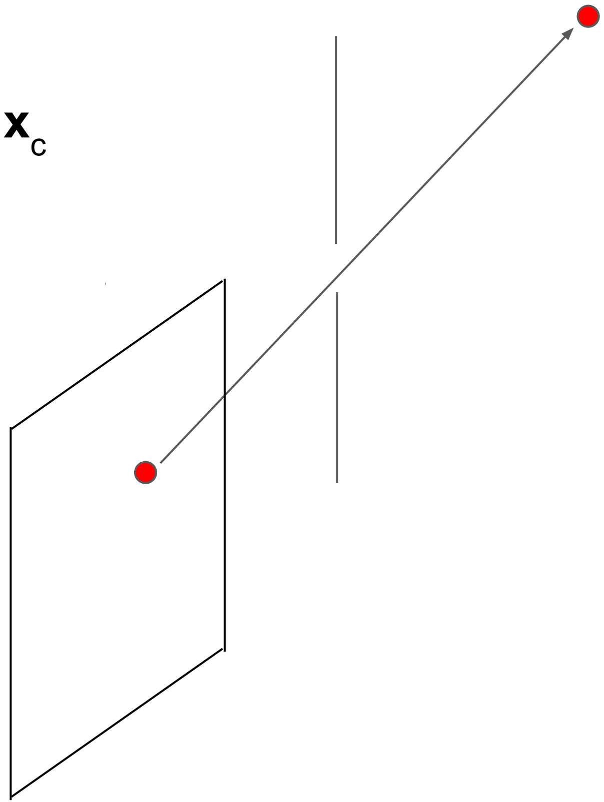
$$\mathbf{x}_c = \mathbf{P}\ \mathbf{x}_w$$

… for any point in scene.



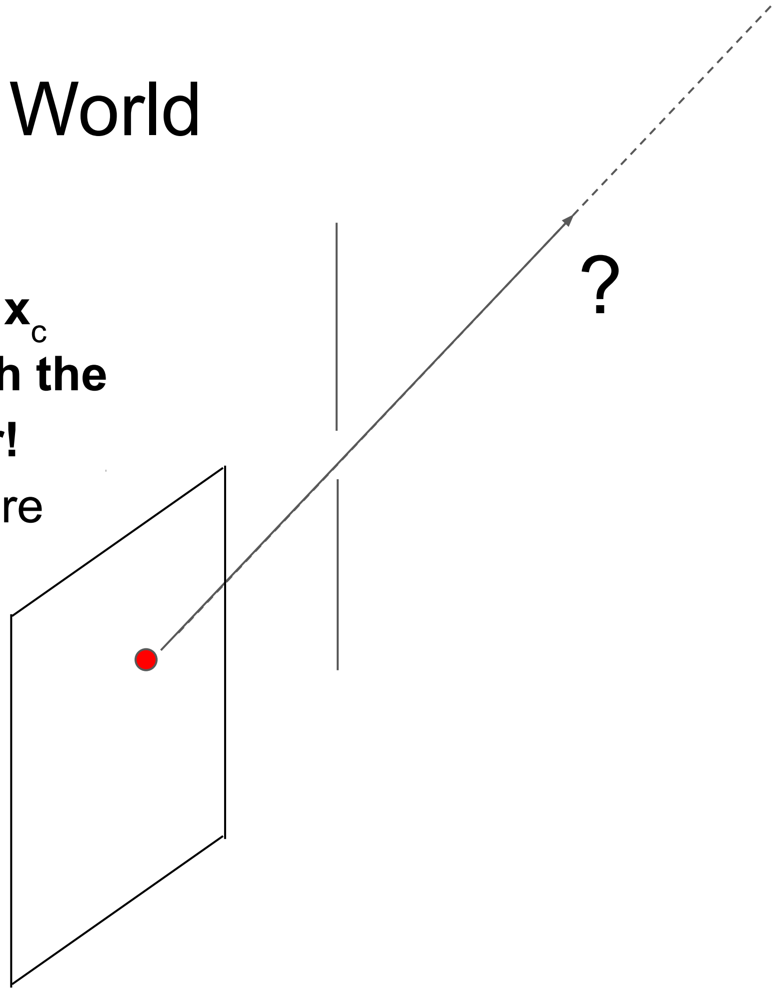
Projecting Back into the World

- Into the image: $\mathbf{x}_c = \mathbf{P}\,\mathbf{x}_w$
- Into the world: compute $\mathbf{x}_w$ given $\mathbf{x}_c$



Projecting Back into the World

- Into the image: $\mathbf{x}_c = \mathbf{P}\,\mathbf{x}_w$
- Into the world: compute $\mathbf{x}_w$ given $\mathbf{x}_c$
    - **can not be fully solved with the information we have so far!**
    - we just know $\mathbf{x}_w$ is somewhere along **a ray**



Key Problem in Vision for Robotics:

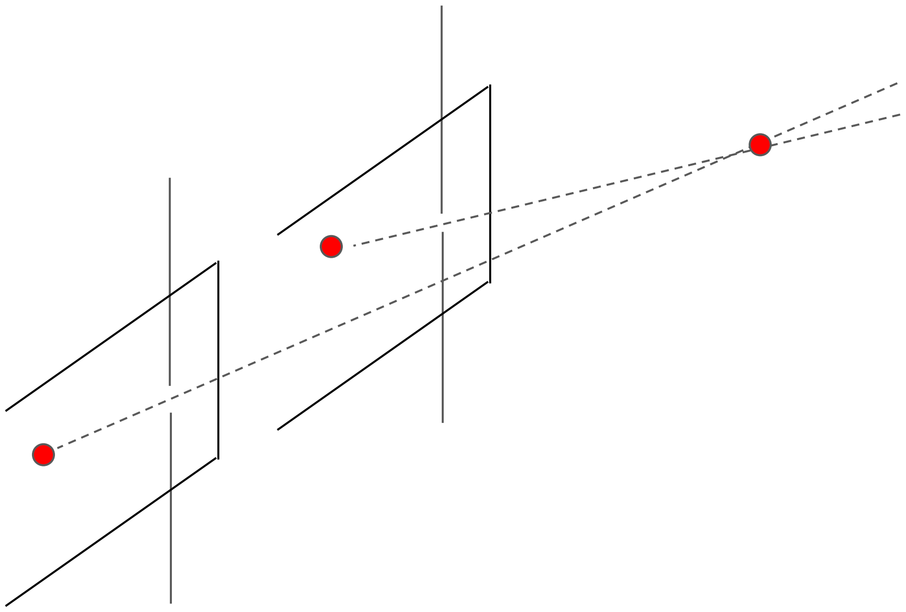
Computing the 3D geometry of a scene from one or multiple 2D images.

