Lecture 19:

Depth Cameras

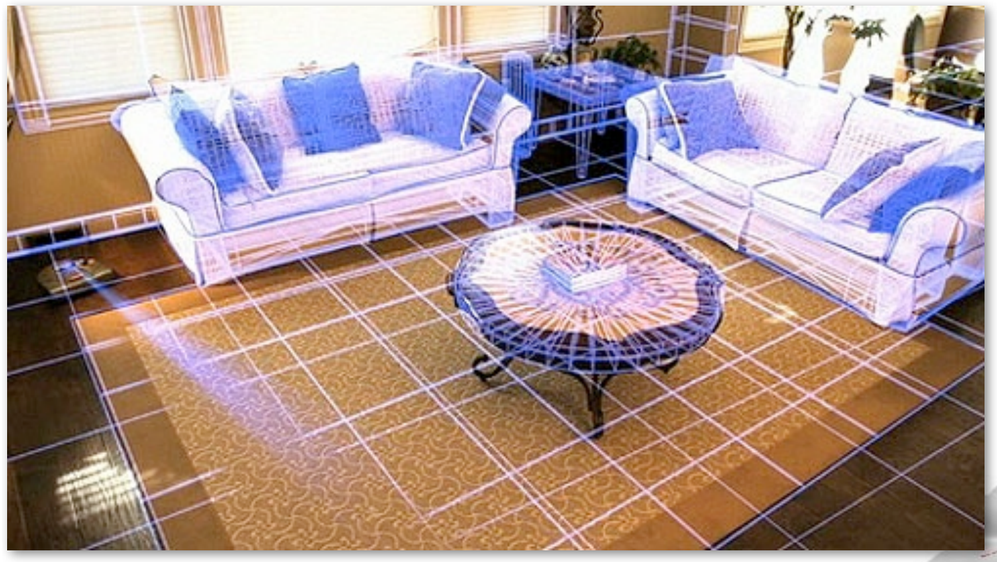
Visual Computing Systems
CMU 15-869, Fall 2013

Continuing theme: computational photography

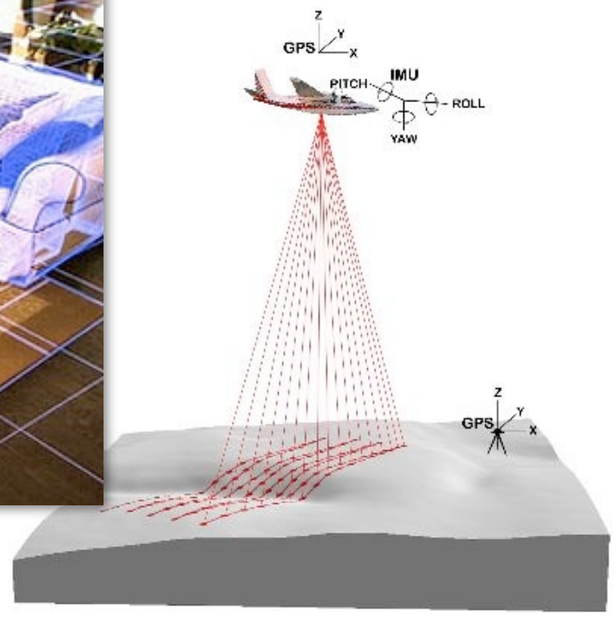
- Cameras capture light, then extensive processing produces the desired image

- Today:
  - Capturing scene depth in addition to light intensity

Why might we want to know the depth of scene objects?



Navigation



Scene Understanding

Object tracking

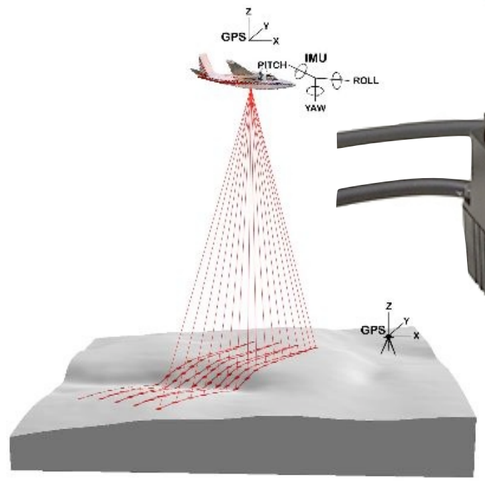
Mapping

Segmentation

Credit: Blashko et al. CVPR 13 Tutorial

Depth from time-of-flight

- **Conventional LIDAR**
  - Laser beam scans scene (rotating mirror)
  - Low frame rate to capture entire scene



- **"Time-of-flight" cameras**
  - No moving beam, capture entire image of scene with each light pulse
  - Special CMOS sensor records a depth image
  - High frame rate
  - Formerly TOF cameras were low resolution, expensive…
  - TOF camera featured in the upcoming XBox One depth sensor (today we will first talk about the original XBox 360 implementation)

Computing depth from images

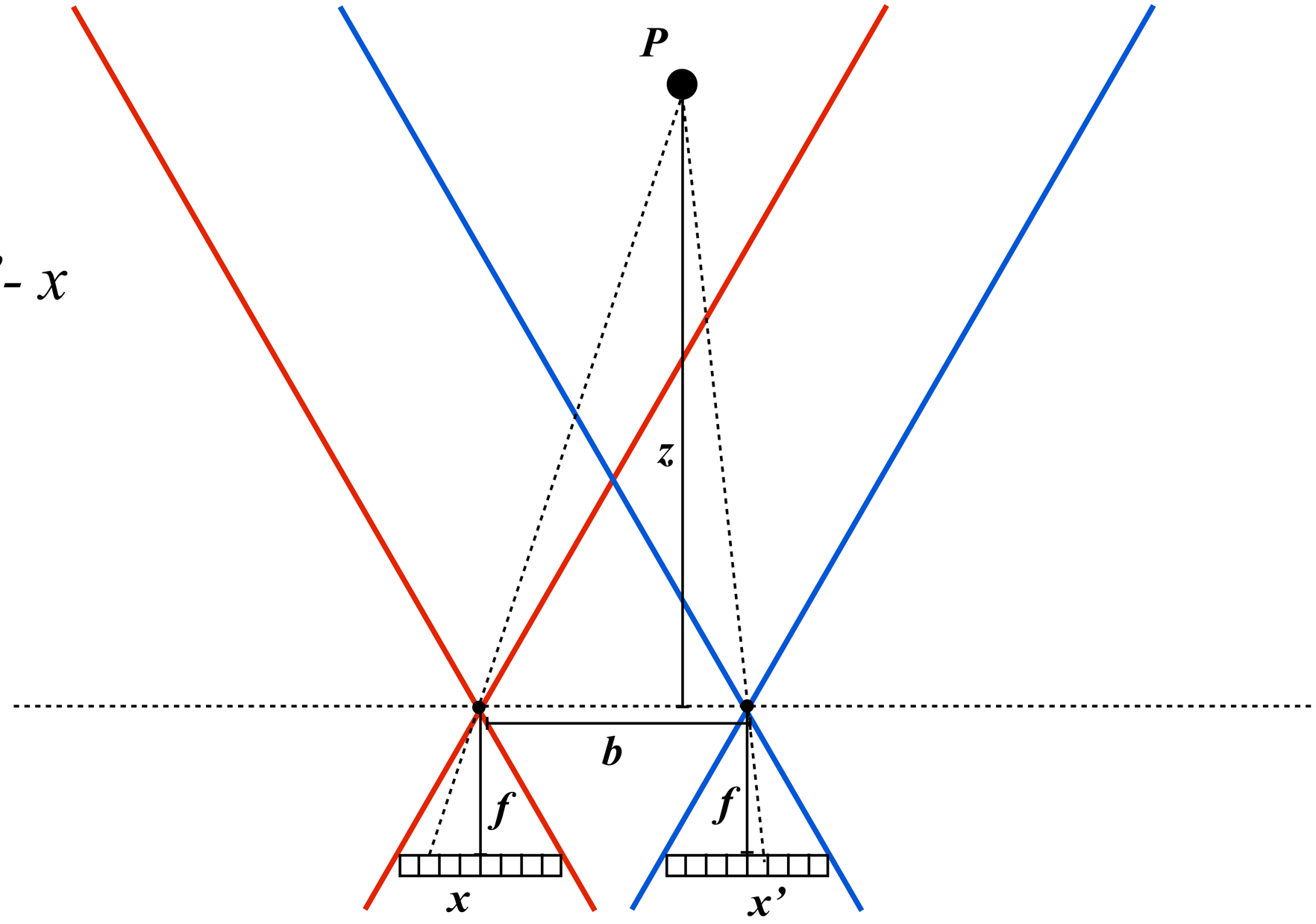
Binocular stereo 3D reconstruction of point *P*: depth from disparity

