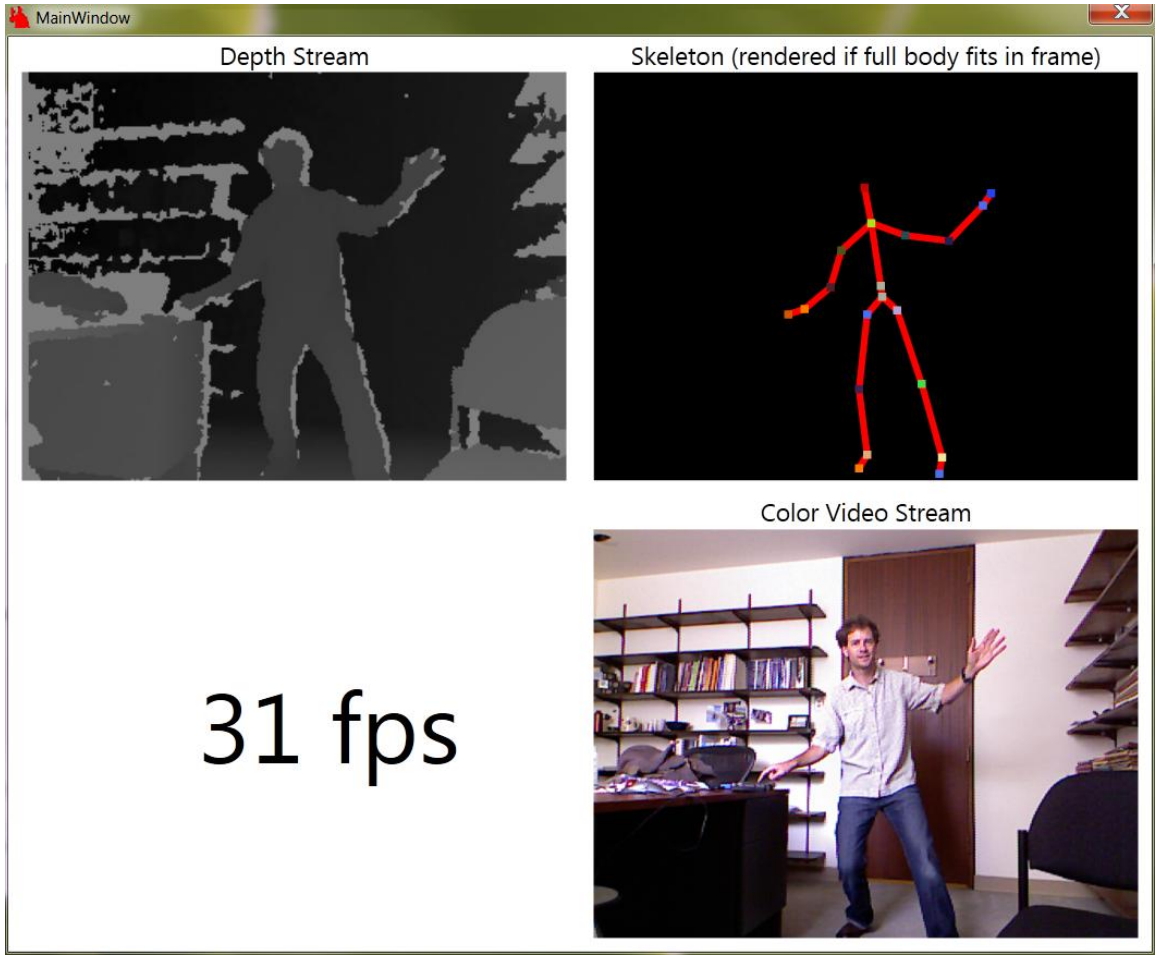
How does the Kinect work?

