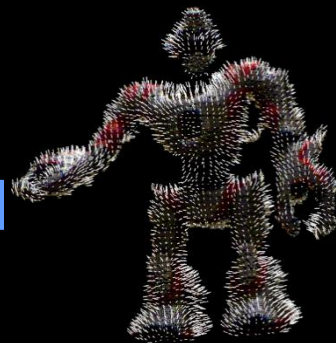


CS4495/6495

Introduction to Computer Vision



9C-L1 *3D perception*



Some slides by Kelsey Hawkins

Motivation

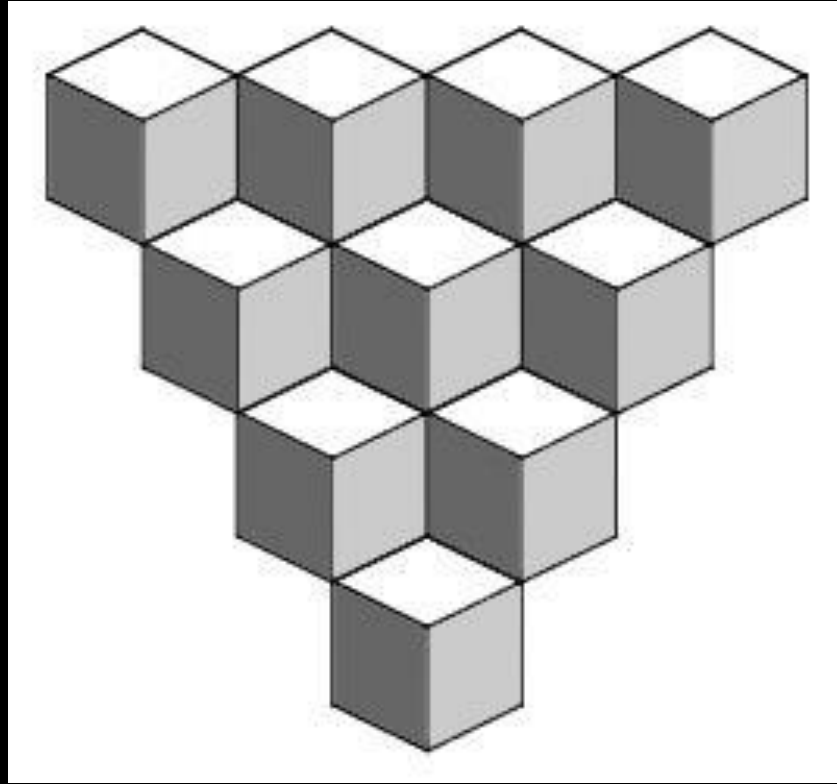
Why do animals, people & robots need vision?

- To detect and recognize objects/landmarks
 - Is that a banana or a snake? A cup or a plate?
- To find relative location of objects
 - Want to grasp fruit/tool, where should I put my body/arm?
 - Changes in elevation: steps, rocks, inclined planes

Motivation

- Determine shape
 - What is the physical 3D structure of this object?
 - Where does an object begin and the background begin?
- Find obstacles and map the environment
 - How do I get my body/arm from A to B without hitting things?
- Others – tracking, dynamics, etc.

Weaknesses of images



Surface Geometry

Weaknesses of images



Color Inconsistency

Weaknesses of monocular vision



Scale

Weaknesses of monocular vision



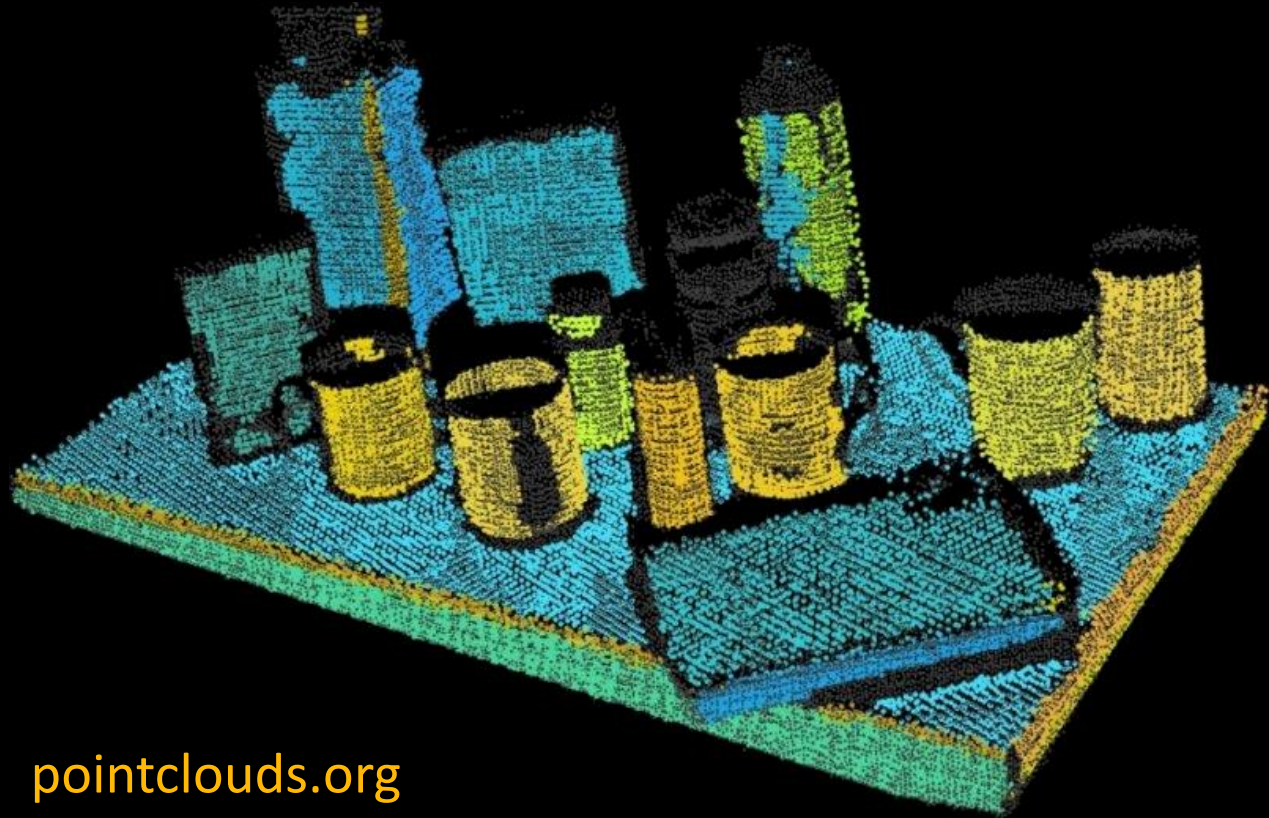
Lack of texture

Weaknesses of monocular vision



Background-foreground similarity

Potential solution: 3D sensing



pointclouds.org

Types of 3D sensing

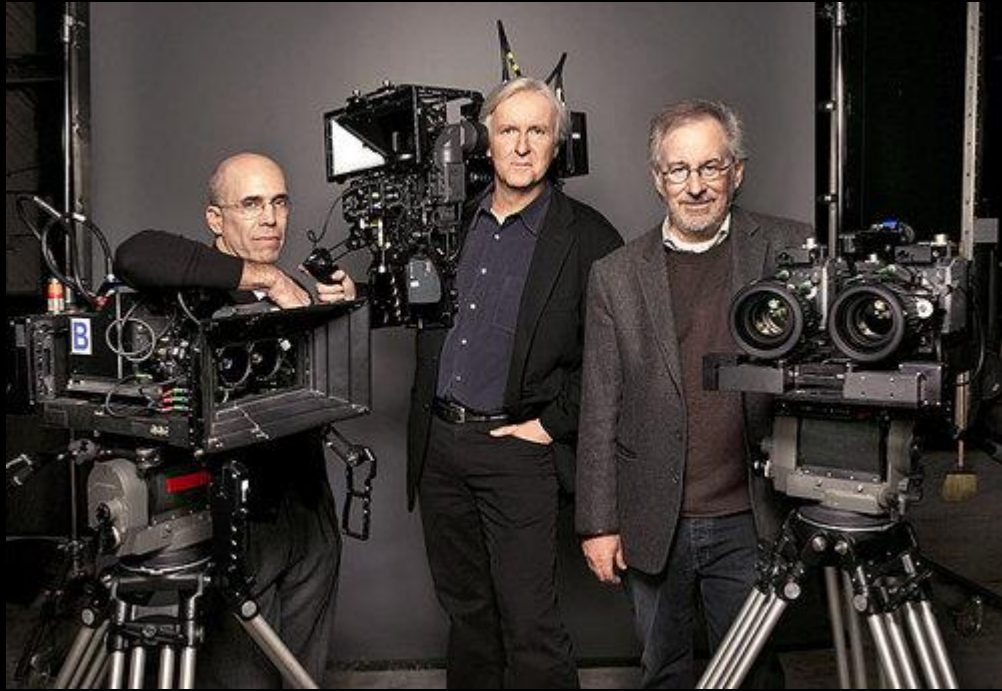
- Passive 3D sensing
 - Work with naturally occurring light
 - Exploit geometry or known properties of scenes

Passive: 3D sensors - stereo



Amateur Stereo Rigs

Passive: 3D sensors - stereo



Professional Stereo Rigs

Passive: 3D sensors – shape from (de)focus

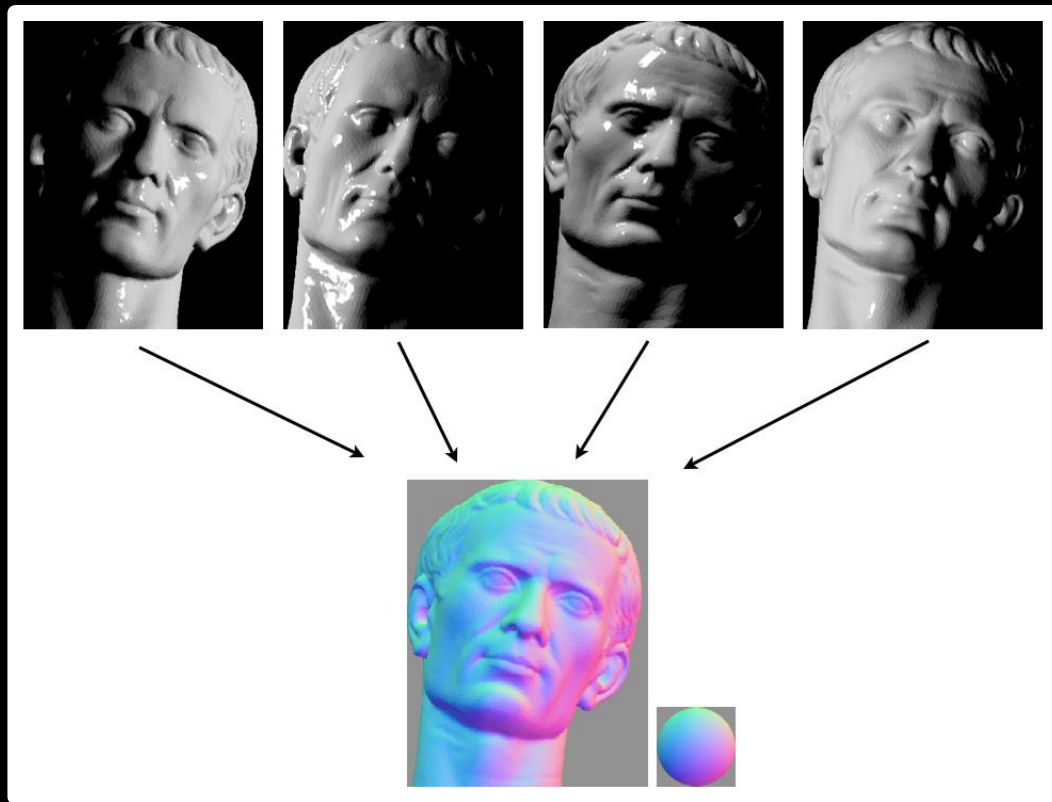


Nayar, Watanabe, and Noguchi 1996

Types of 3D sensing

- Passive 3D sensing
 - Work with naturally occurring light
 - Exploit geometry or known properties of scenes
- Active 3D sensing
 - Project light or sound out into the environment and see how it reacts
 - Encode some pattern which can be found in the sensor readings