Focal length: f

Baseline: b

Disparity: $d = x' - x$

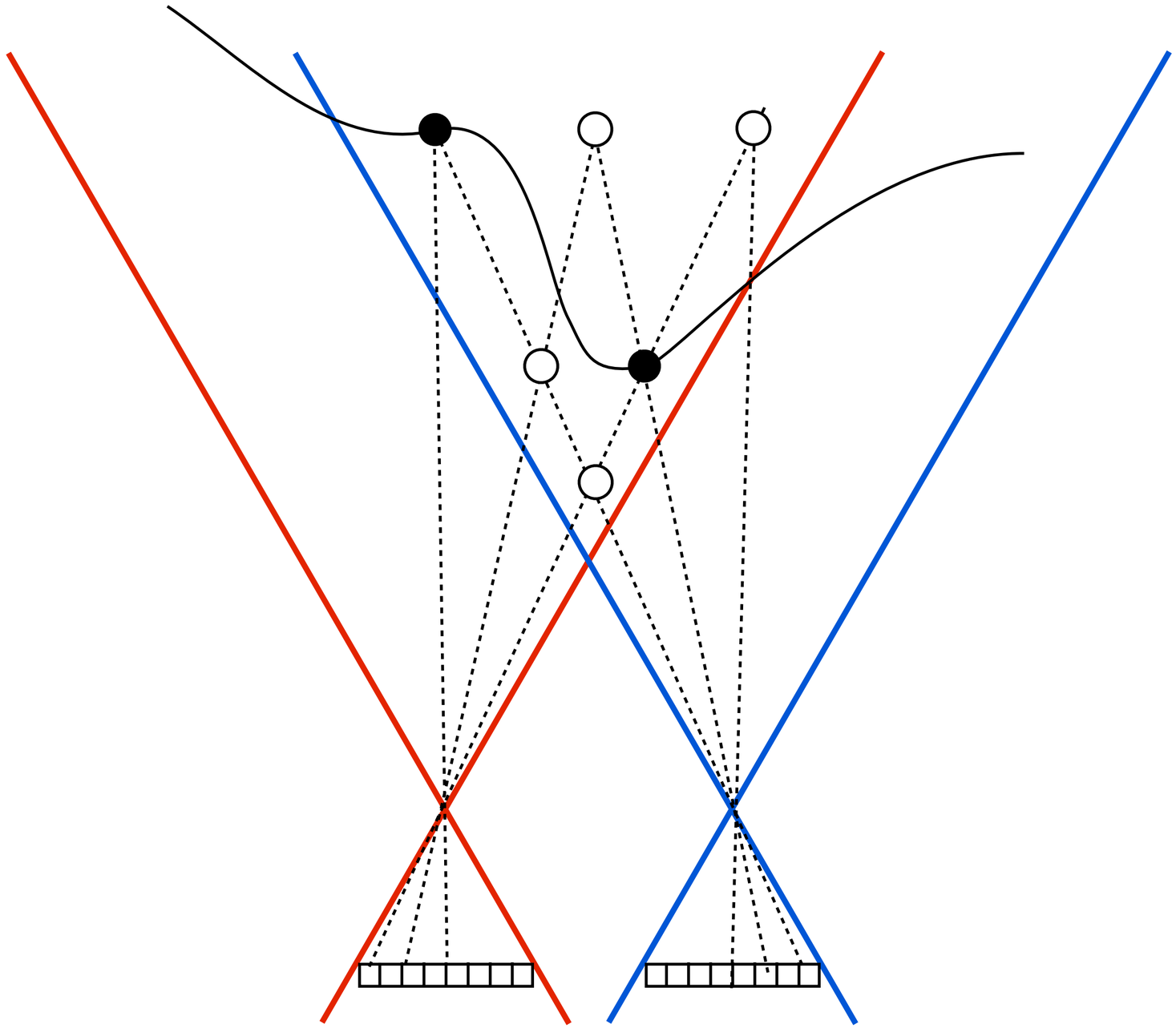
$$z = \frac{bf}{d}$$



Simple reconstruction example: cameras aligned (coplanar sensors), separated by known distance, same focal length

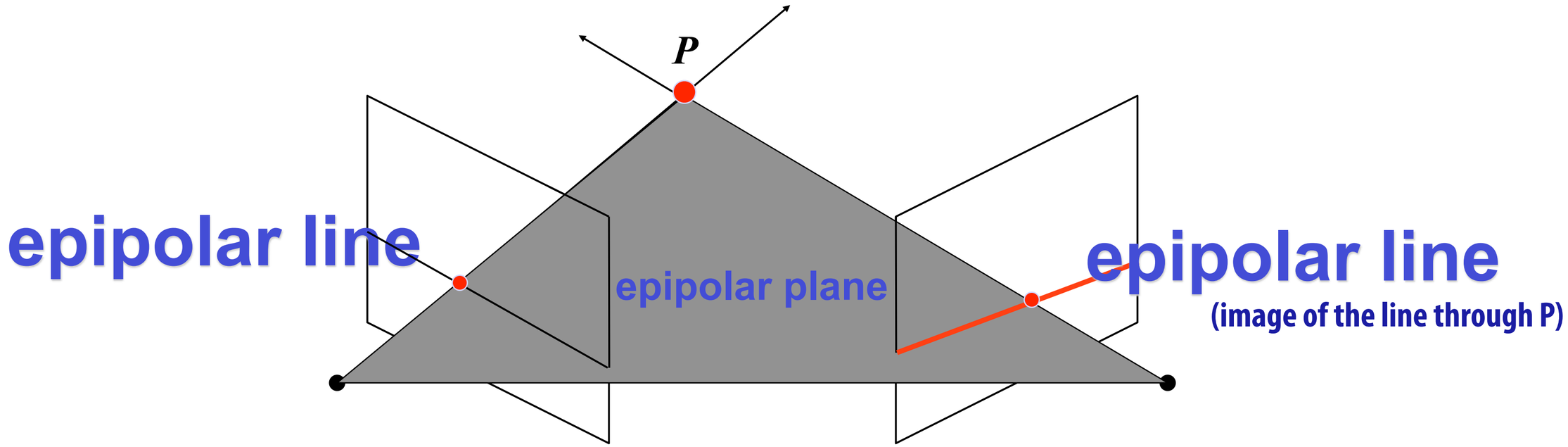
Correspondence problem

How to determine which pairs of pixels in image 1 and image 2 correspond to the same scene point?



Epipolar constraint

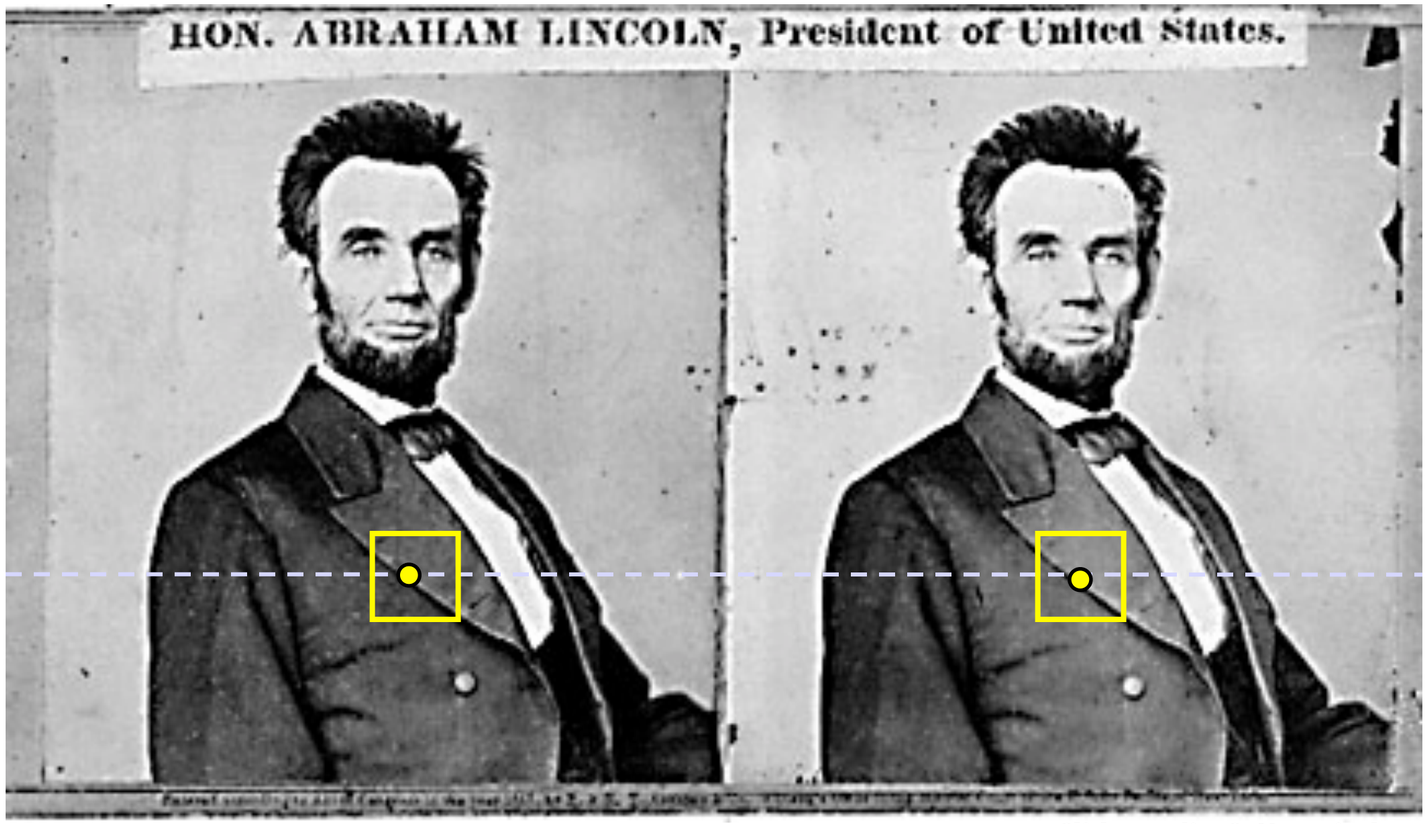
- **Goal: determine pixel correspondence**
  - Corresponding pixels = pairs of pixels that correspond to same scene point