John MacCormick

Xbox demo

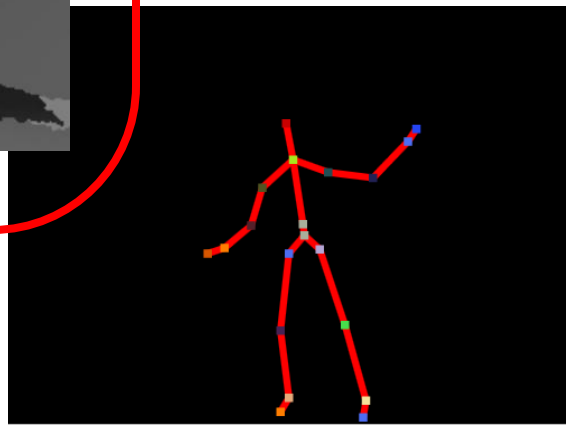
Laptop demo


MainWindow
Depth Stream
Skeleton (rendered if full body fits in frame)
Color Video Stream
31 fps

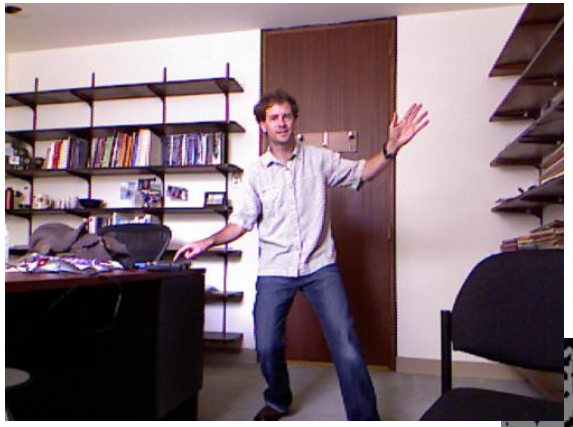
The Kinect uses *structured light* and *machine learning*

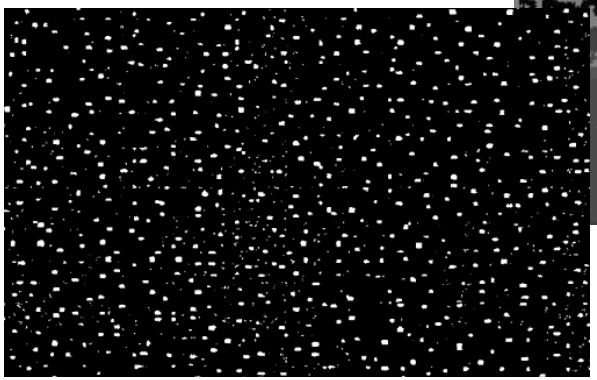
- Inferring body position is a two-stage process: first compute a depth map (using structured light), then infer body position (using machine learning)
- The results are great!
- The system uses many college-level math concepts, and demonstrates the remarkable advances in computer vision in the last 20 years

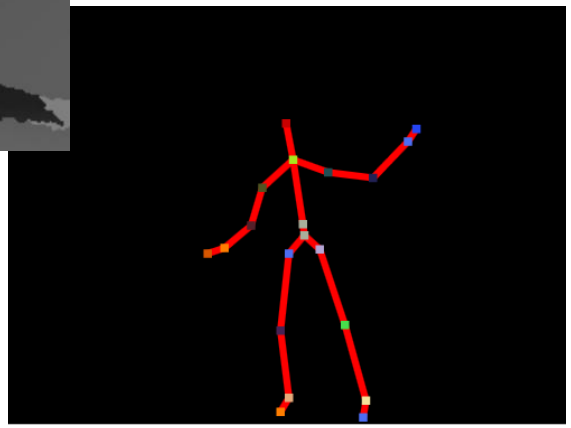
Inferring body position is a two-stage process: first compute a depth map, then infer body position



Inferring body position is a two-stage process: first compute a depth map, then infer body position







Stage 1: The depth map is constructed by analyzing a speckle pattern of infrared laser light