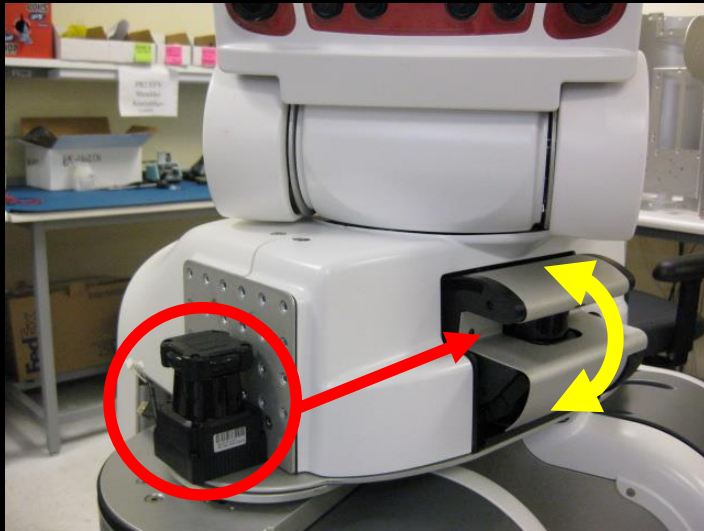
Active: Photometric stereo



Active: Time of flight

Bounce signal off of surface, record time to come back

$$d = v * t / 2$$



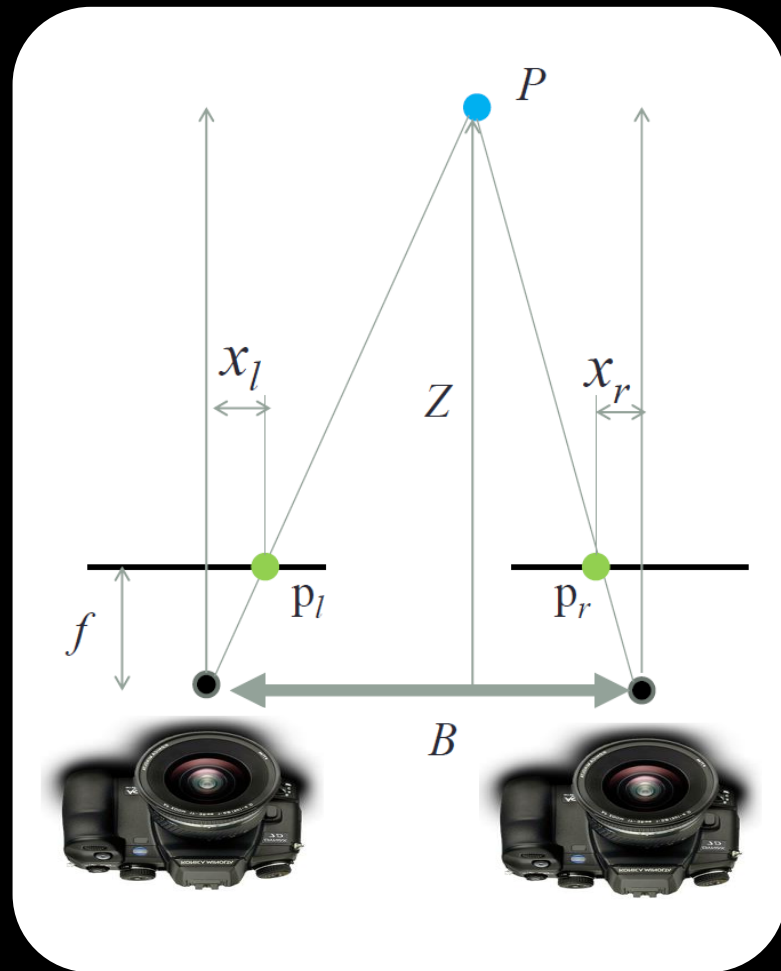
LIDAR: Laser Range finder



SONAR: Sound Transceiver

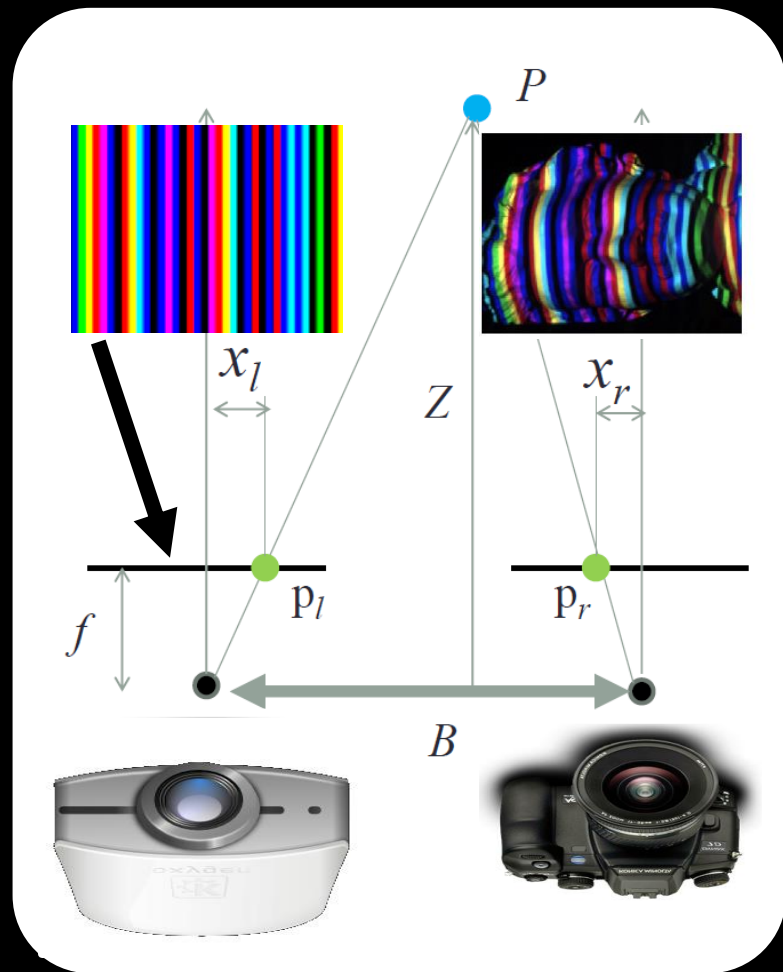
Structured light

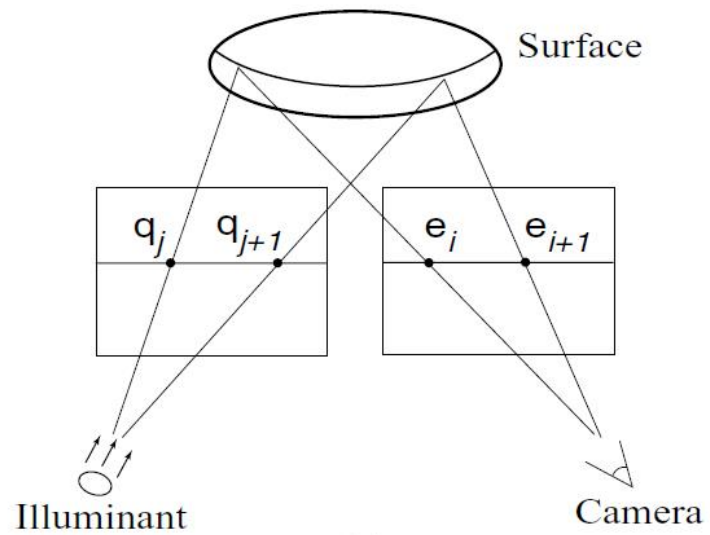
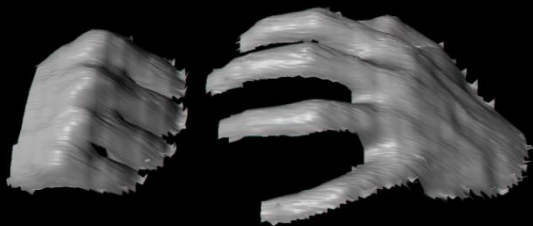
- Like stereo



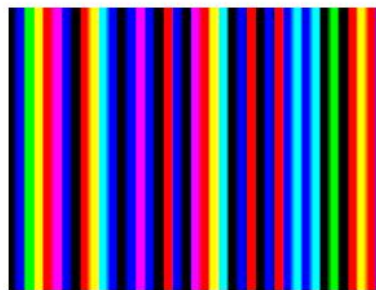
Structured light

- Like stereo
- But replace one camera with a projector

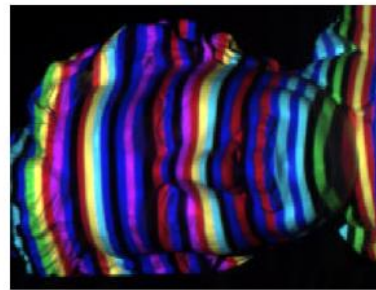




(a)