Question: how do people do it?

- We can tell 3D scene geometry from a single image!

Stereo Vision

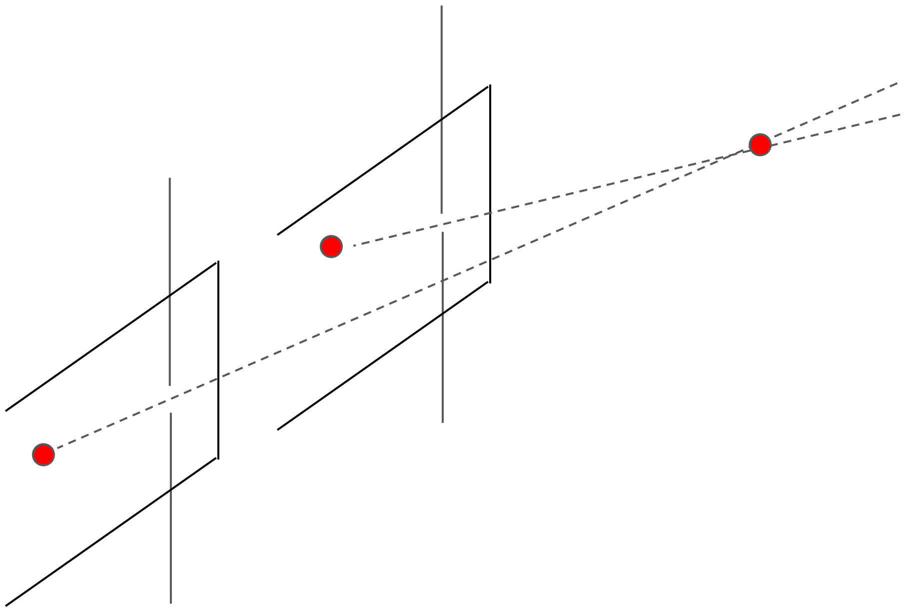
What if we have two cameras?



Stereo Vision

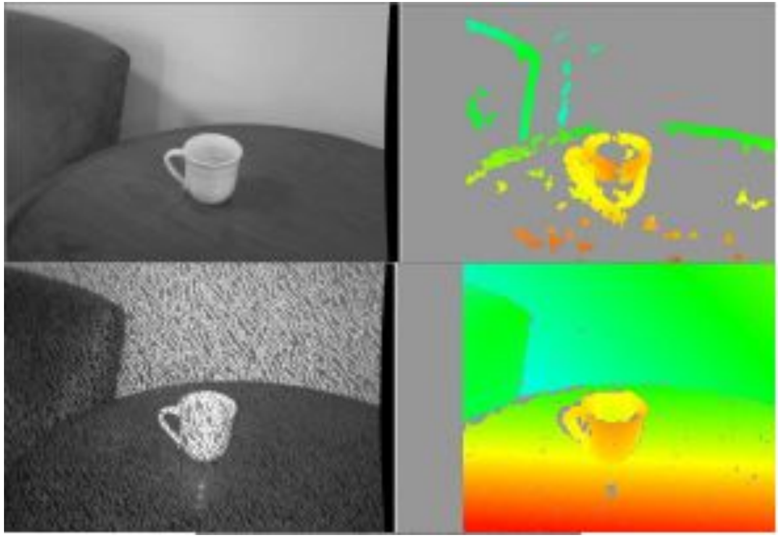
What if we have two cameras?

- we can determine 3D scene structure via **triangulation**
- calibration problem: must know **relative position** of the two cameras
- must solve a **stereo correspondence problem**: identify the same scene point in both images



Stereo Vision

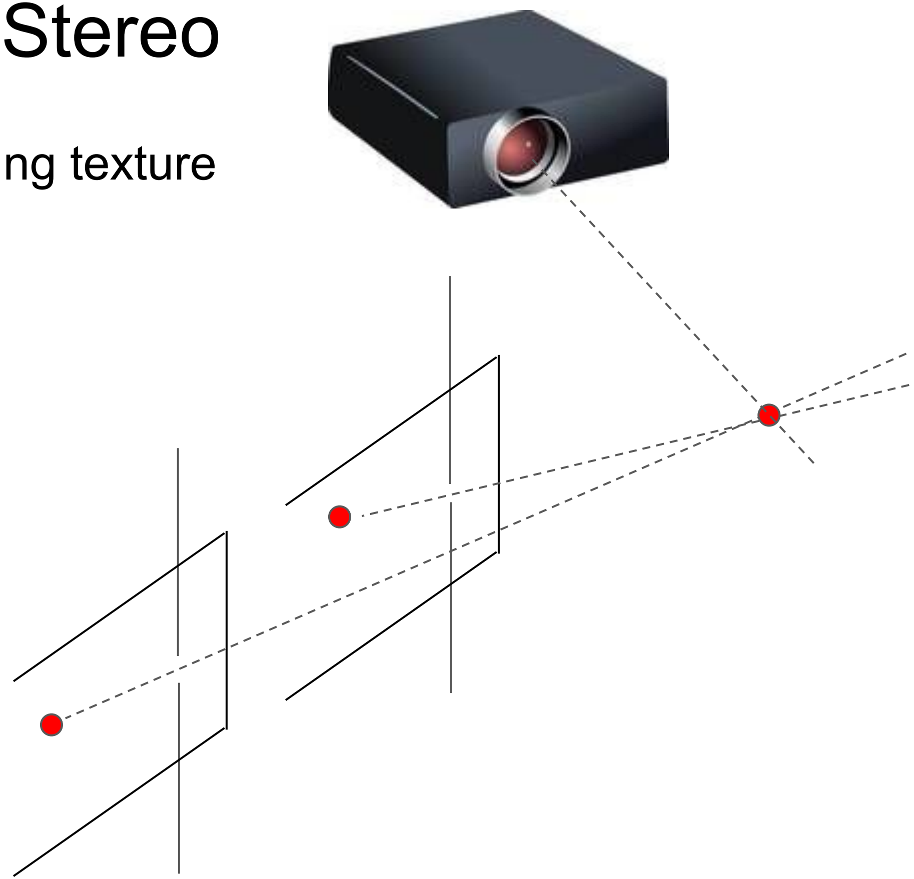
- Stereo only works where the scene has **interesting texture**!
- Can we add texture to the scene?
  - sure, with a projector