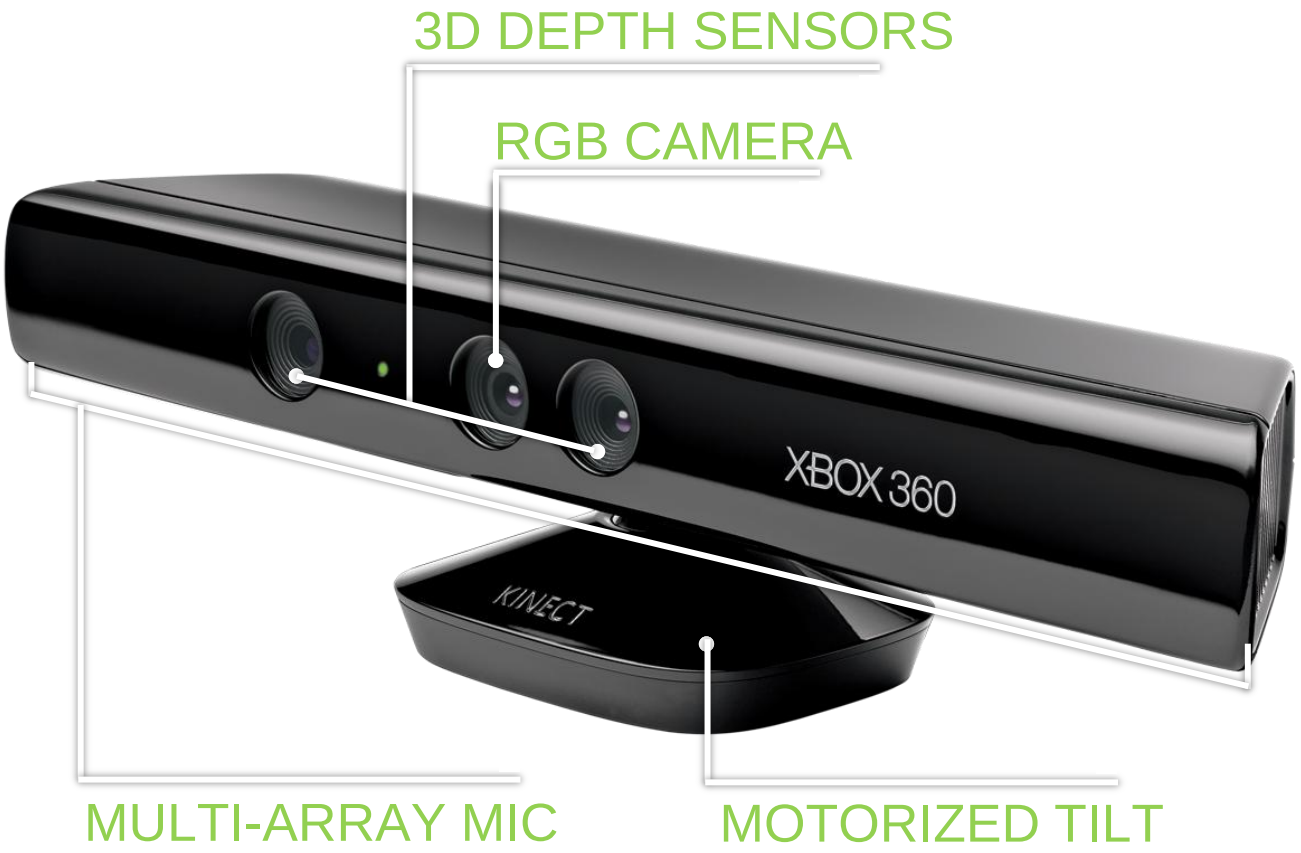


Figure copied from Kinect for Windows SDK Quickstarts

Stage 1: The depth map is constructed by analyzing a speckle pattern of infrared laser light

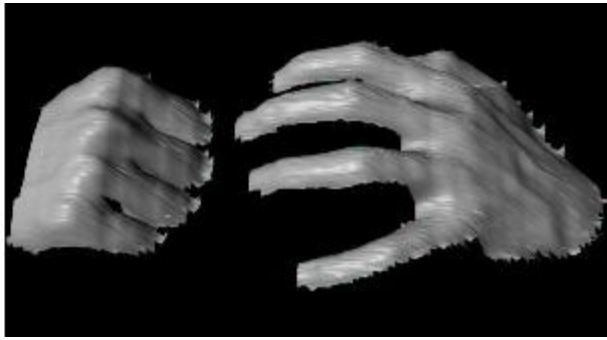
Remarks:

- Microsoft licensed this technology from a company called PrimeSense
- The depth computation is all done by the PrimeSense hardware built into Kinect
- Details are not publicly available; this description is speculative (based mostly on PrimeSense patent applications) and may be wrong

The technique of analyzing a known pattern is called *structured light*

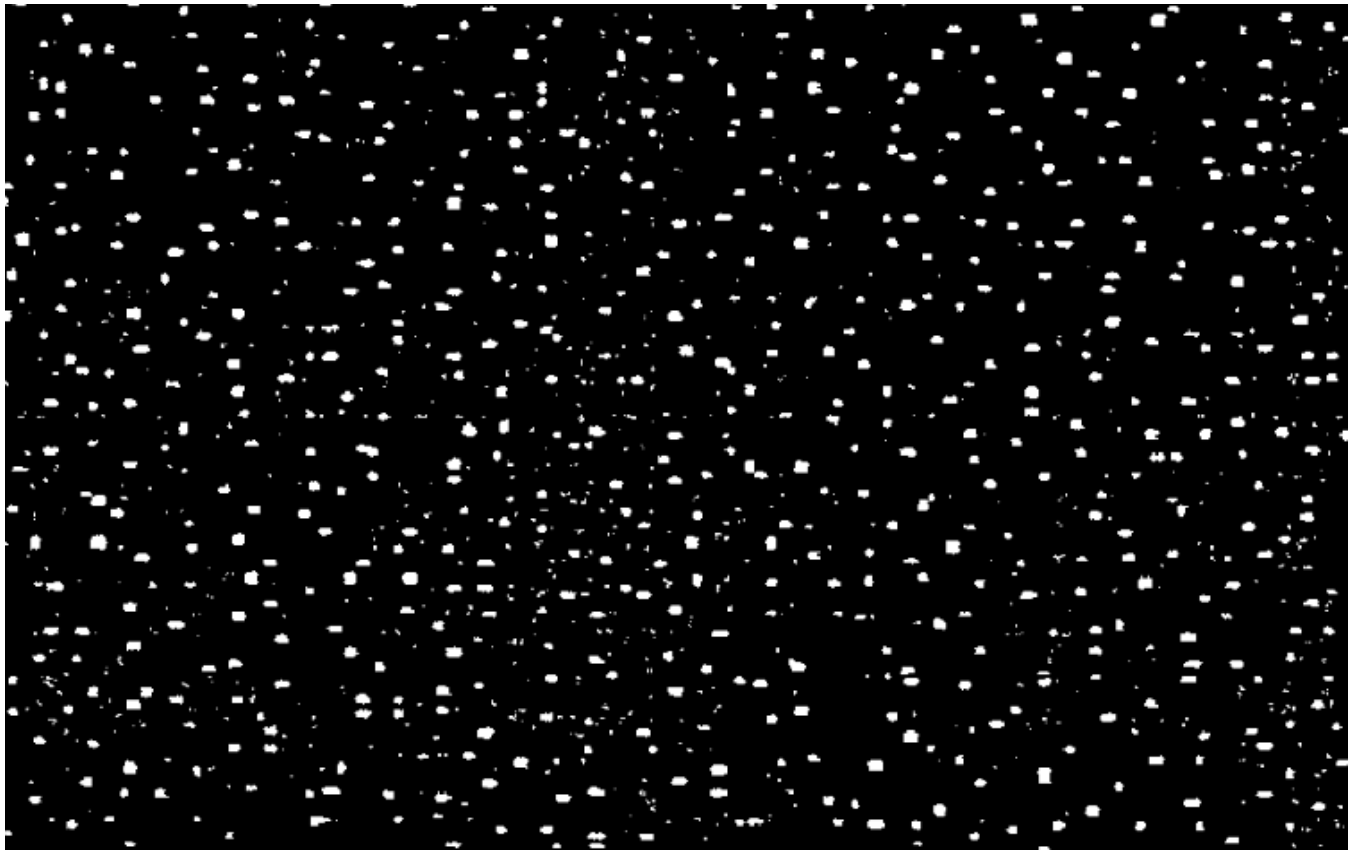
Structured light general principle:

project a known pattern onto the scene and infer depth from the deformation of that pattern