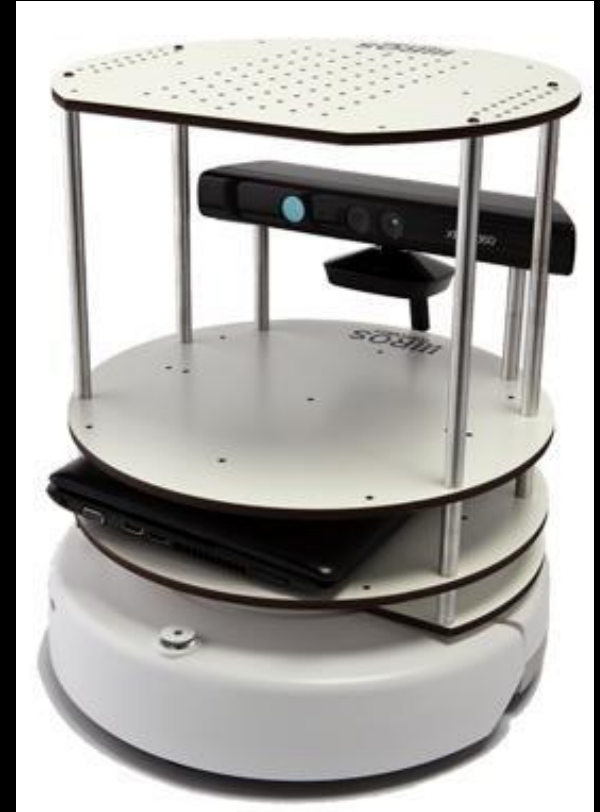
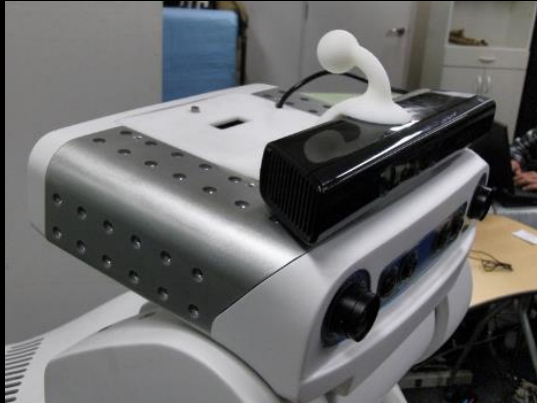
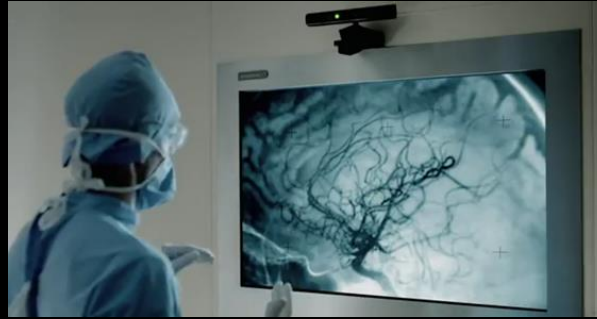


(b)



(c)

Infrared structured light



Infrared structured light



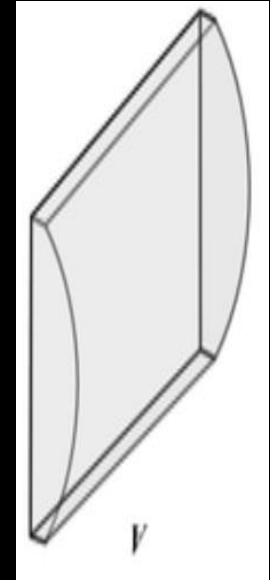
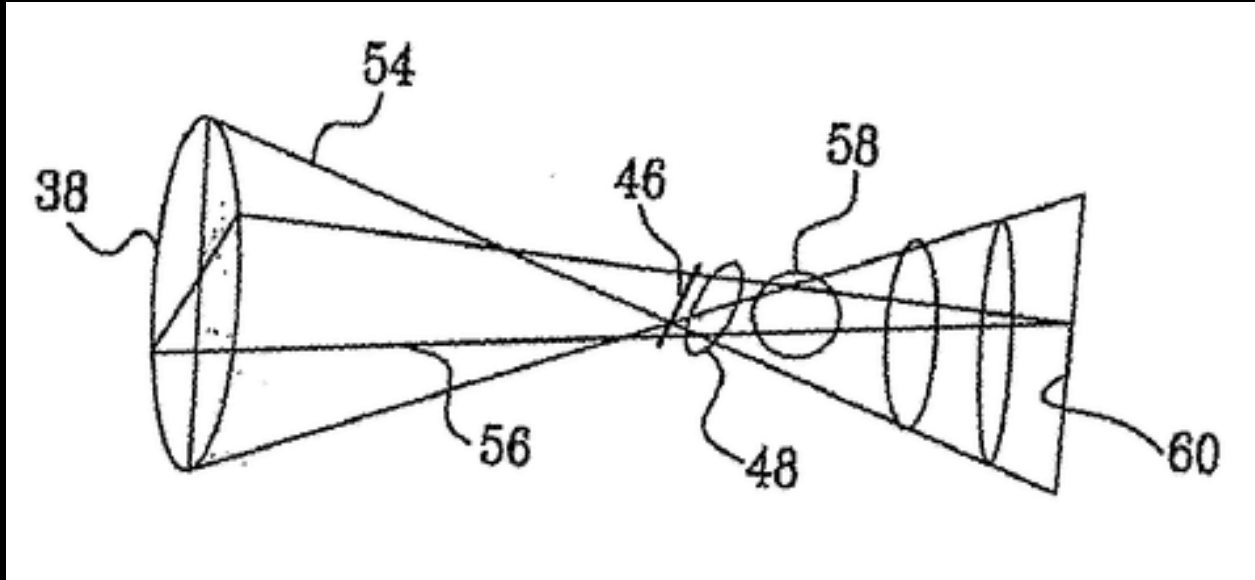
How does Kinect work?

Not public, but PrimeSense patent(s) describe at least two methods:

- A very clever optics trick that changes the shape of a of projected pattern as a function of depth
- A more standard yet well engineered approach counting on have a known fixed geometry and a fixed pseudo random pattern of projected points