- **Epipolar Constraint**
  - Reduces correspondence problem to 1D search along conjugate epipolar lines
  - Point in left image will lie on line in right image (epipolar line)

Slide credit: S. Narasimhan

Solving correspondence (basic algorithm)



Slide credit: S. Narasimhan

For each epipolar line

For each pixel in the left image

Compare with every pixel on same epipolar line in right image

Pick pixel with minimum match cost

What are assumptions?

Basic improvements: match windows, adaptive size match windows…

- **This should sound familiar given our discussion of image processing algorithms…**
- **Correlation, sum of squared difference (SSD), etc.**

Solving correspondence: robustness challenges

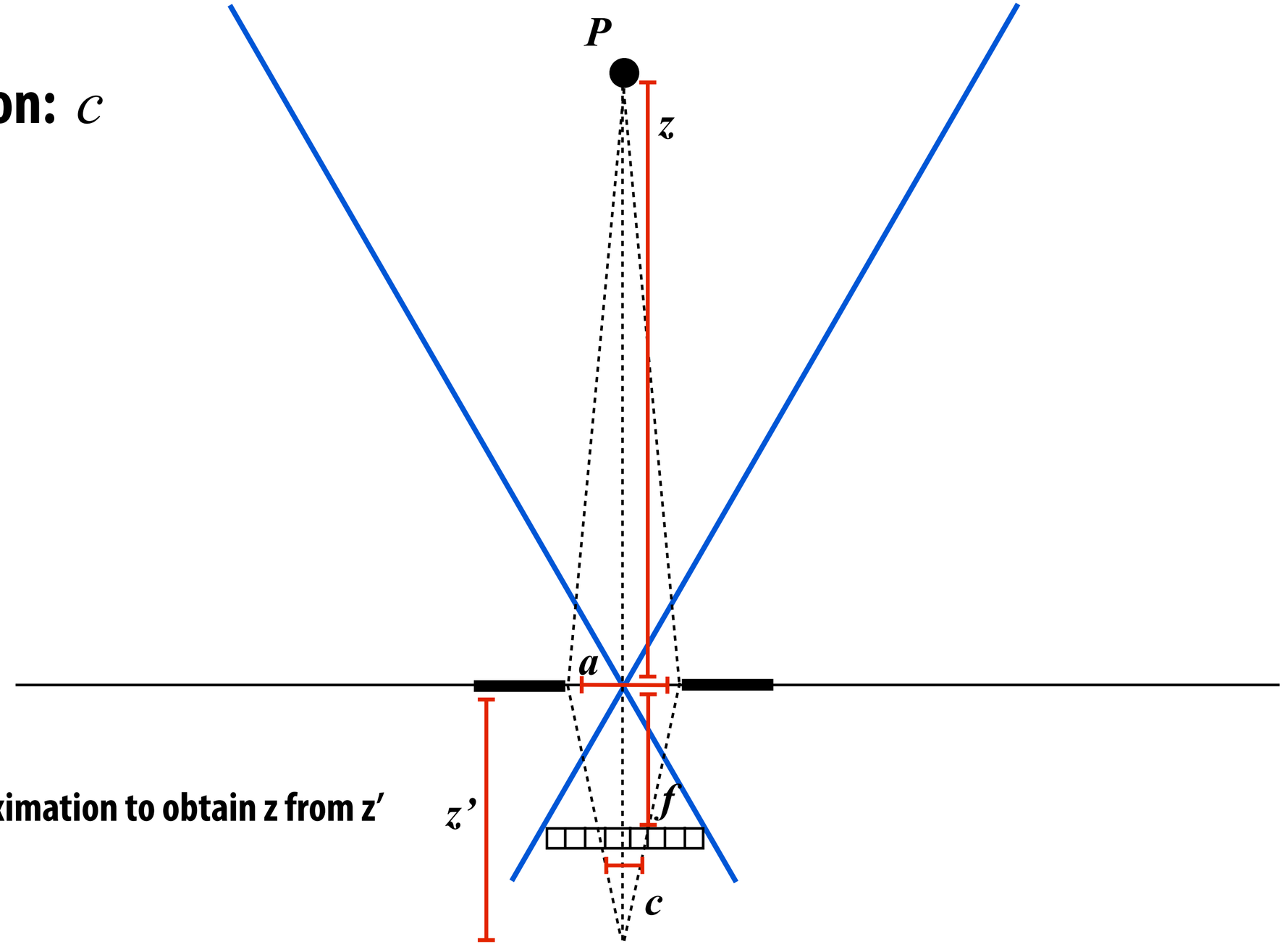
- Scene with no texture (many parts of the scene look the same)
- Non-lambertian surfaces (surface appearance is dependent upon view)
- Pixel pairs may not be present (point on surface is occluded from one view)

Alternative: depth from defocus

Aperture: a

Circle-of-confusion: c

$$\frac{a}{z'} = \frac{c}{z' - f}$$



$$z' = \frac{fa}{a - c}$$

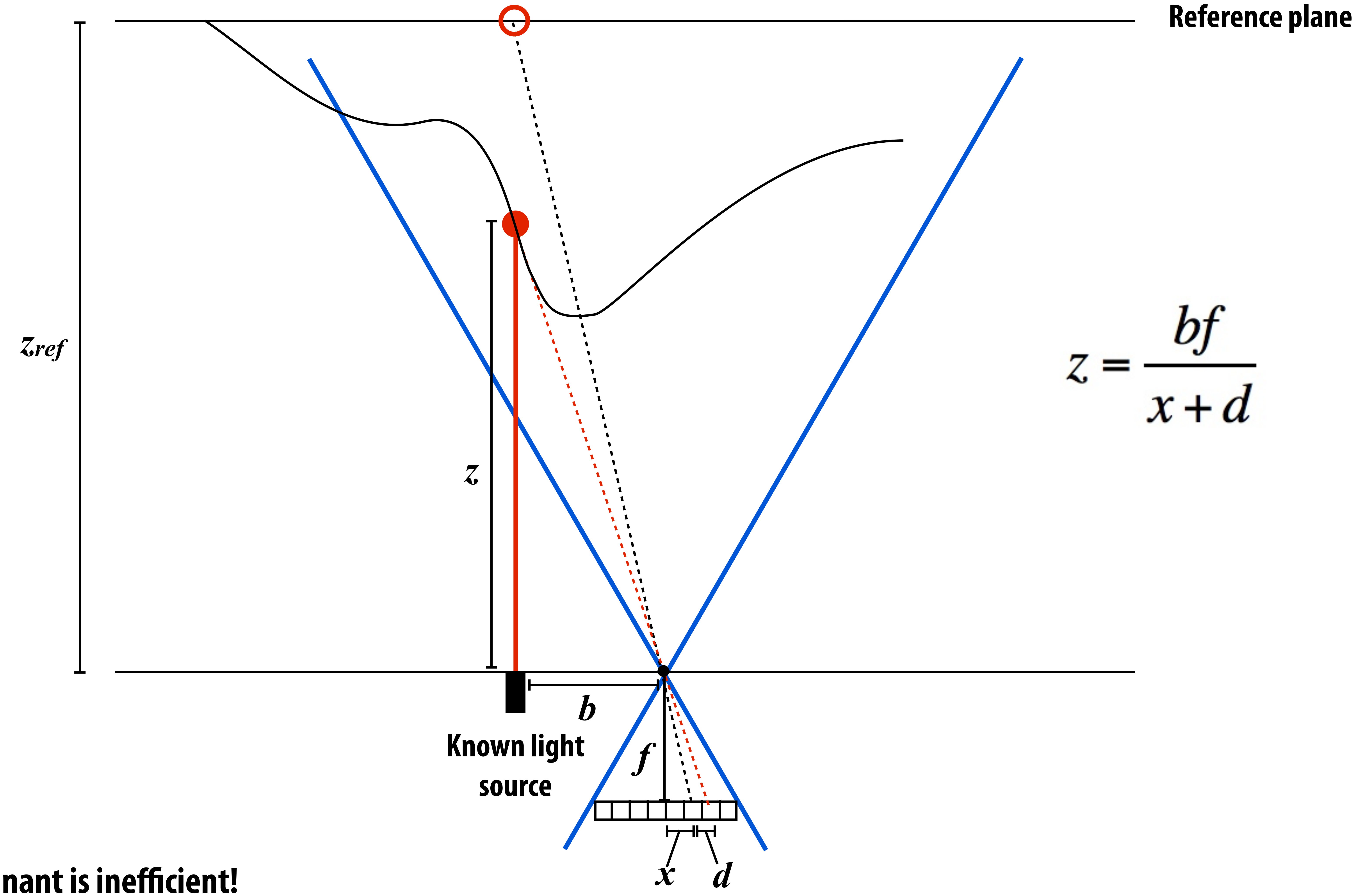
Then use thin lens approximation to obtain z from z′