Zhang et al, 3DPVT (2002)

The Kinect uses infrared laser light, with a speckle pattern

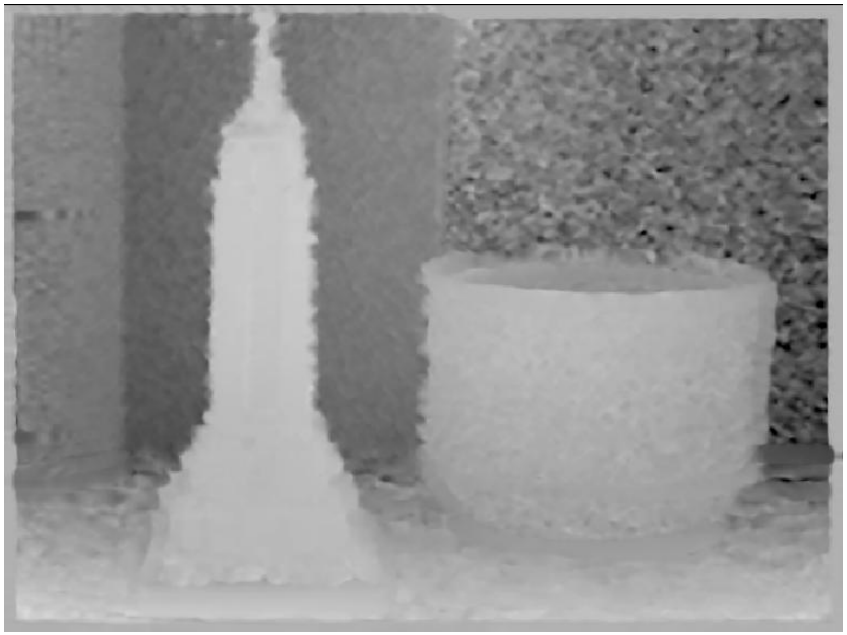


Shpunt et al, PrimeSense patent application US 2008/0106746

Stage 1: The depth map is constructed by analyzing a speckle pattern of infrared laser light

- The Kinect uses an infrared projector and sensor; it does not use its RGB camera for depth computation
- The technique of analyzing a known pattern is called *structured light*
- The Kinect combines structured light with two classic computer vision techniques: depth from focus, and depth from stereo

Depth from focus uses the principle that stuff that is more blurry is further away