How the Kinect sensor works - focus

Cylindrical lens: Only focuses light in one direction



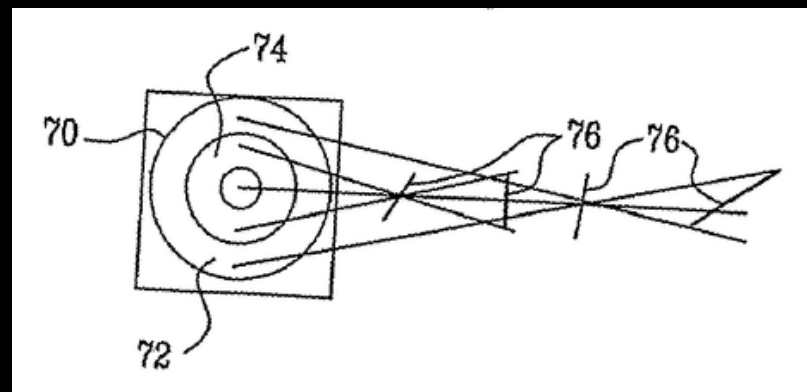
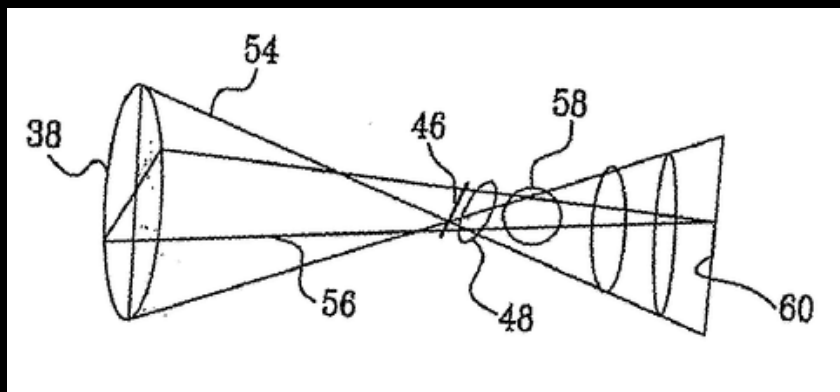
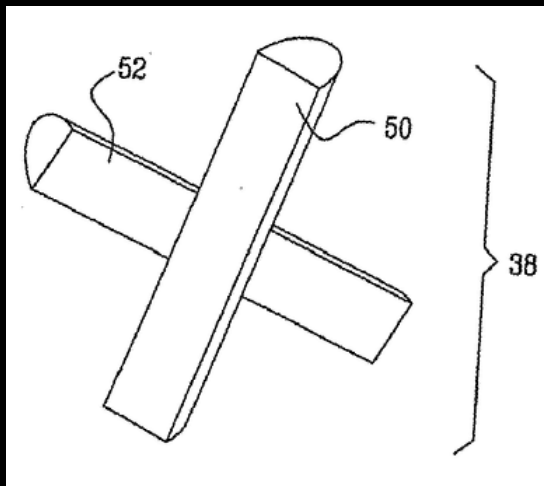
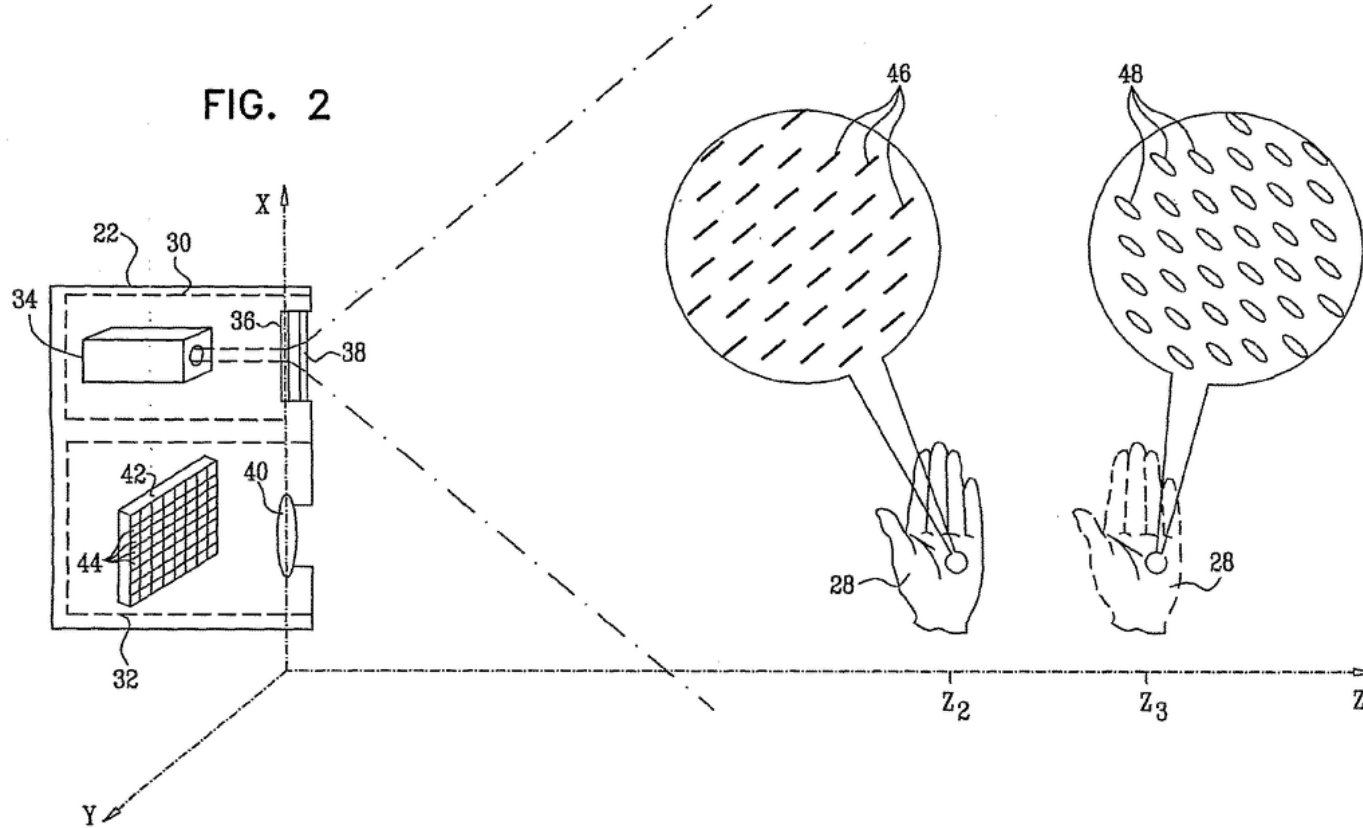
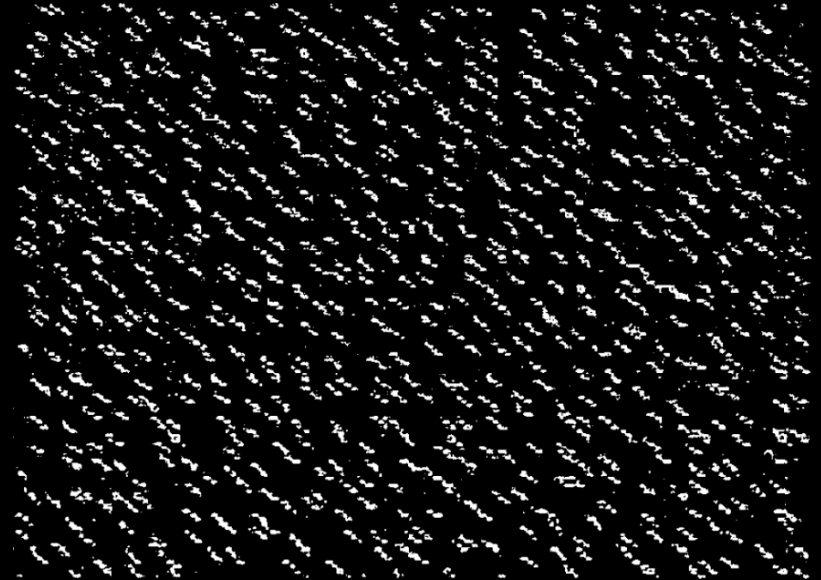
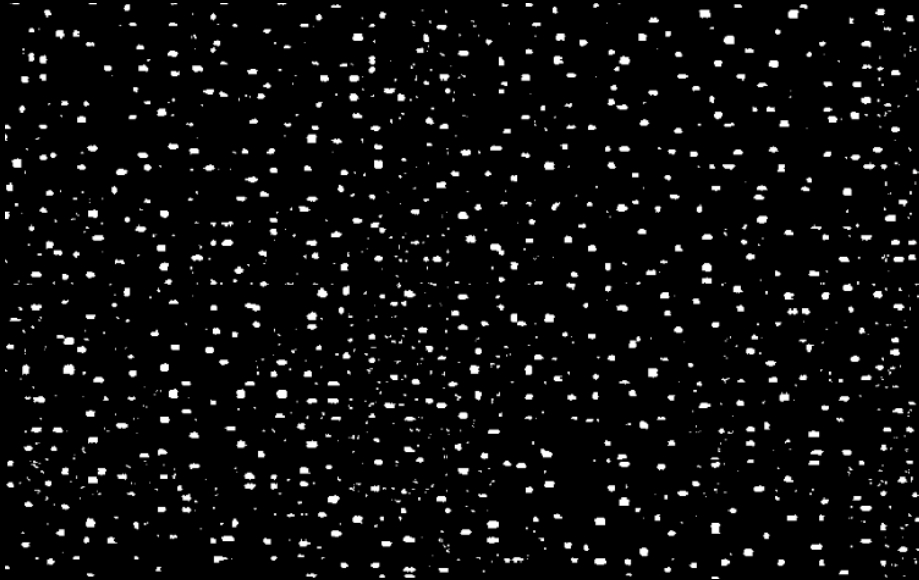
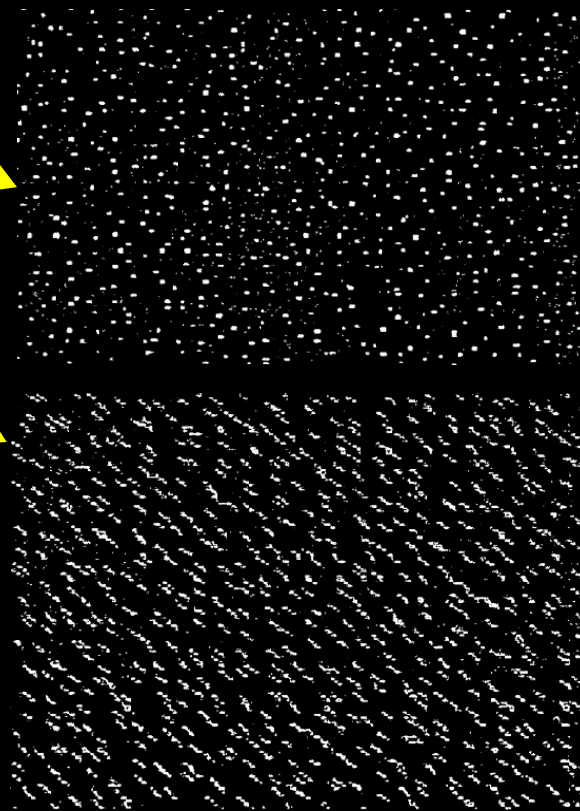
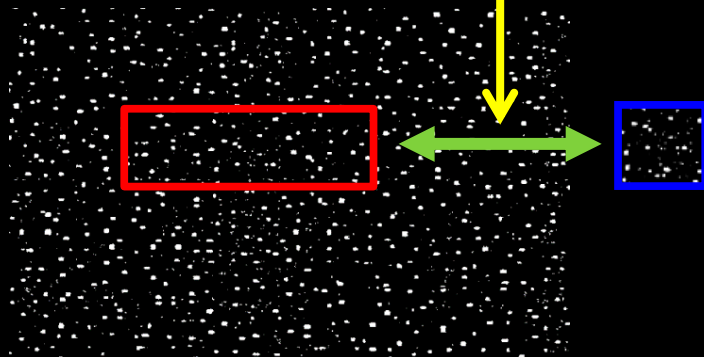
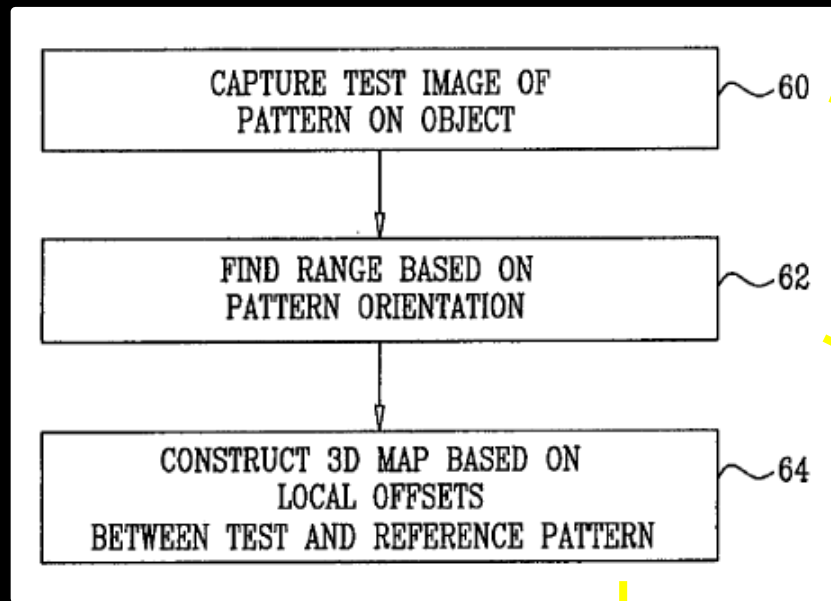


FIG. 2



Orientation is a function of distance!