$$\frac{1}{z'} = \frac{1}{z} + \frac{1}{f}$$

Avoids correspondence problem of depth from disparity, but system must know location of sharp edges in scene to estimate circle of confusion *c*.

Structured light

System: one light source emitting known beam + one camera measuring scene appearance
If the scene is at reference plane, image that will be recorded by camera is known
(correspondence between pixel in recorded image and scene point is known)

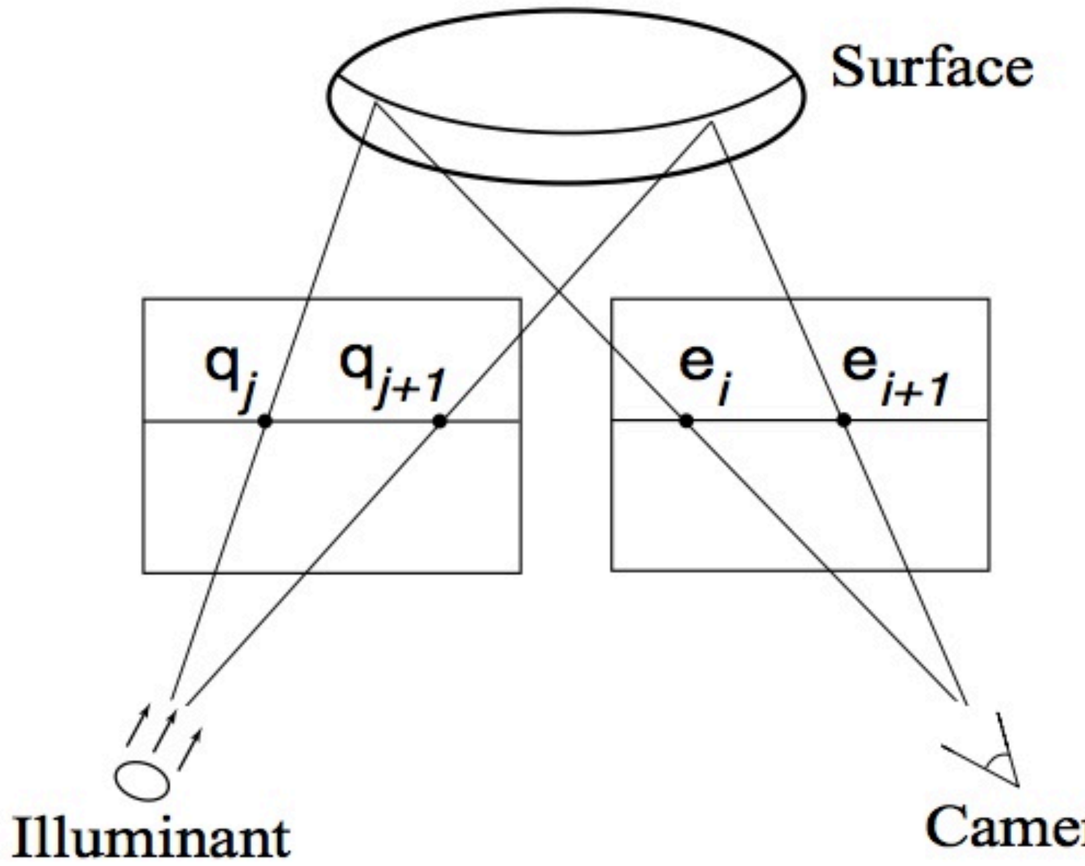


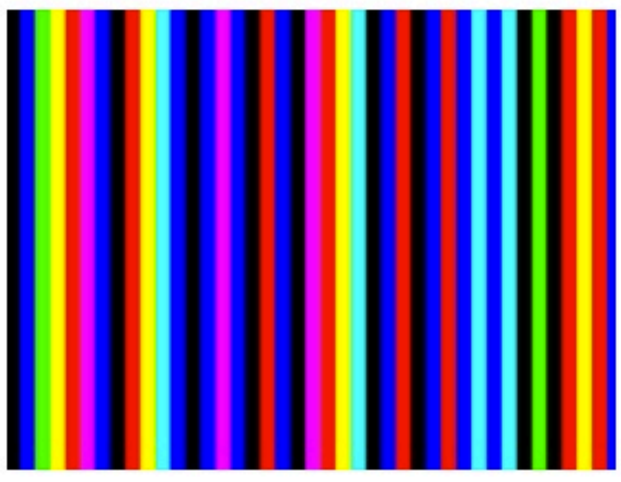
$$z = \frac{bf}{x + d}$$

Single spot illuminant is inefficient!
(must "scan" scene with spot to get depth, so high latency to retrieve a single depth image)

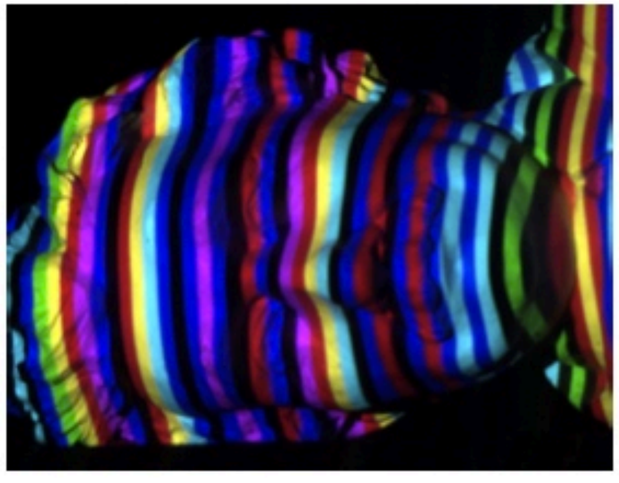
Structured light

Simplify correspondence problem by encoding spatial position in illuminant





Projected light pattern



Camera image

Image: Zhang et al.