[Konolige, “Projected Texture Stereo”]

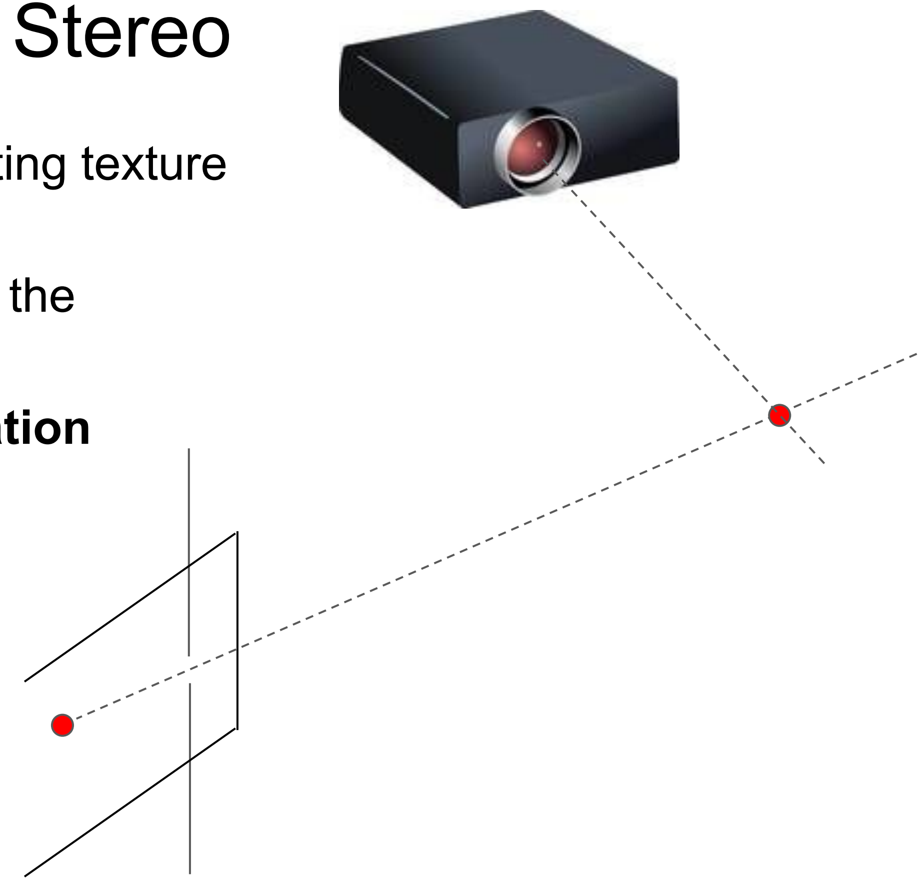
Camera - Projector Stereo

- Projector can add interesting texture to the scene



Camera - Projector Stereo

- Projector can add interesting texture to the scene
- … or even replace one of the cameras.
- Same principle: **triangulation**



Primesense Sensor

- Created structured light sensor
- Licensed by Microsoft for Kinect
- Also used in other similar sensors
- Acquired by Apple in 2013 for $350M

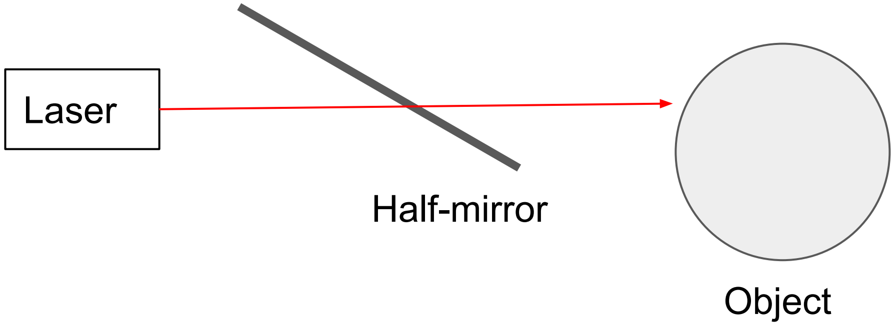
Primesense Sensor

- "Blind spots":
  - **black** objects that absorb IR light
  - **shiny** objects that bounce IR light off
  - **transparent** objects that let IR light through
  - **sunlight** drowns out IR pattern

[Mahler, Patil, Kehoe, van den Berg, Ciocarlie, Abbeel and Goldberg, ICRA 2015]

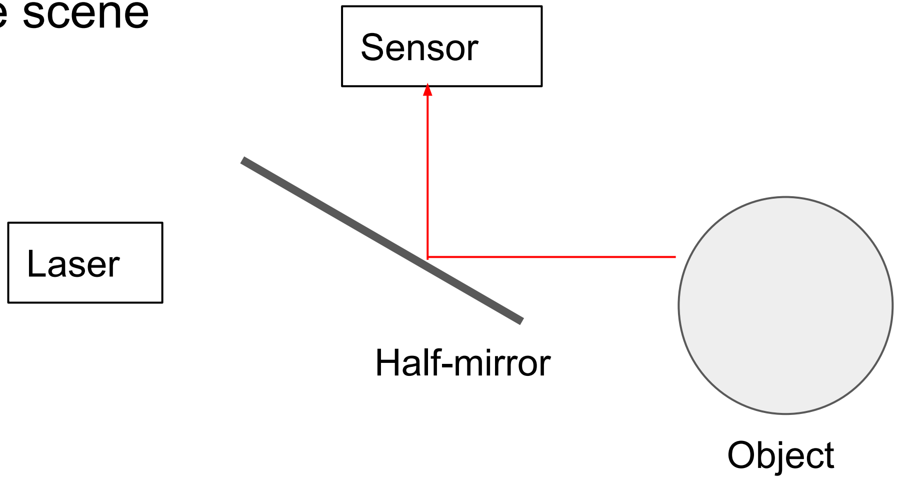
Time-of-flight Lasers

- Send a pulse of light into the world
- Measure time until light hits something and comes back
  - depends on distance to object
- Move entire assembly to sweep entire scene