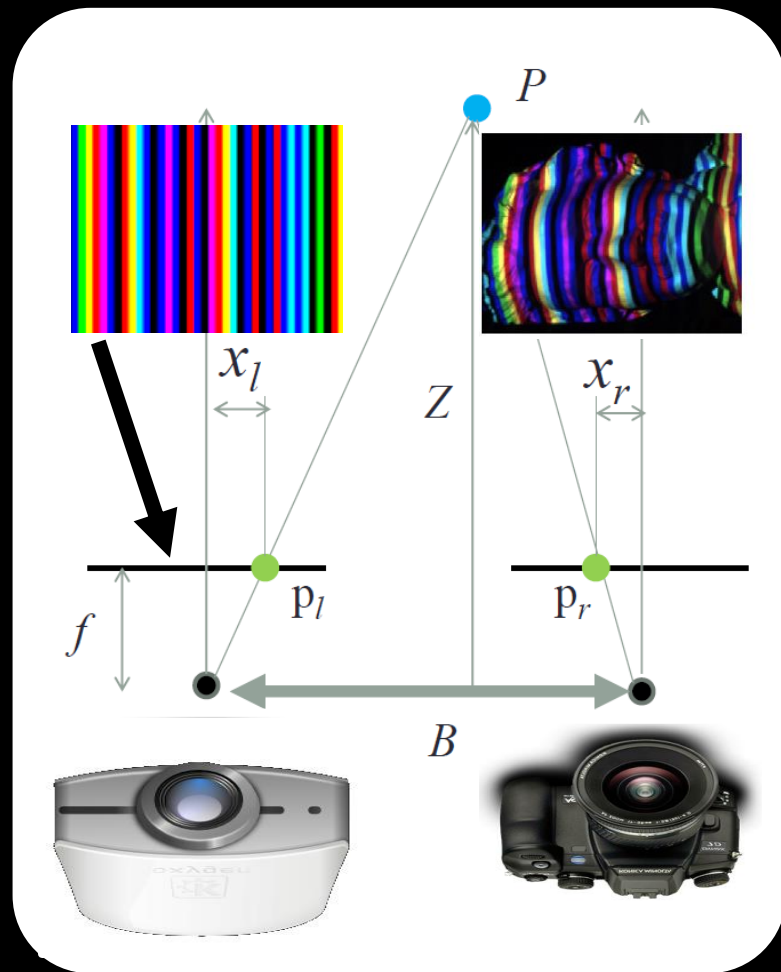




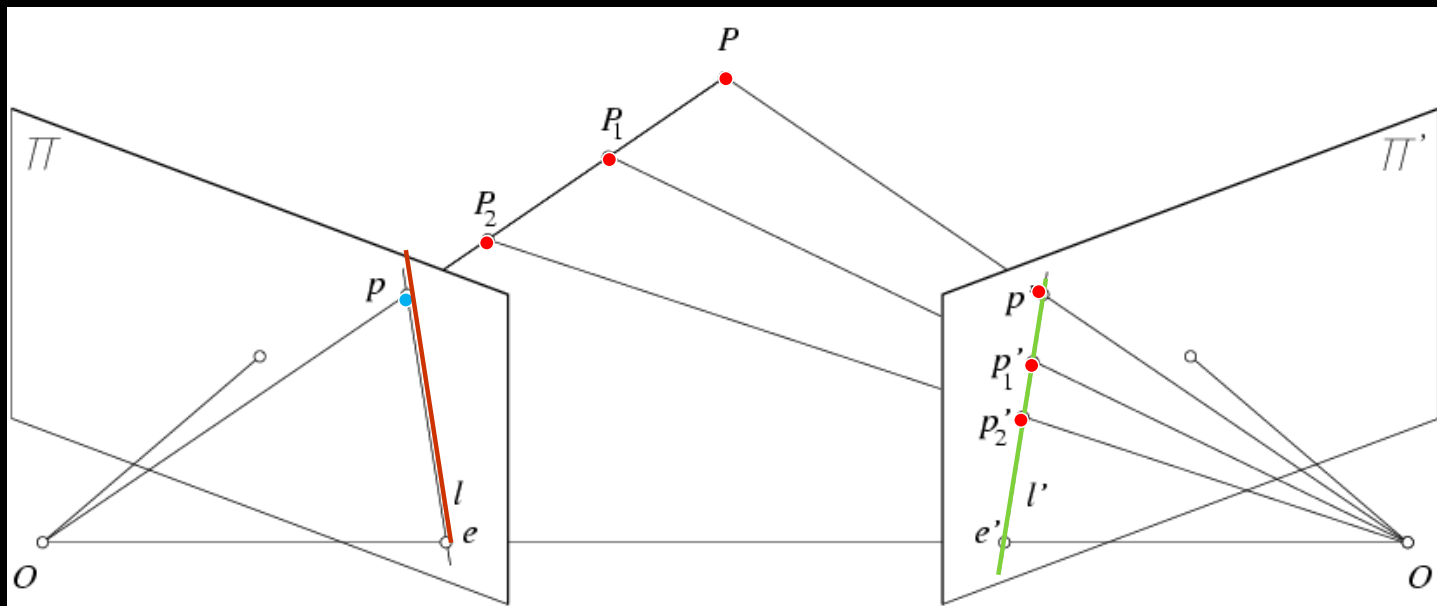
Pseudo-random speckle pattern

Structured light

- Like stereo
- But replace one camera with a projector



Same stereo algorithms apply

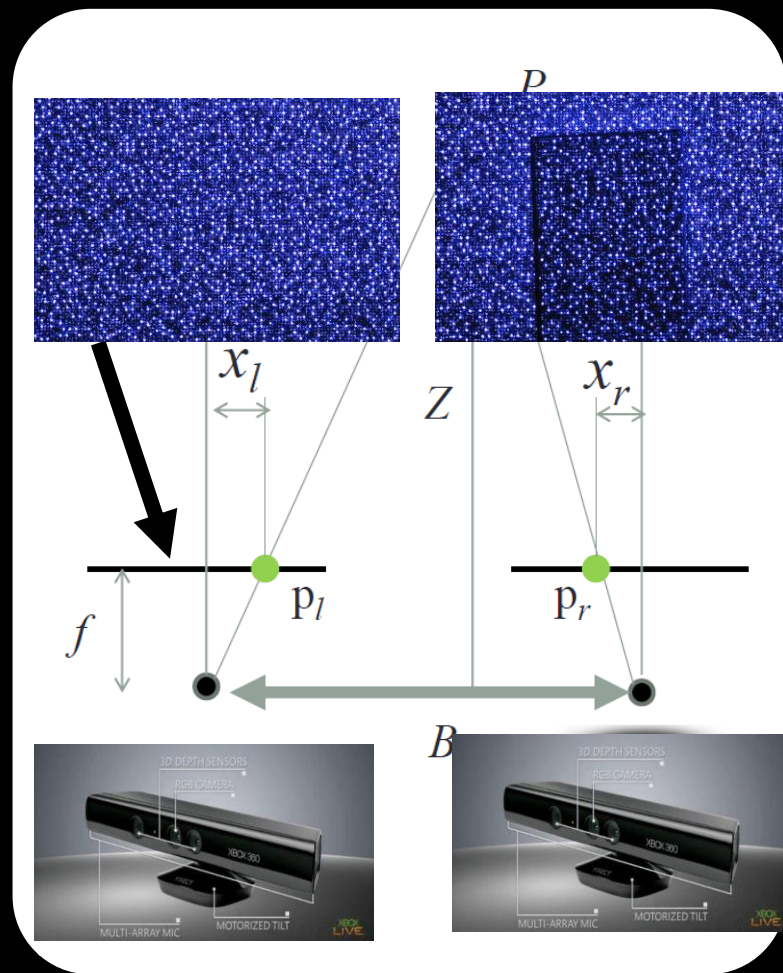


Projector

Sensor

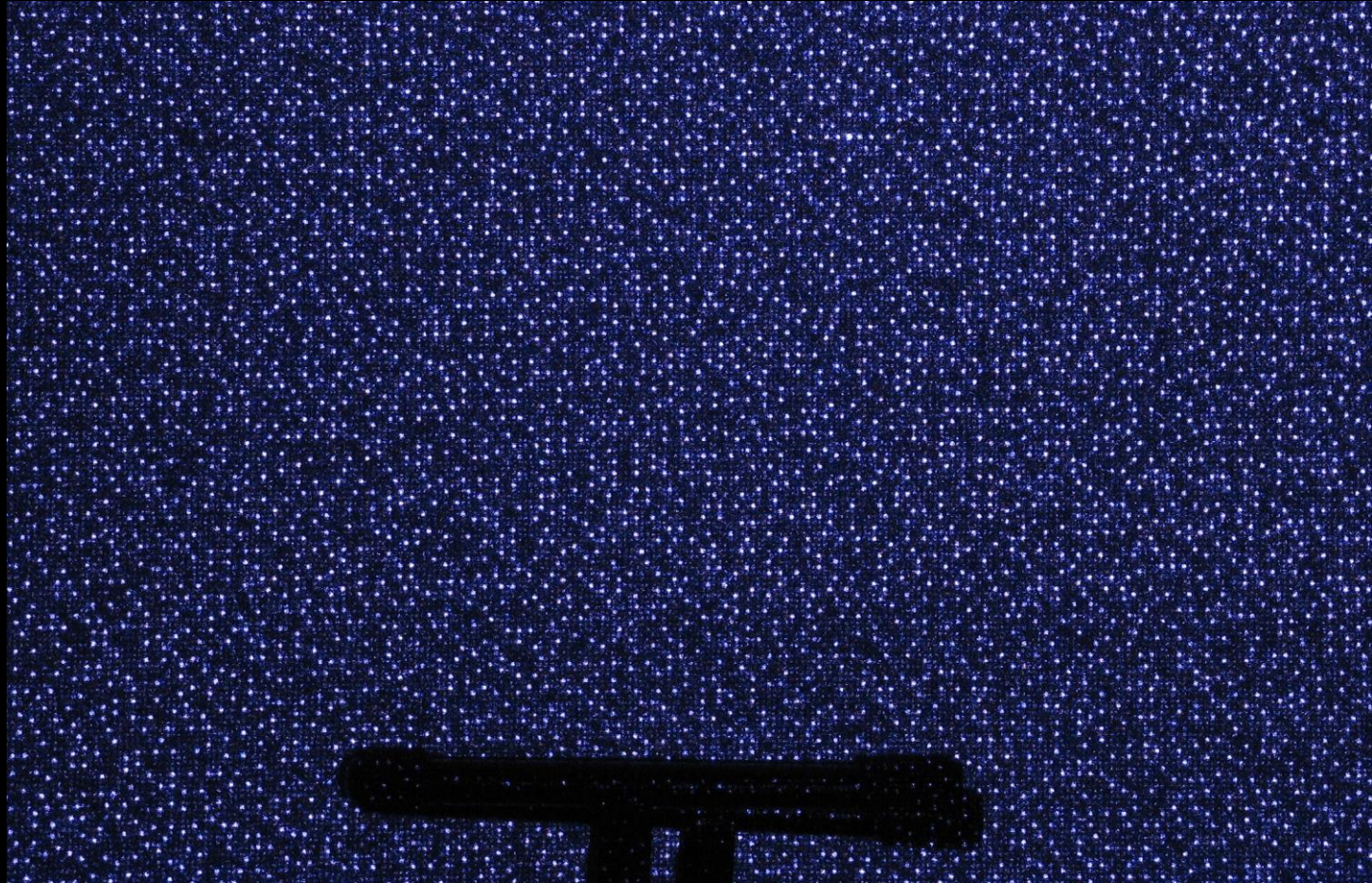
Structured light

- Like stereo
- But replace one camera with a projector
- Can do it with speckle pattern too...