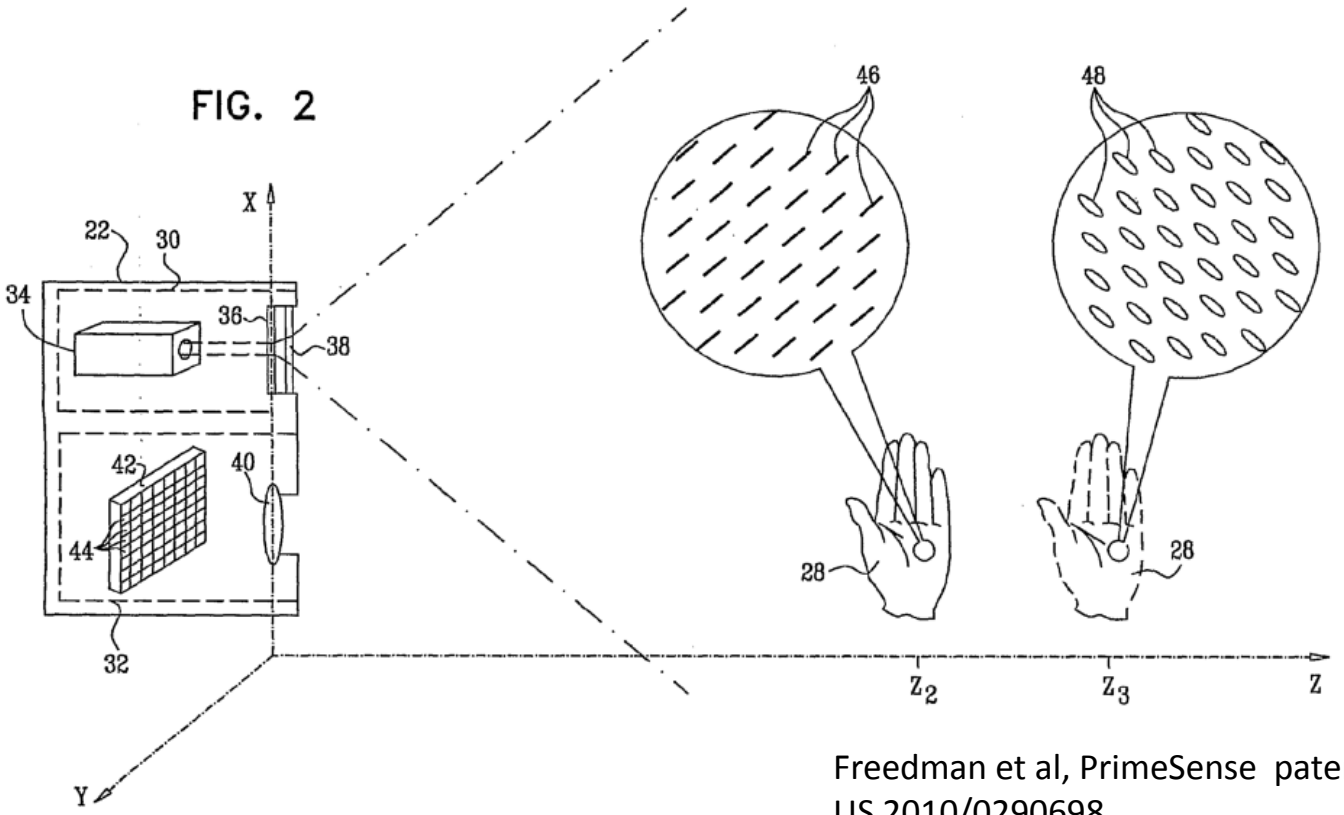


Watanabe and Nayar, IJCV 27(3), 1998

Depth from focus uses the principle that stuff that is more blurry is further away

- The Kinect dramatically improves the accuracy of traditional depth from focus
- The Kinect uses a special (“astigmatic”) lens with different focal length in x- and y-directions
- A projected circle then becomes an ellipse whose orientation depends on depth

The astigmatic lens causes a projected circle to become an ellipse whose orientation depends on depth