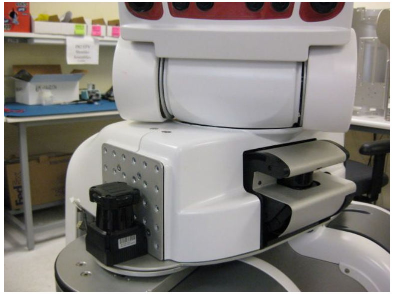
Time-of-flight Lasers

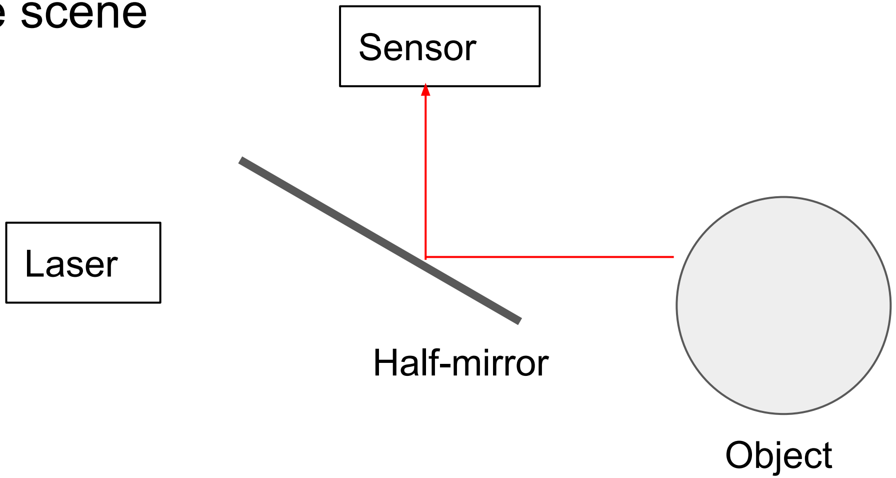
- Send a pulse of light into the world
- Measure time until light hits something and comes back
    - depends on distance to object
- Move entire assembly to sweep entire scene



Time-of-flight Lasers

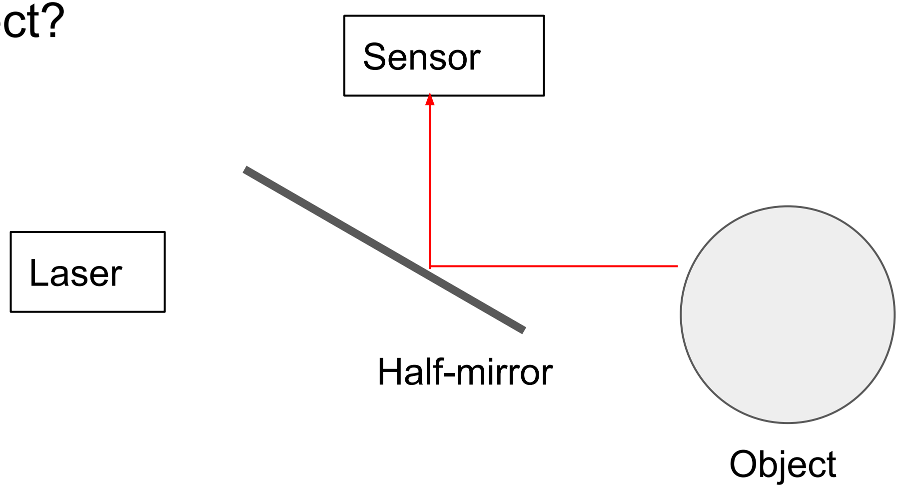
- Send a pulse of light into the world
- Measure time until light hits something and comes back
  - depends on distance to object
- Move entire assembly to sweep entire scene
- Examples
  - line lasers (e.g. Sick, Hokuyo)
  - 3D sweep sensors (e.g. Leica)





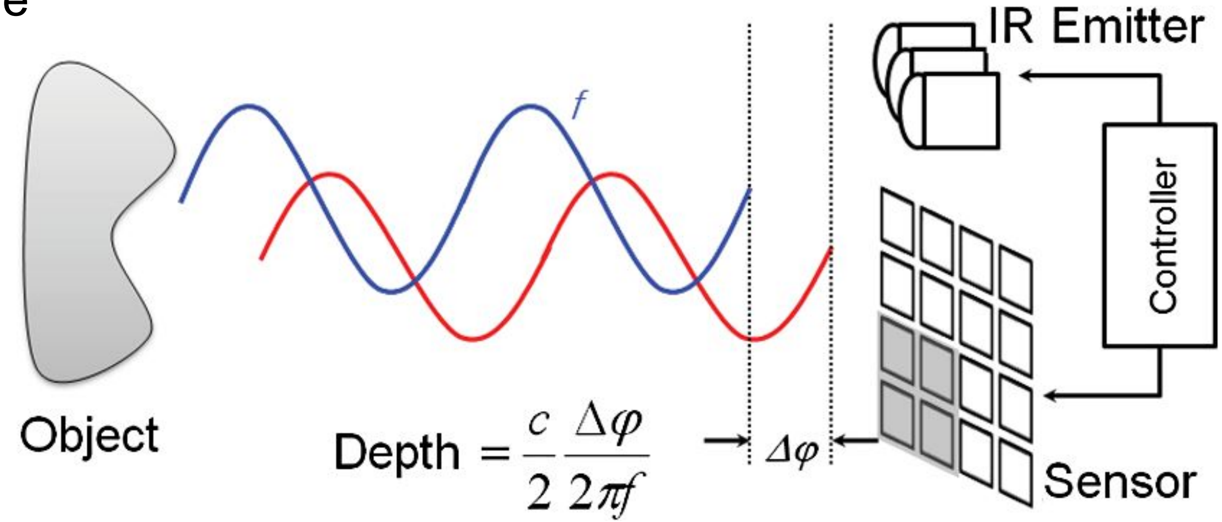
Time-of-flight Lasers

- Problems:
  - need very precise time measurement for each reading
  - sweeping a scene takes time
  - what if laser bounces off the object?
  - ambient light / eye safety