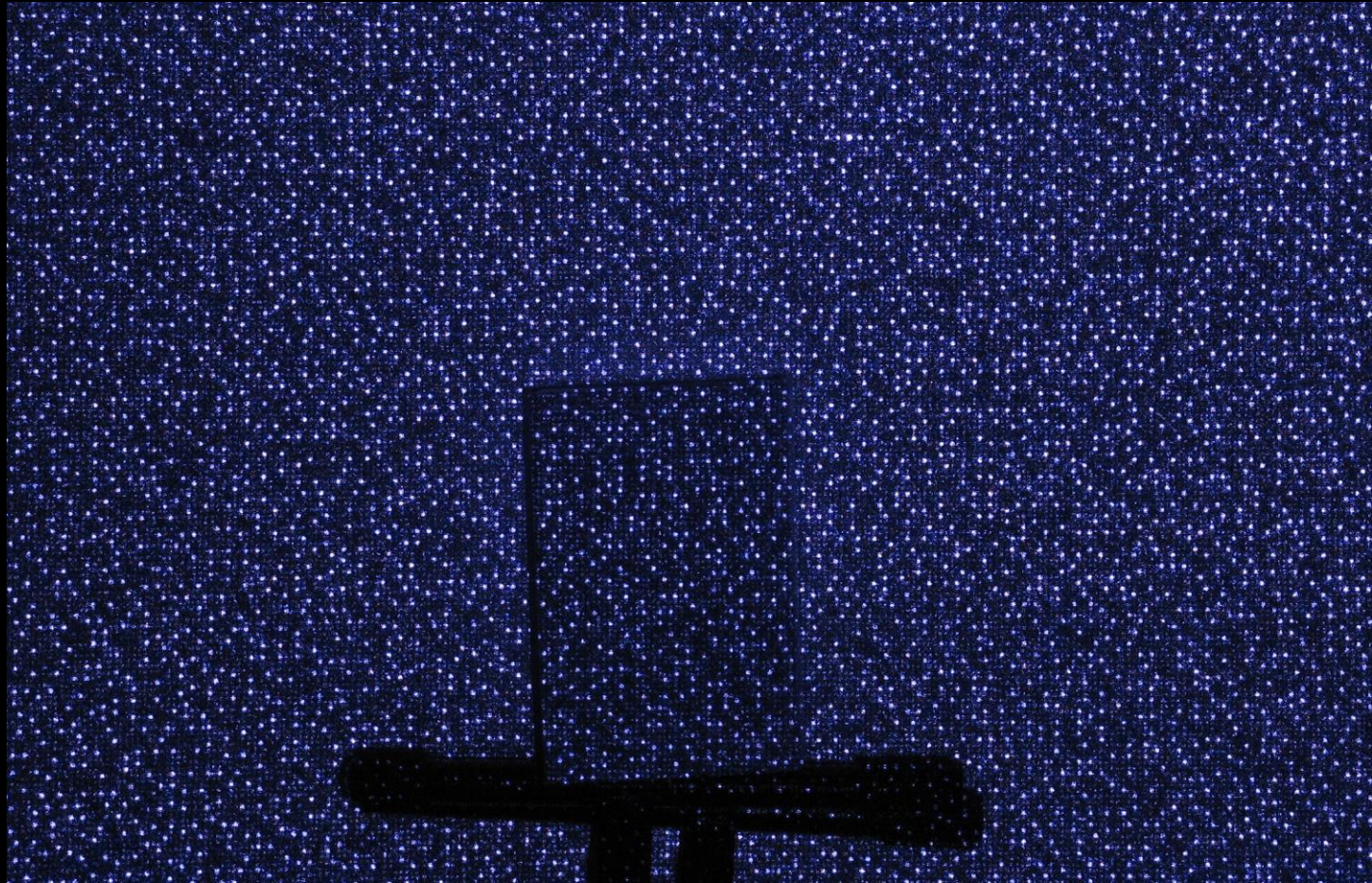
Example: Book vs. No Book

futurepicture.org/?p=97



Example: Book vs. No Book

futurepicture.org/?p=97



1. Detect dots (“speckles”) and label them unknown

2. Randomly select a region anchor, a dot with unknown depth

a) Windowed search via normalized cross correlation along scanline

- Check that best match score is greater than threshold; if not, mark as “invalid” and go to 2

b) Region growing

1. Neighboring pixels are added to a queue
2. For each pixel in queue, initialize by anchor's shift; then search small local neighborhood; if matched, add neighbors to queue
3. Stop when no pixels are left in the queue

3. Stop when all dots have known depth or marked “invalid”

<http://www.wipo.int/patentscope/search/en/WO2007043036>

Projected IR vs. Natural Light Stereo

- What are the advantages of IR?
 - Works in low light conditions
 - Does not rely on having textured objects
 - Not confused by repeated scene textures
 - *Can tailor algorithm to produced pattern*

Projected IR vs. Natural Light Stereo

- What are advantages of natural light?
 - **Works outside**, anywhere with sufficient light
 - Resolution limited only by sensors, not projector

Projected IR vs. Natural Light Stereo

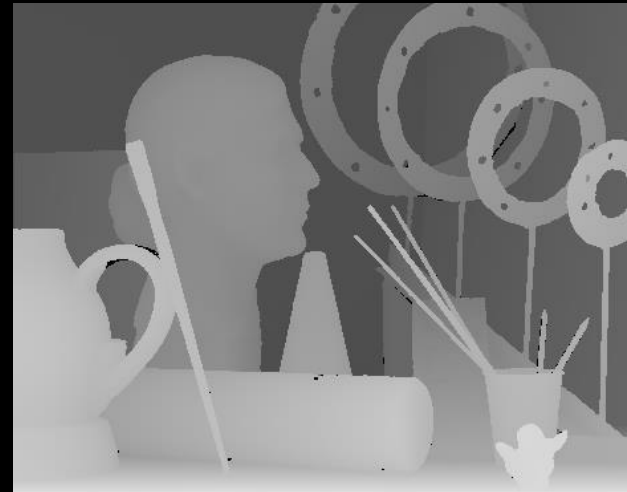
- Difficulties with both
 - Very dark surfaces may not reflect enough light
 - Specular reflection in mirrors or metal causes trouble

Representing depth scenes

- Natural: depth image
- A little more nuanced: point clouds

Depth Images

- Advantages
 - Dense representation
 - Gives intuition about occlusion and free space
 - Depth discontinuities are just edges on the image



Depth Images

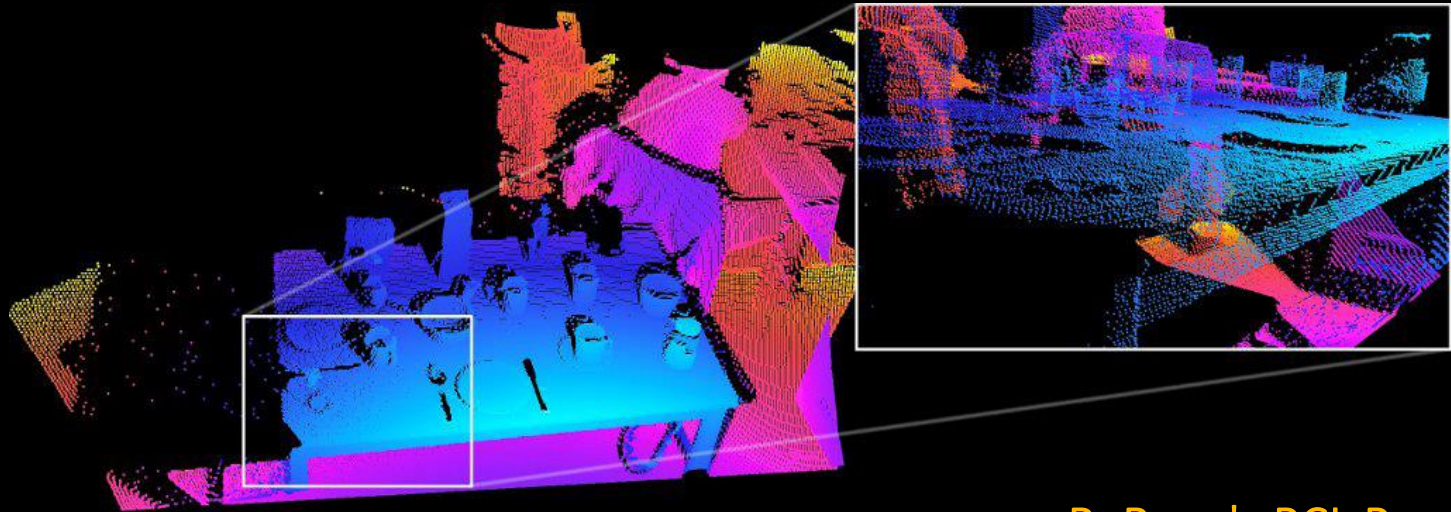
- Disadvantages
 - Viewpoint dependent, can't merge
 - Doesn't capture physical geometry
 - Need actual 3D locations of camera(s)



Point Clouds

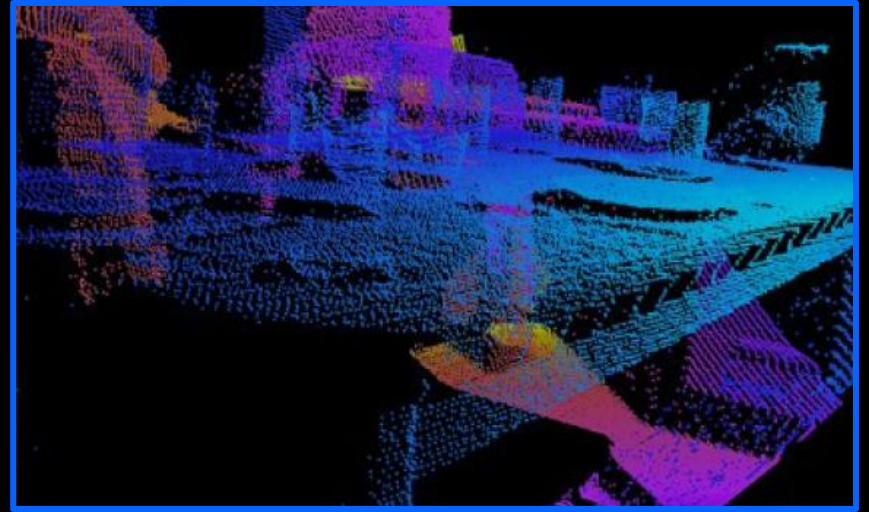
Take every depth pixel and put it out in the world

- What can this representation tell us?
- What information do we lose?



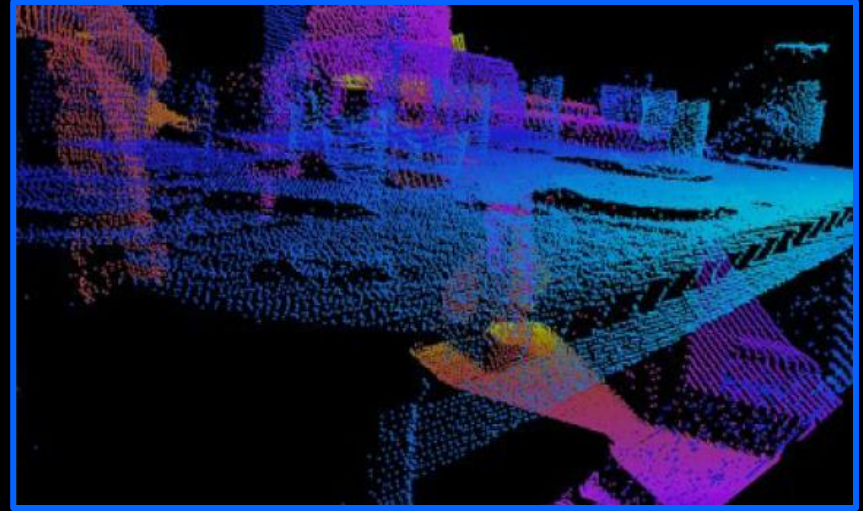
Point Clouds

- Advantages
 - Viewpoint independent
 - Captures surface geometry
 - Points represent physical locations



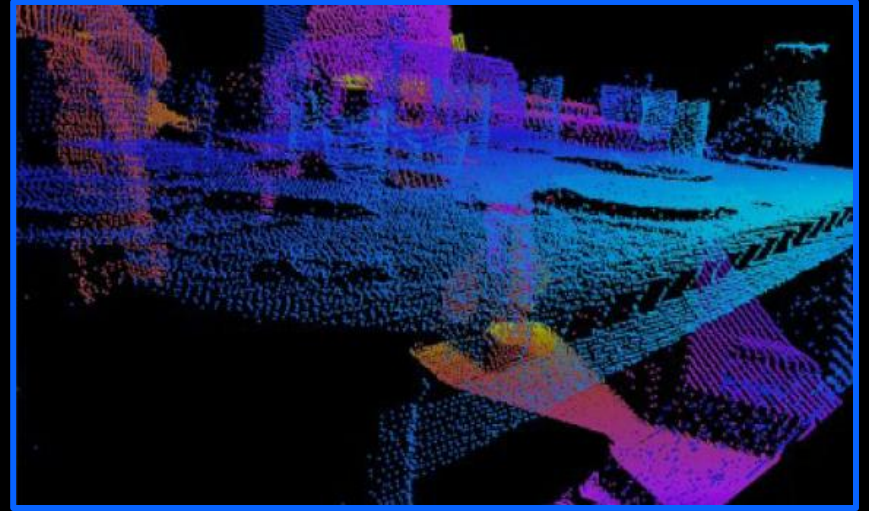
Point Clouds

- Disadvantages
 - Sparse representation
 - Lost information about free space and unknown space
 - Variable density based on distance from sensor



Point Clouds

- Biggest advantage:
 - PCL – Point Cloud Library



Point Clouds

- <http://pointclouds.org>

The screenshot shows the Point Cloud Library (PCL) website. The browser address bar displays pointclouds.org/about/. The website header includes the PCL logo and navigation links: About, News, Blog, Downloads, Media, Jobs, Documentation, Contact, and GSoC'14. A green banner below the header contains the text "DONATE TO OPEN PERCEPTION FOUNDATION".