Microsoft XBox 360 Kinect

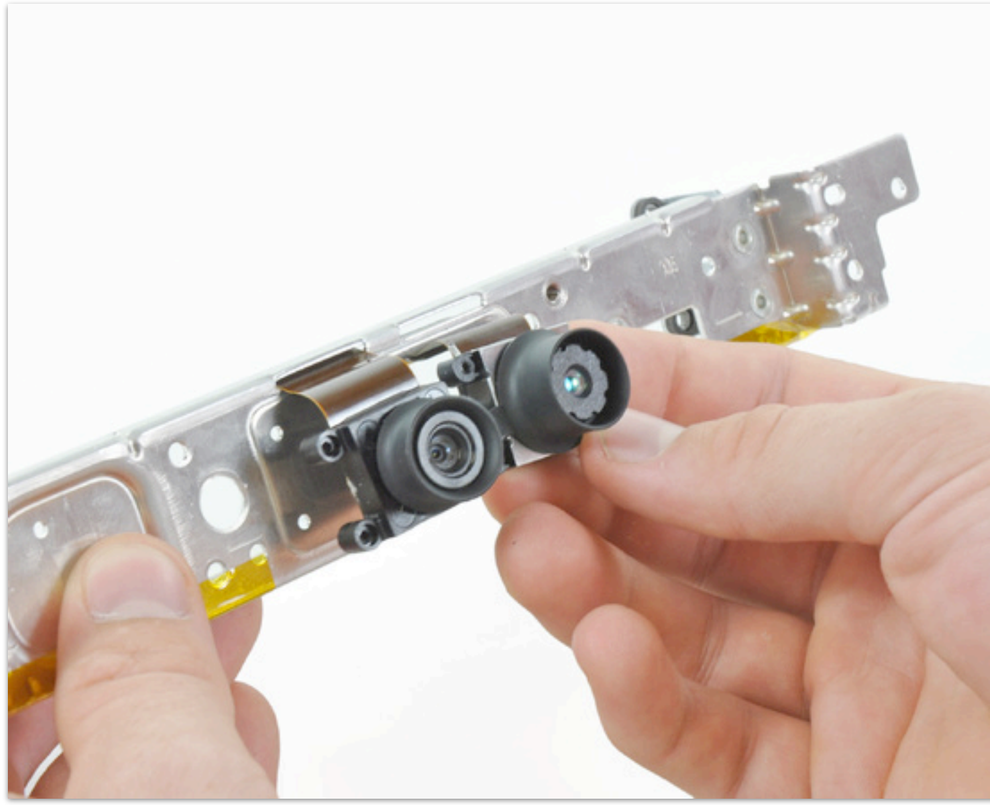


Image credit: iFixIt

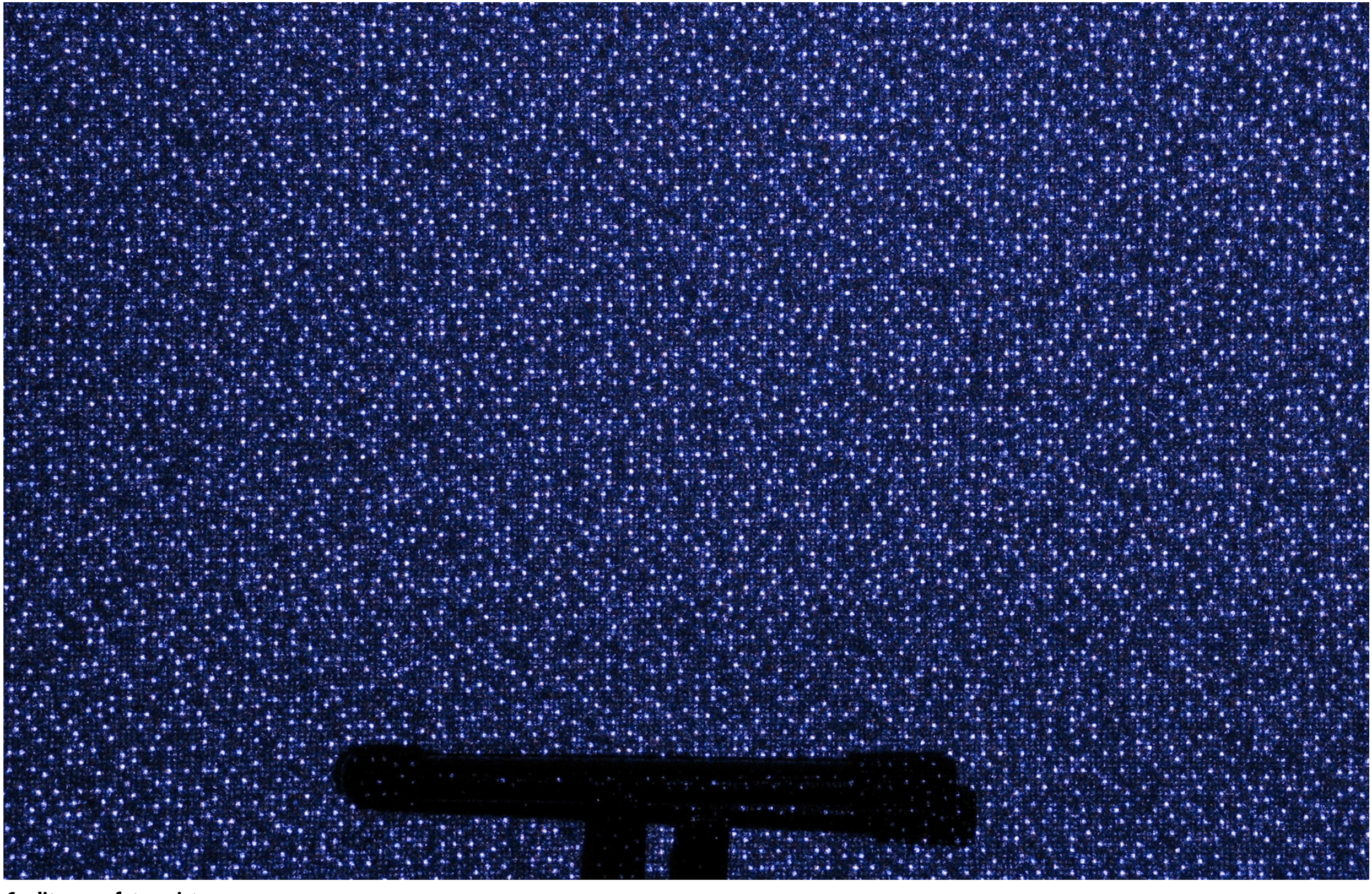
Illuminant
(Infrared Laser + diffuser)

RGB CMOS Sensor
640x480 (w/ Bayer mosaic)

Monochrome Infrared
CMOS Sensor
(Aptina MT9M001)
1280x1024 **

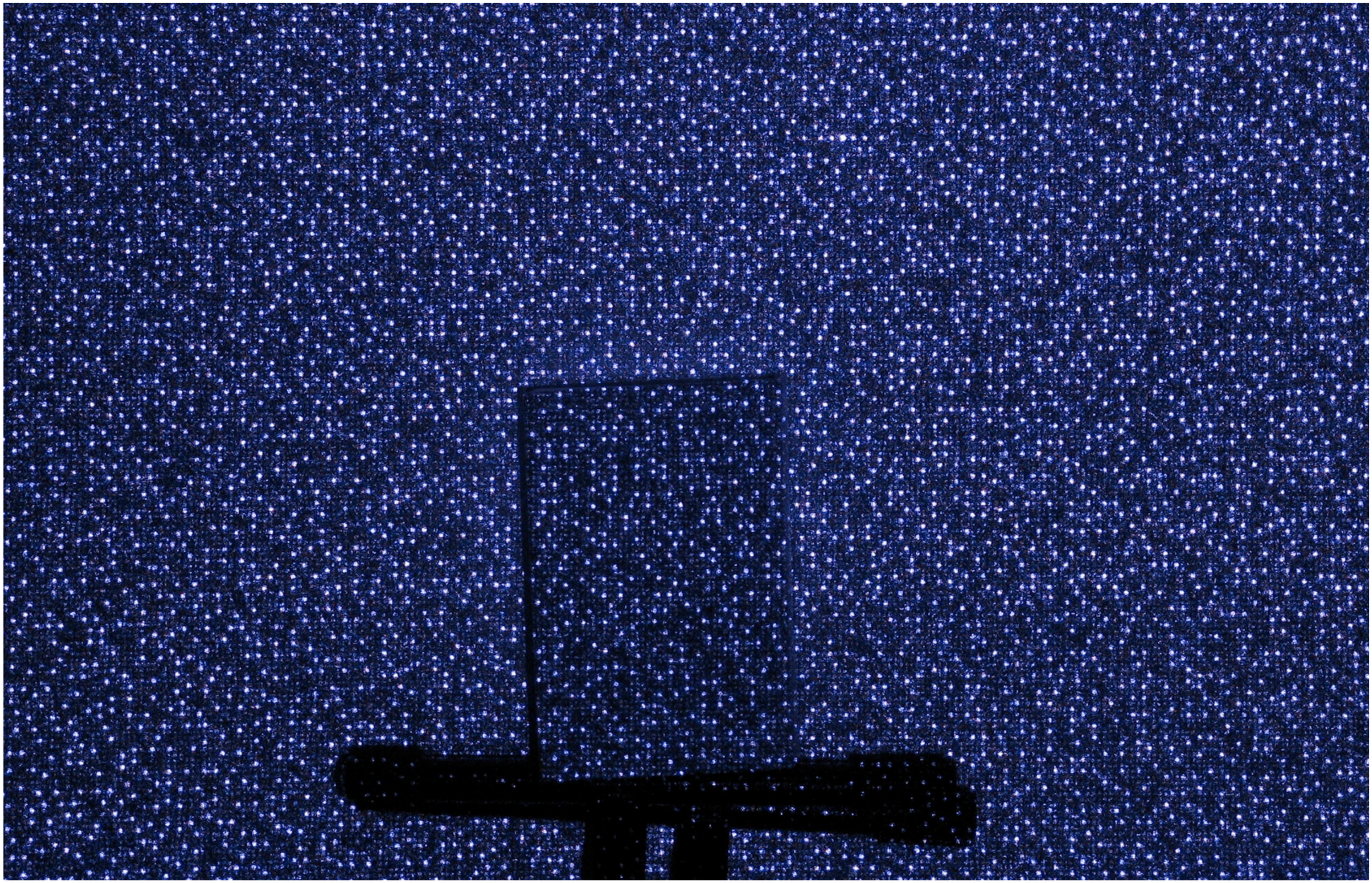
**** Kinect returns 640x480 disparity image, suspect sensor is configured for 2x2 pixel binning down to 640x512, then crop**