Phaseshift Sensors

- Project modulated light into the scene
- Measure phase shift between emission and response
- Multiple “pixels” possible



[Shim and Lee, Performance evaluation of time-of-flight and structured light depth sensors in radiometric/geometric variations, Opt. Eng. 51(9)]

Phaseshift Sensors

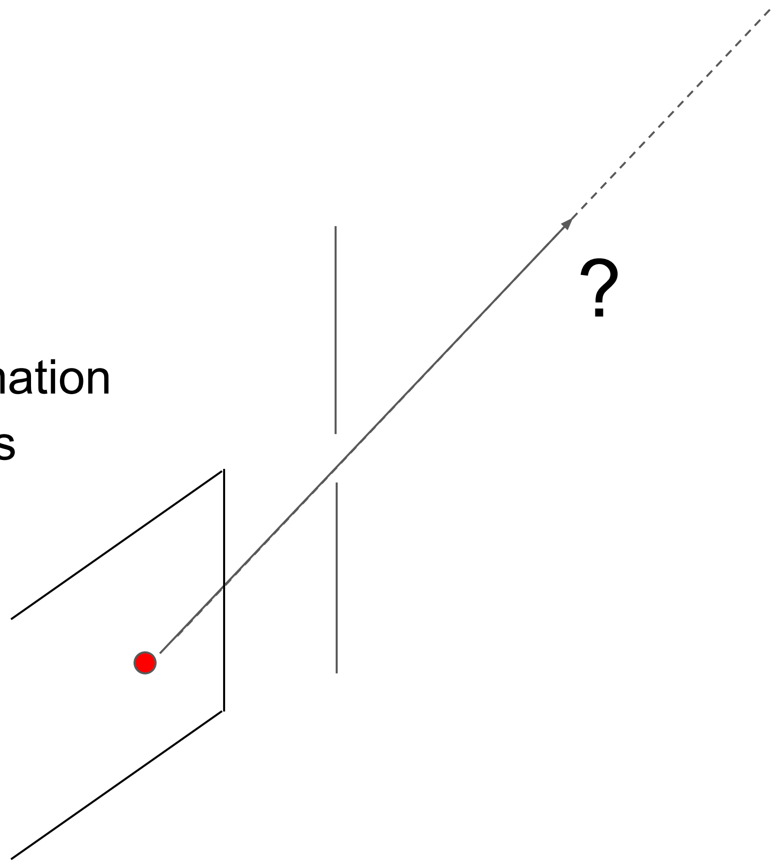
- Project modulated light into the scene
- Measure phase shift between emission and response
- Multiple “pixels” possible
- Examples
  - Swiss Ranger
  - Canesta (acquired by Microsoft in 2010)
  - Microsoft Kinect One (rumored)

Phaseshift Sensors

- Problems
  - indeterminate beyond one wavelength
    - need to know working range
    - perhaps combine with time-of-flight?
  - projected light absorption, reflection
  - ambient light

3D Vision

- Images project a 3D world into a 2D reading
  - “depth” information is missing
- We can use images to recreate the 3D information
- Sensors that provide such info are sometimes referred to as “depth cameras”



Common 3D Scene Representation: Point Clouds

- A list of known “points” in the scene:
  - ...
  - $\mathbf{p_i} = [x_i, y_i, z_i], [r_i, g_i, b_i], \ldots$
  - ...

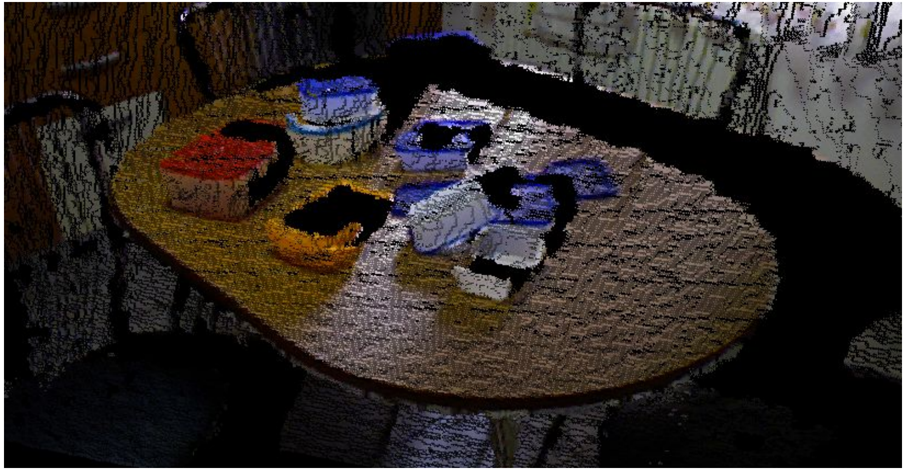


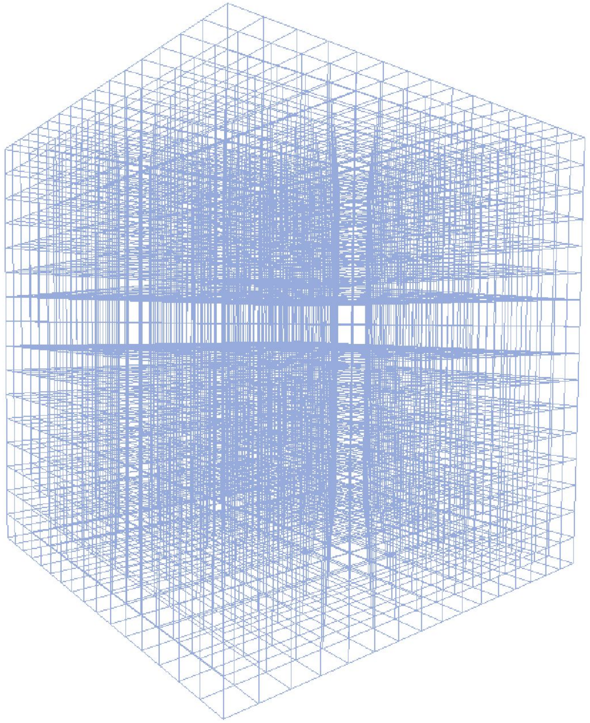
image: [www.pointclouds.org]