About

What is PCL?

The Point Cloud Library (or PCL) is a **large scale, open project** [1] for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the-art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them -- to name a few.

PCL is released under the terms of the [3-clause BSD license](#) and is open source software. It is **free for commercial and research use**.

PCL is **cross-platform**, and has been successfully compiled and deployed on Linux, MacOS, Windows, and **Android/iOS**. To simplify development, PCL is split into a series of smaller code libraries, that can be compiled separately. This modularity is important for distributing PCL on platforms with reduced computational or size constraints (for more information about each module see the [documentation](#) page). Another way to think about PCL is as a graph of code libraries, similar to the [Boost](#) set of C++ libraries. Here's an example:

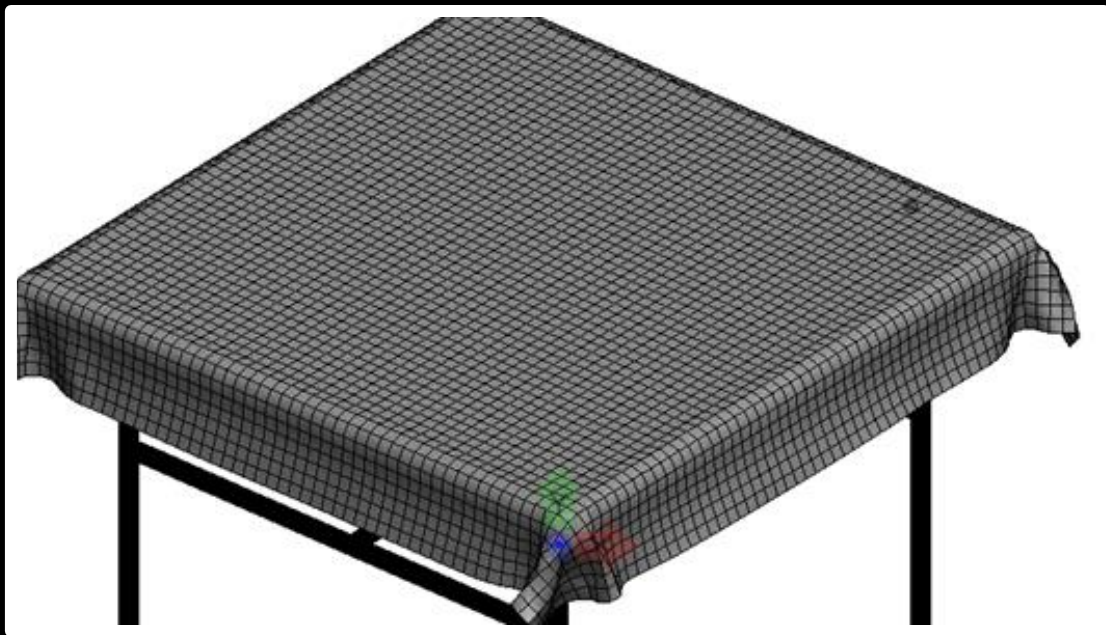
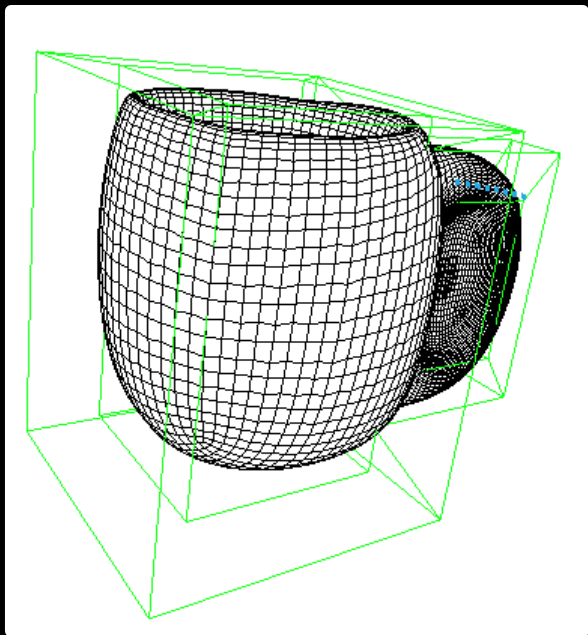
```
graph TD
    search --> octree
    io --> octree
    octree --> sample_consensus
    sample_consensus --> filters
    sample_consensus --> segmentation
    filters --> visualization
    filters --> registration
    segmentation --> visualization
    segmentation --> registration
    segmentation --> kdtree
    kdtree --> visualization
    kdtree --> registration
    kdtree --> features
    kdtree --> surface
    kdtree --> common
    features --> visualization
    features --> registration
    features --> keypoints
    features --> range_image
    surface --> visualization
    surface --> registration
    surface --> keypoints
    surface --> range_image
    range_image --> common
    range_image --> tracking
    tracking --> common
```

Who is developing PCL?

The project is being developed by a large number of engineers and scientists from many different organizations, geographically distributed all around the world, including:

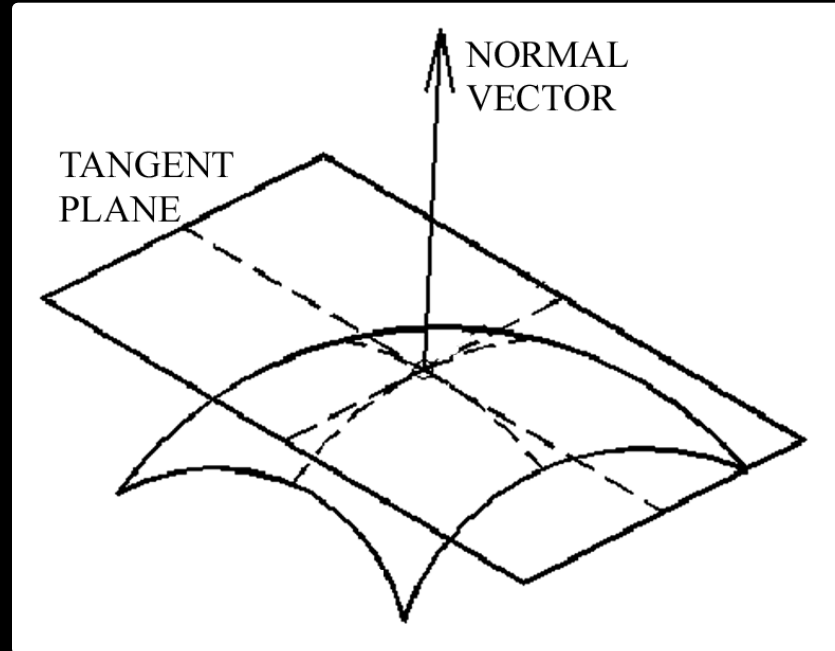
Point Clouds and Surfaces

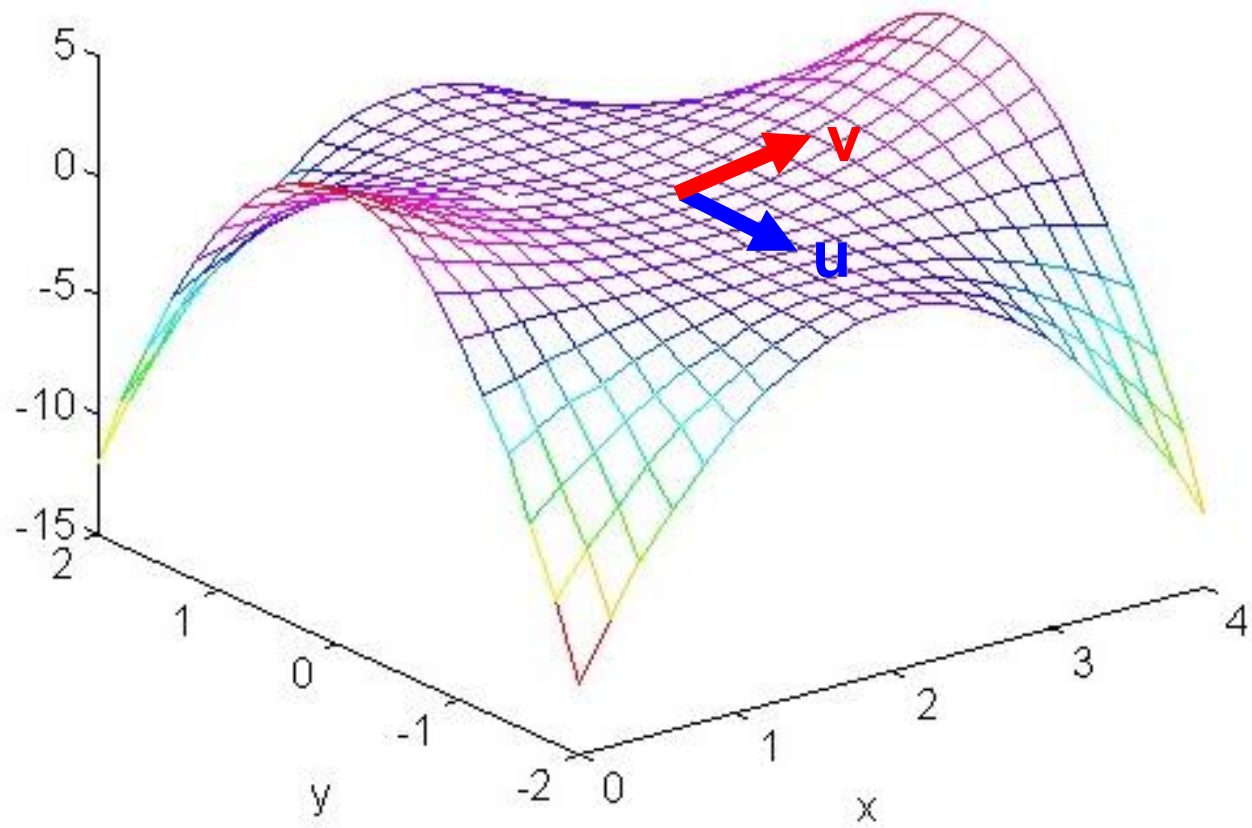
- Point clouds are sampled from object surfaces
- The concept of volume is inferred, not perceived