Infrared image of Kinect illuminant output



Credit: www.futurepicture.org

Infrared image of Kinect illuminant output



Credit: www.futurepicture.org

Computing disparity for scene

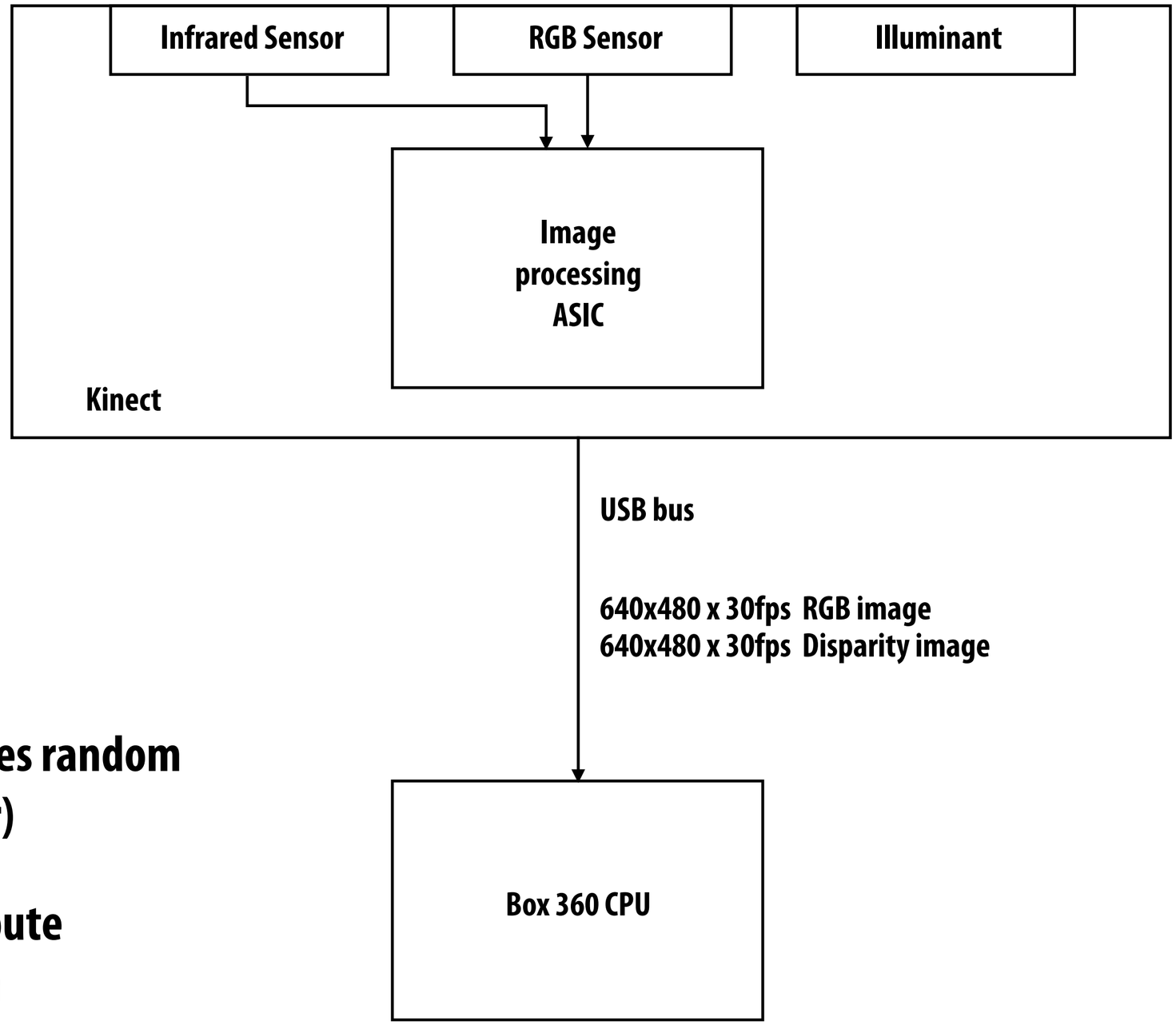
Region-growing algorithm for compute efficiency **
(Assumption: spatial locality likely implies depth locality)

1. Choose output pixels in infrared image, classify as UNKNOWN or SHADOW (based on whether speckle is found)
2. While significantly large percentage of output pixels are UNKNOWN
   - Choose an UNKNOWN pixel.
   - Correlate surrounding MxN pixel window with reference image to compute disparity $D = (dx,dy)$
     (note: search window is a horizontal swath of image, plus some vertical slack)
   - If sufficiently good correlation is found:
     - Mark pixel as a region anchor (it's depth is known)
     - Attempt to grow region around the anchor pixel:
       - Place region anchor in FIFO, mark as ACTIVE
       - While FIFO not empty
       - Extract pixel P from FIFO (known disparity for P is D)
       - Attempt to establish correlations for UNKOWN neighboring pixels P_n of P (left,right,top,bottom neighbors) by searching region given by $P_n + D + (+/-1,+/1)$
       - If correlation is found, mark P_n as ACTIVE, set parent to P, add to FIFO
       - Else, mark P_n as EDGE, set depth to depth of P.

** Source: PrimeSense Patent WO 2007/043036 A1

Kinect block diagram

Disparity calculations performed by PrimeSense ASIC in Kinect, not by XBox 360 CPU



Cheap sensors: ~ 1 MPixel

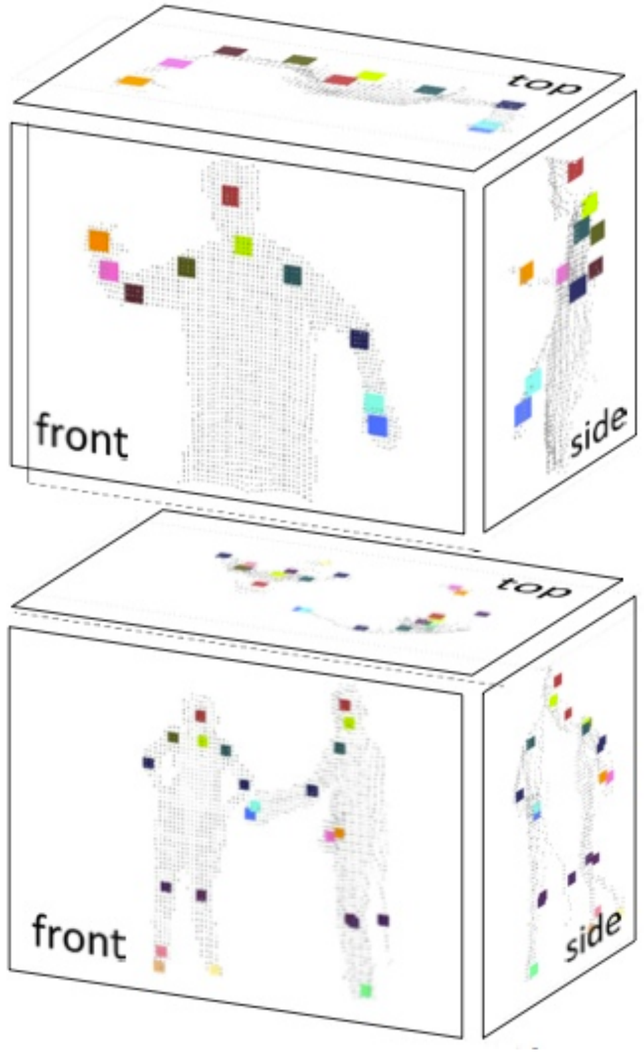
Cheap illuminant: laser + diffuser makes random dot pattern (not a traditional projector)

Custom image-processing ASIC to compute disparity image (scale-invariant region correlation involves non-trivial compute cost)

Extracting the player's skeleton

[Shotton et al. 2011]

(enabling full-body game input)