3D Scene Representation

Dense grids: very large in 3D

128	144	120	130	128	165	184
131	152	128	114	180	200	230
122	101	111	190	205	240	255
100	40	42	20	34	50	98
22	30	25	22	72	90	108

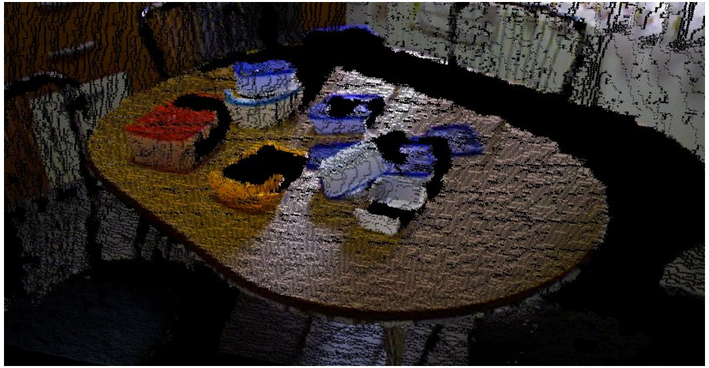
2D dense grid



3D dense grid

Learning on 3D Vision

- Unlike an image, a point cloud is
  - in arbitrary order
  - of varying size



$$\mathbf{p_1} = [x_1, y_1, z_1]$$

$$\mathbf{p_2} = [x_2, y_2, z_2]$$

$$\ldots\ldots$$

$$\mathbf{p_n} = [x_n, y_n, z_n]$$

Learning on 3D Vision

- Finding planar surfaces
  - Common problem due to prevalence of flat surfaces in human settings
  - What does it remind us of?

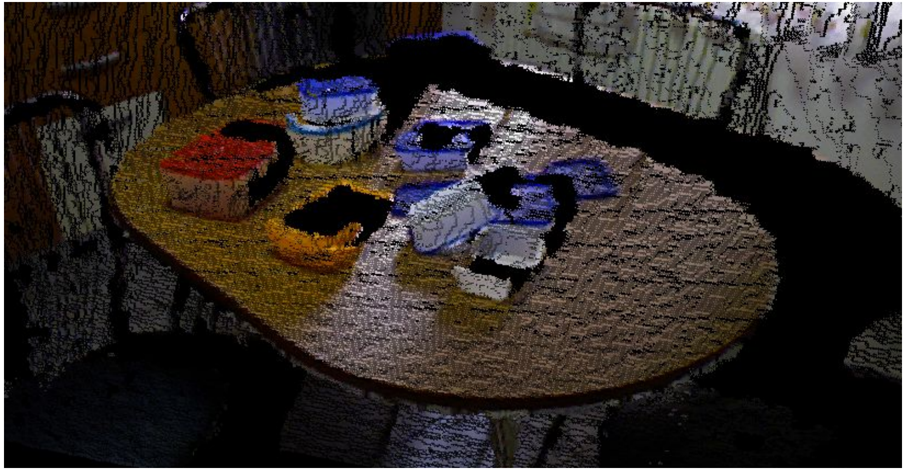
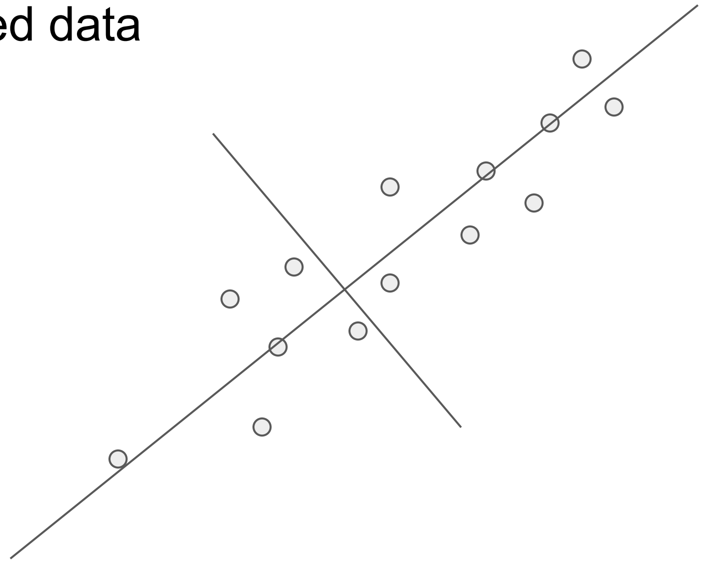


image: [www.pointclouds.org]