Surfaces

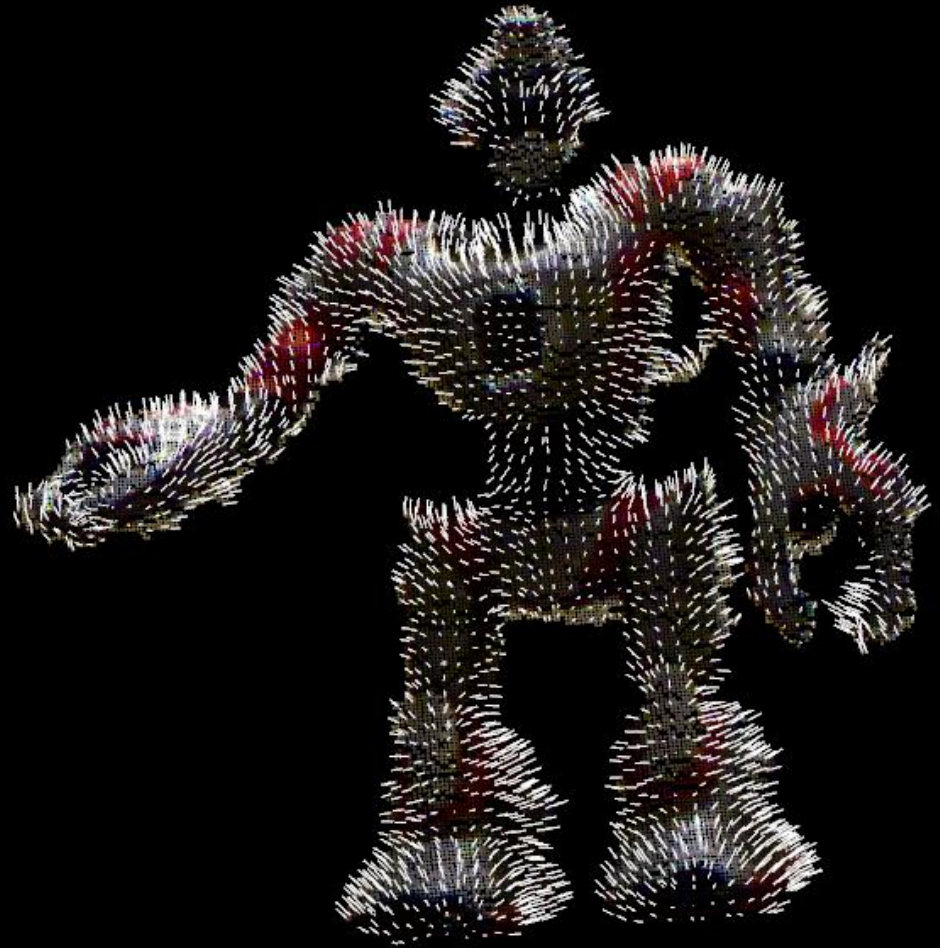
- Tangent plane – defined by normal
- First-order approximation





Surface Normals

- Size of patch is like width of Gaussian in image gradient calculation
- We can use them to find planes

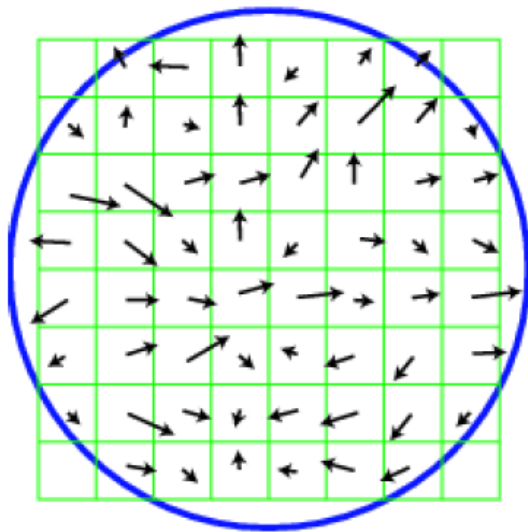


Viewpoint Invariance

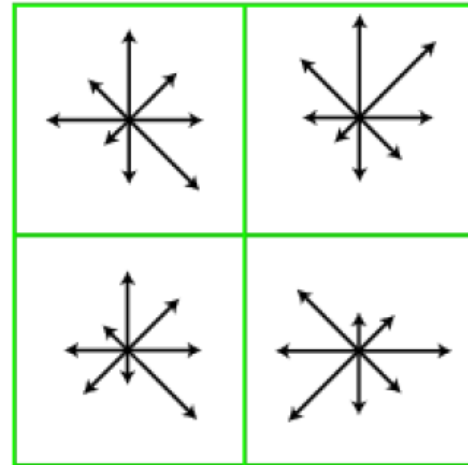
- Want to find unique patches of surface geometry
- What type of invariance do we need?
- Need *viewpoint* invariance
 - Translation + orientation
 - Color and texture come automatically!

Point Feature Histograms

Remember SIFT?



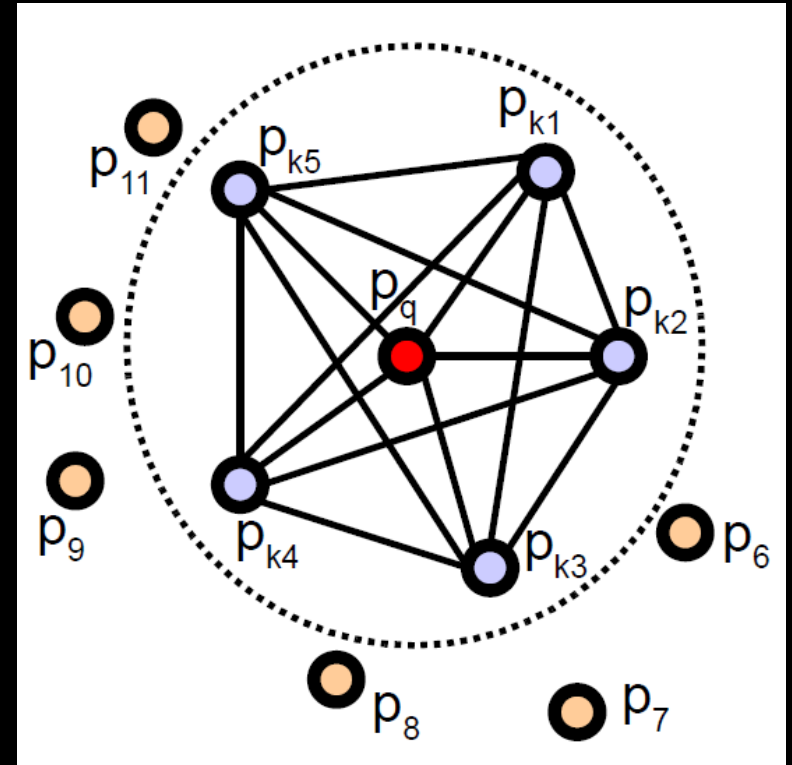
(a) image gradients



(b) keypoint descriptor

Point Feature Histograms

- At a point, take a ball of points around it
- For every pair of points, find the relationship between the two points and their normals
- Must be frame independent



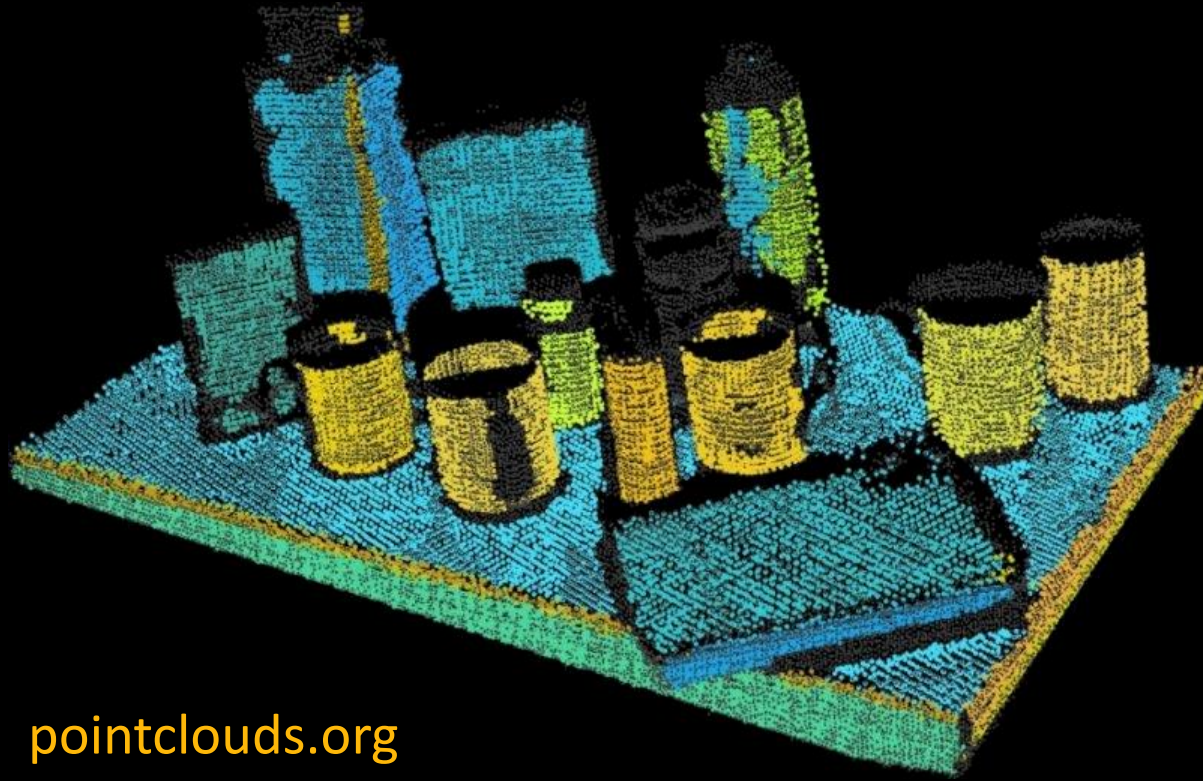
Point Feature Histograms

- Find these for variables for every pair in the ball
- Build a $5 \times 5 \times 5 \times 5$ histogram of the variables
 - Often the distance variable is excluded
 - In this case, we have a 125-long feature vector
- Use this just like a SIFT feature descriptor
- Usually, a sped-up version called Fast Point Feature Histograms is used for real-time applications

Point Cloud Software

- Point Cloud Library (PCL)
 - <http://pointclouds.org>
- Robot Operating System (ROS)
 - Framework for building systems
 - <http://www.ros.org>
- Drivers for Kinect and other PrimeSense sensors
 - http://www.ros.org/wiki/openni_launch

PCL Example: RANSAC Cylinder Segmentation



pointclouds.org