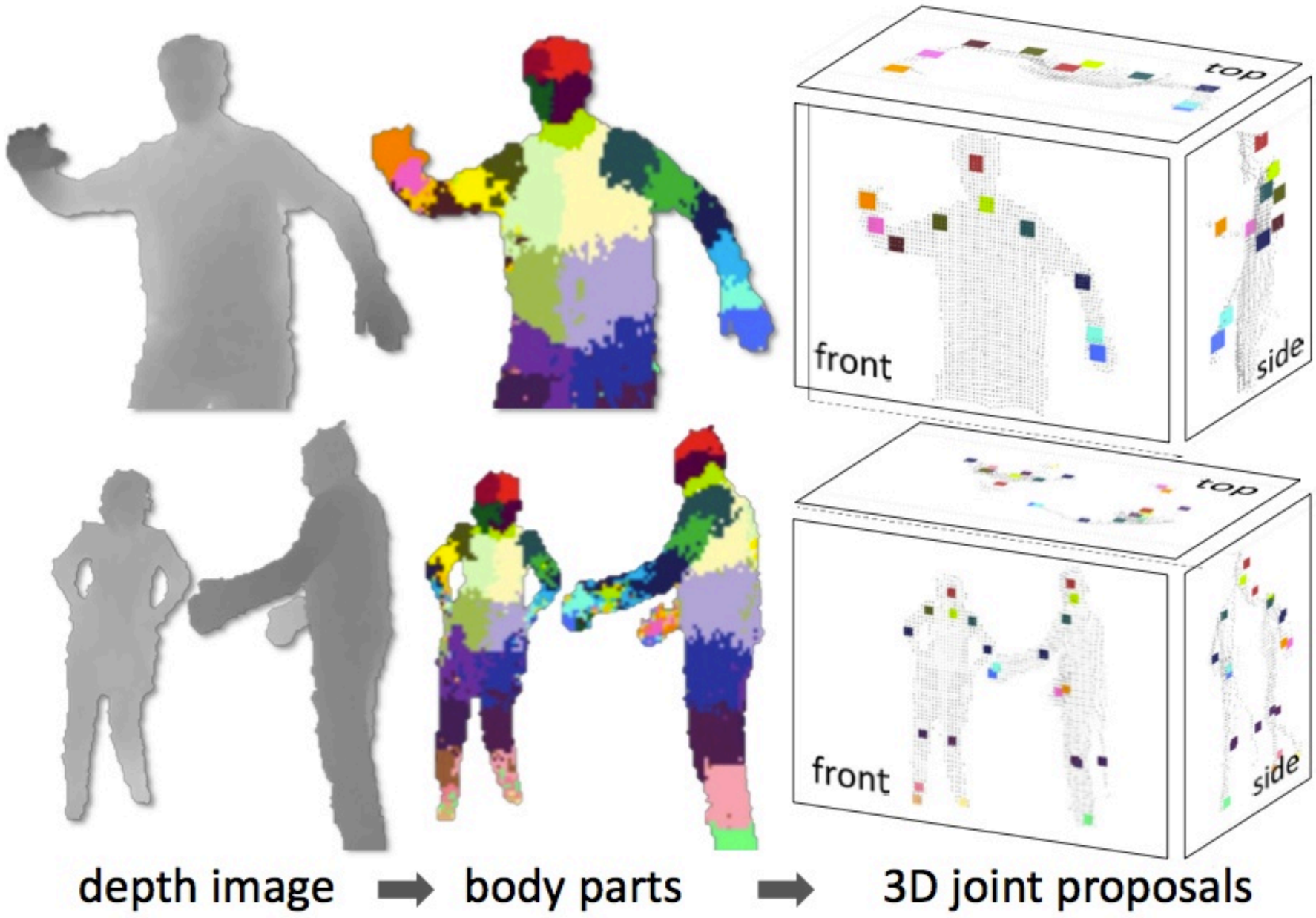
Challenge: how to determine player's position and motion from (noisy) depth images... without consuming a large fraction of the XBox 360's compute capability?

Depth Image

Character Joint Angles

Key idea: classify pixels into body regions

[Shotton et al. 2011]



Shotton et al. represents body with 31 regions

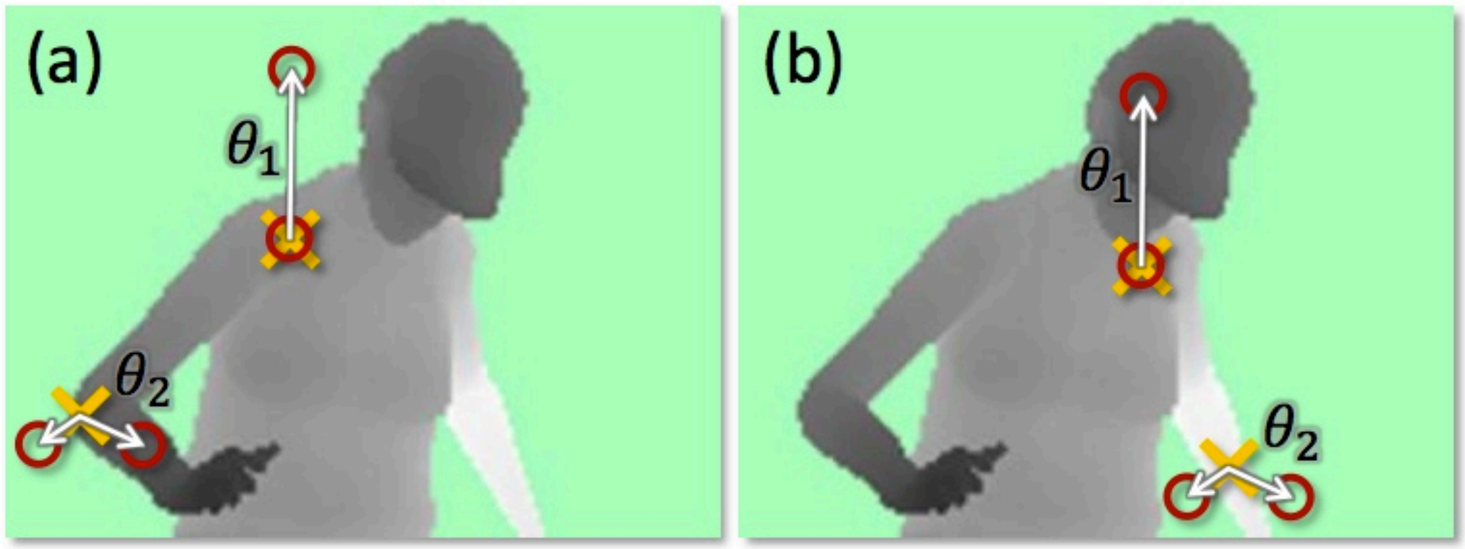
Pixel classification

[Shotton et al. 2011]

For each pixel: compute features from depth image

$$f_\theta(I,\mathrm{X}) = d_I\left(\mathrm{X} + \frac{u}{d_I(\mathrm{X})}\right) + d_I\left(\mathrm{X} + \frac{v}{d_I(\mathrm{X})}\right)$$

Where $\theta = (u,v)$ and $d_I(X)$ is the depth image value at pixel X.



Two example depth features

Features are cheap to compute + can be computed for all pixels in parallel

- Do not depend on velocities: only information from current frame

Classify pixels into body parts using randomized decision forest classifier

- Trained on 100K motion capture poses + database of rendered images as ground truth

Result of classification: $P(c|I,\mathbf{x})$ (probability pixel x in depth image I is body part c)