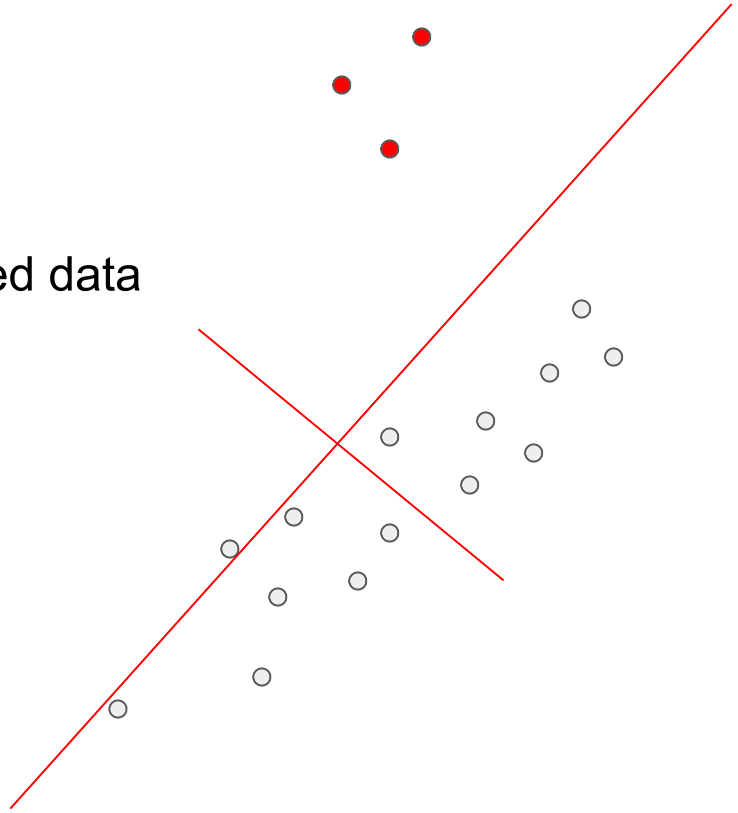
Plane (line) fitting

- Least Squares Fit
  - same as PCA on normalized data



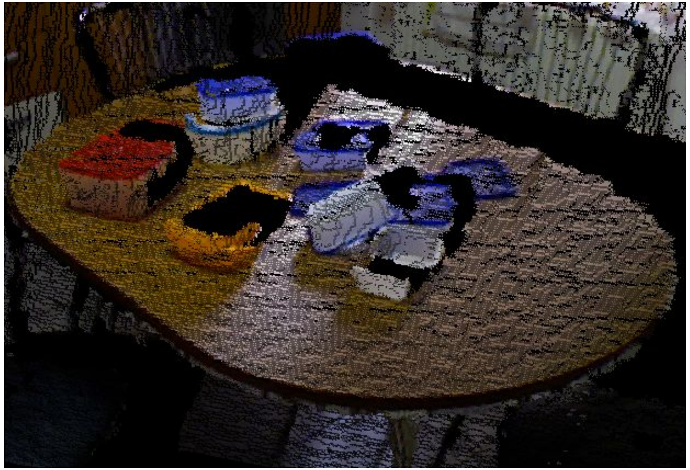
Plane (line) fitting

- Least Squares Fit
  - same as PCA on normalized data
  - but… sensitive to outliers!



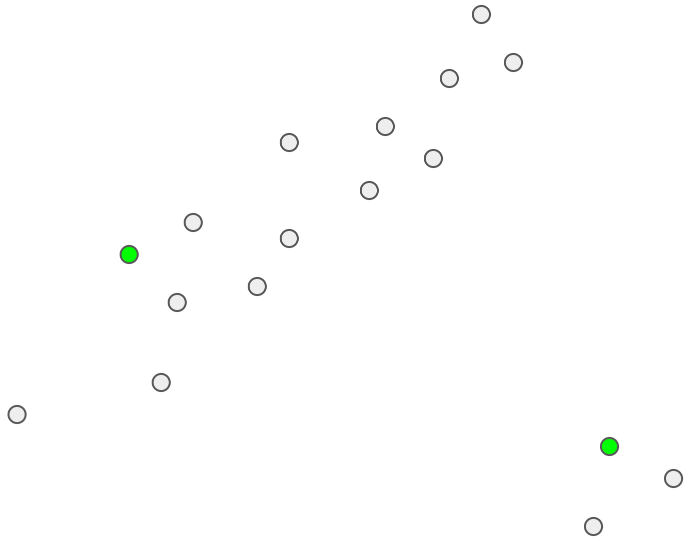
Plane (line) fitting

- Least Squares Fit
  - same as PCA on normalized data
  - but… sensitive to outliers!
- A linear subspace could:
  - fit "reasonably" for all my data (PCA)
  - fit "very well" for a subset of my data (???)
    - but which subset?