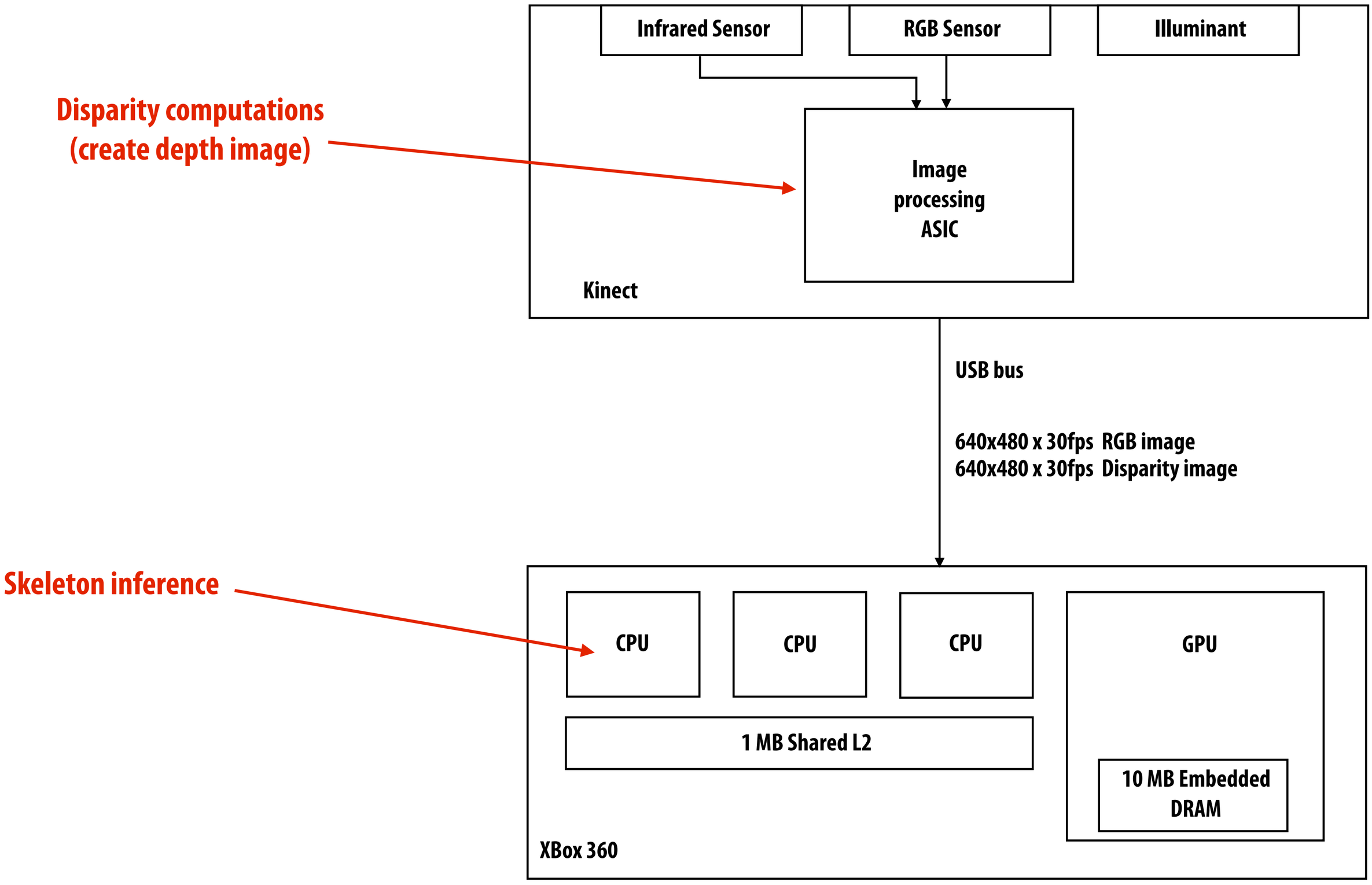
Per-pixel probabilities pooled to compute 3D spatial density function for each body part c
(joint angles inferred from this density)

Performance result

- Real-time skeleton estimation from depth image requires < 10% of Xbox 360 CPU

- XBox GPU-based implementation @ 200Hz (research implementation described in publication, not used in product)
  - Actual XBox 360 product implementation is likely far more efficient

XBox 360 + Kinect system



Xbox 360 Kinect summary

- Hardware = cheap depth sensor + custom image processing ASIC
  - Structured light pattern generated by scattering infrared laser
  - Depth obtained from triangulation (depth from disparity), not time-of-flight
  - Custom ASIC to convert infrared image into depth values (high computational cost is searching for correspondences)

- Interpretation of depth values is performed on CPU
  - Low-cost, data-patallel skeleton estimation made computationally feasible by machine learning approach

Xbox One Sensor

- Time-of-flight sensor (not based on structured light like the original Kinect)
- 1080p depth sensing + wider field-of-view than original Kinect
- "Computer vision" challenges in obtaining high-quality signal:
  - Flying pixels
  - Segmentation
  - Motion blur

Another TOF camera: Creative Depth Camera

Time of flight cameras

- Measure phase offset of light reflected off environment
  - Phase shift proportional to distance from object

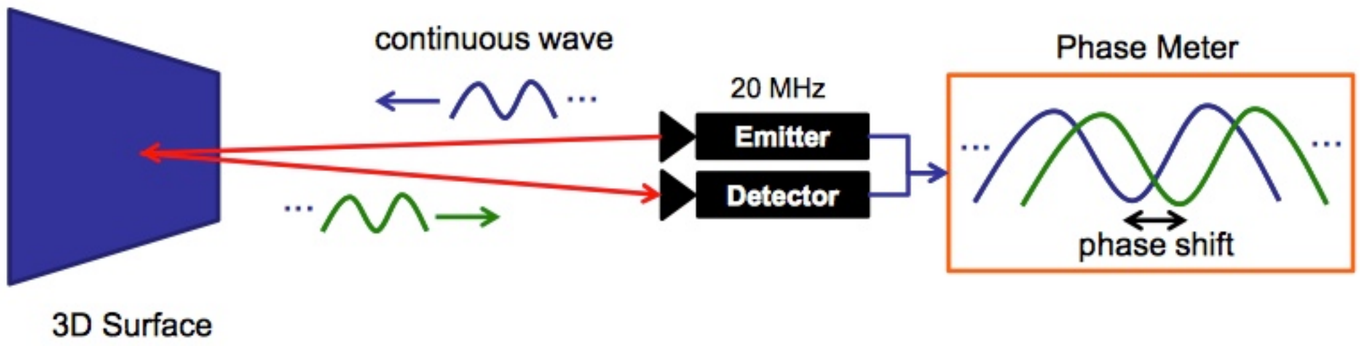
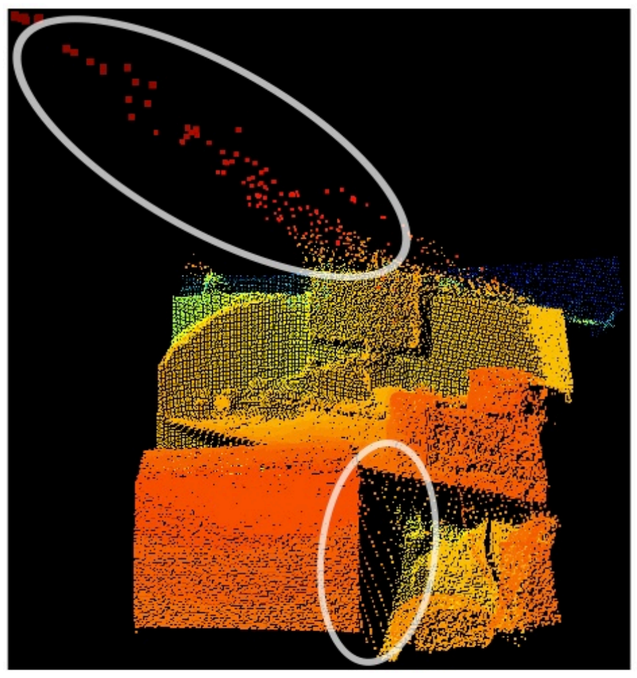


Image credit: V. Castaneda and N. Navab
http://campar.in.tum.de/twiki/pub/Chair/TeachingSs11Kinect/2011-DSensors_LabCourse_Kinect.pdf

- Many "computer vision" challenges to achieving high quality depth estimate
  - Measurement error
  - Motion blur
  - Flying pixels
  - Segmentation

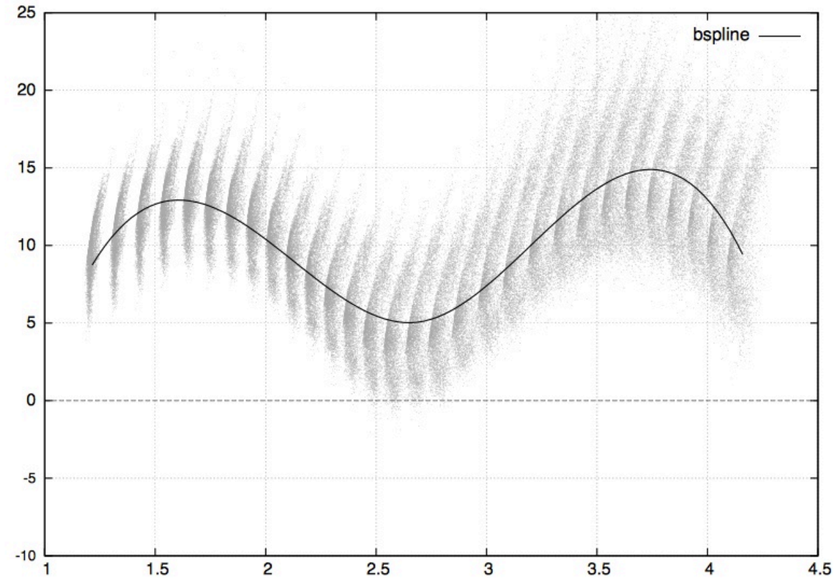


(a) 3D Point Cloud

[Reynolds et al. CVPR 2011]

(b) Intensities

(c) Amplitudes



[A Kolb et al. 2009]

Reading

- *KinectFusion: Real-time 3D Reconstruction and interaction Using a Moving Depth Camera*. S. Izadi et al. UIST 11
  - Note: may wish to Google ICP algorithm (iterative closest point)