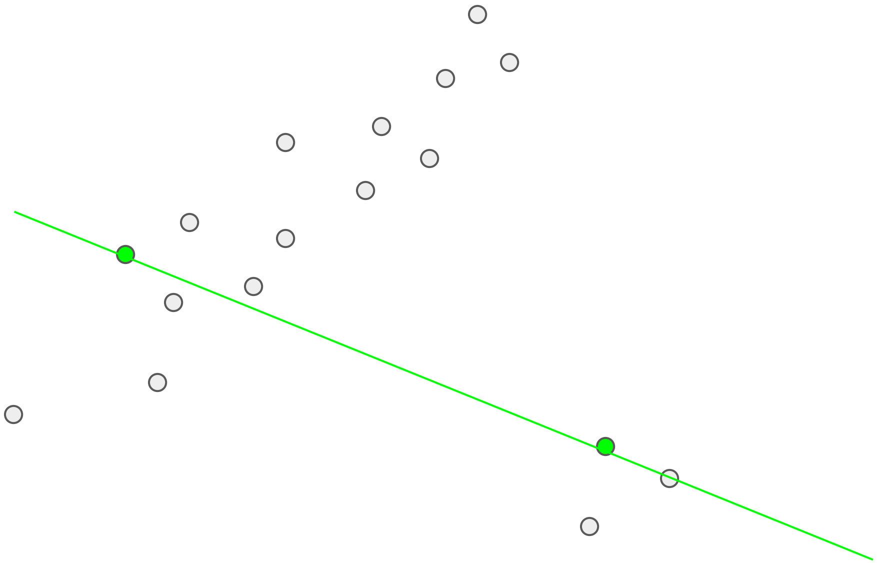
RANSAC - RANdom SAmple Consensus

- Repeat:
  - randomly pick **subset** of points



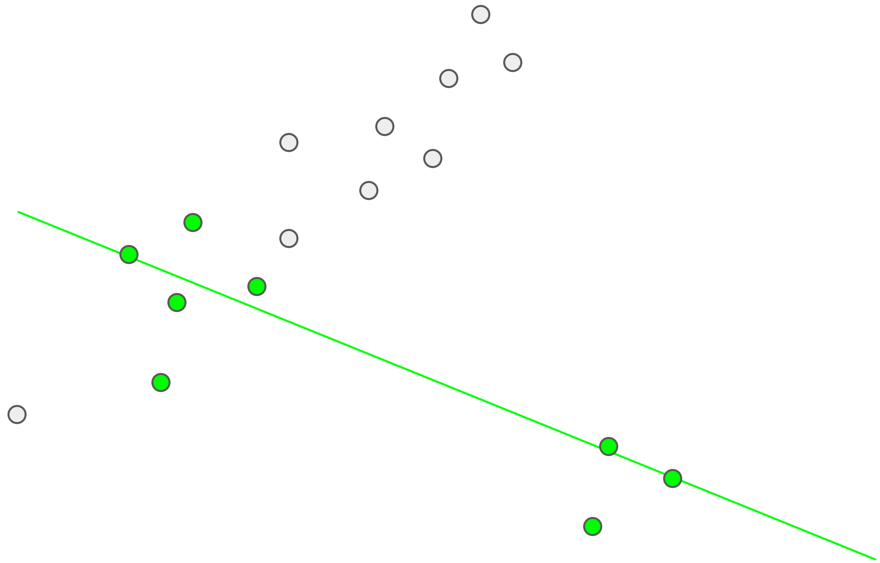
RANSAC - RANdom SAmple Consensus

- Repeat:
  - randomly pick **subset** of points
  - generate **hypothesis**



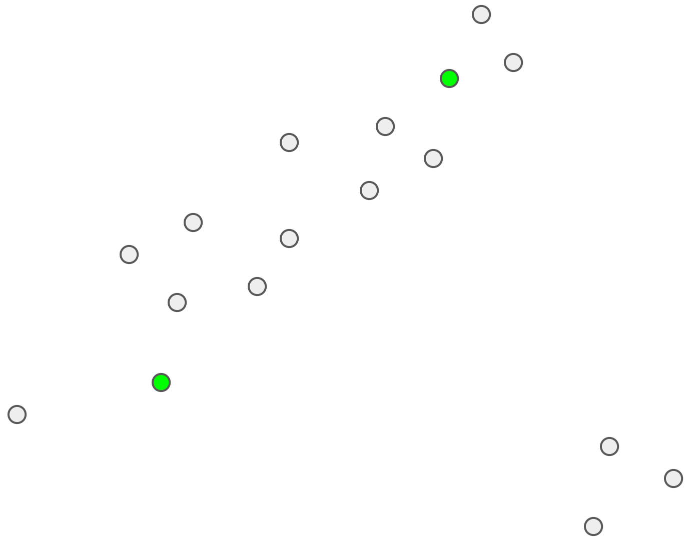
RANSAC - RANdom SAmple Consensus

- Repeat:
  - randomly pick **subset** of points
  - generate **hypothesis**
  - count **inliers**: 8



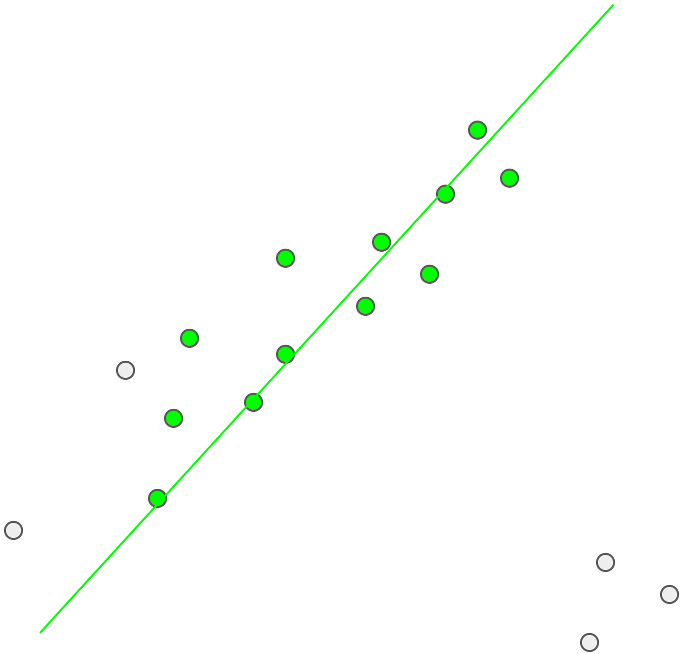
RANSAC - RANdom SAmple Consensus

- Repeat:
  - randomly pick **subset** of points
  - generate **hypothesis**
  - count **inliers**:



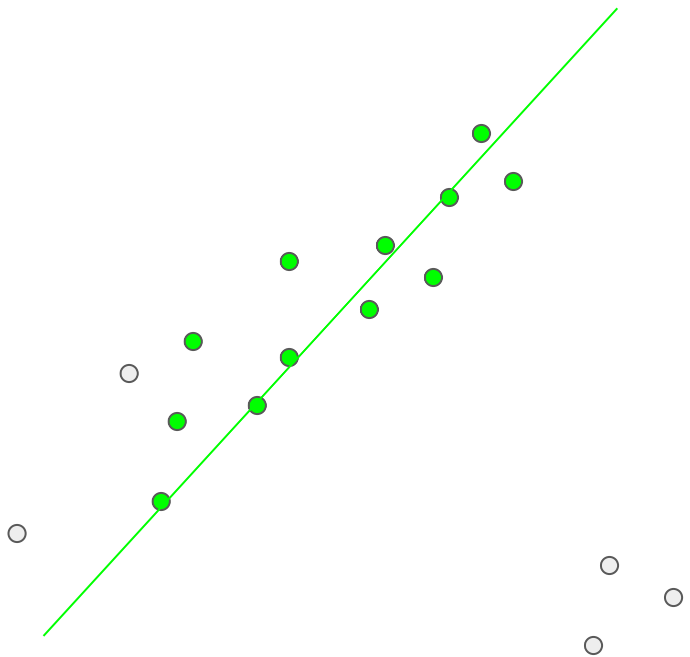
RANSAC - RANdom SAmple Consensus

- Repeat:
  - randomly pick **subset** of points
  - generate **hypothesis**
  - count **inliers**: 12



RANSAC - RANdom SAmple Consensus

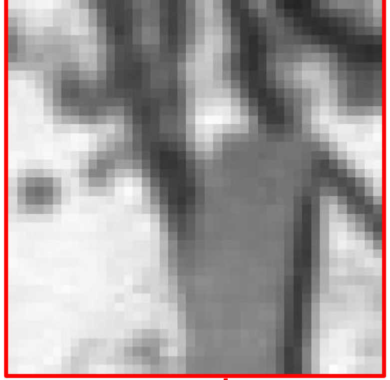
- Repeat:
  - randomly pick **subset** of points
  - generate **hypothesis**
  - count **inliers**: 12
- Return hypothesis with most inliers



Stereo Vision

Stereo Vision

Stereo Vision



128	144	120	130	128	165	184
131	152	128	114	180	200	230
122	101	111	190	205	240	255
100	40	42	20	34	50	98

Stereo Vision