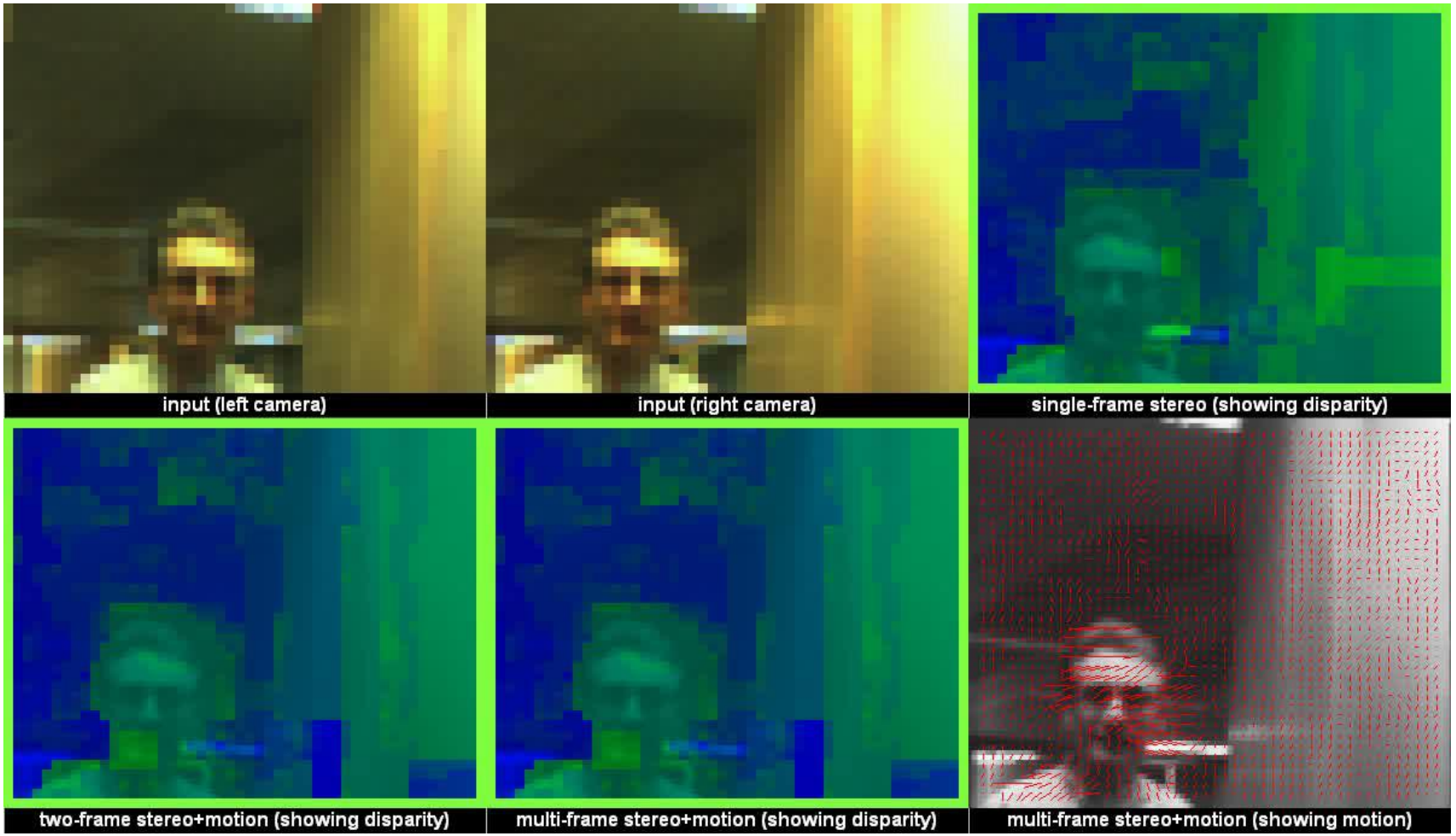


Freedman et al, PrimeSense patent application US 2010/0290698

Depth from stereo uses *parallax*

- If you look at the scene from another angle, stuff that is close gets shifted to the side more than stuff that is far away
- The Kinect analyzes the shift of the speckle pattern by projecting from one location and observing from another

Depth from stereo uses *parallax*



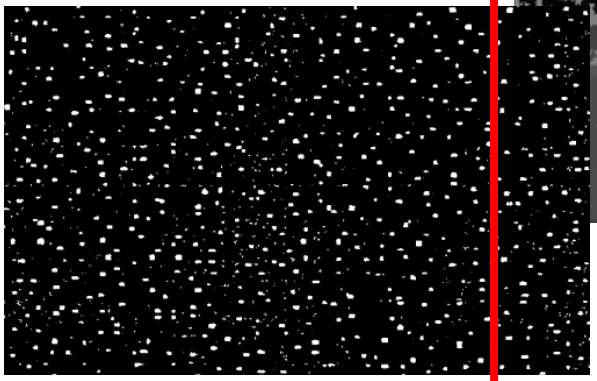
Isard and M, ACCV (2006)

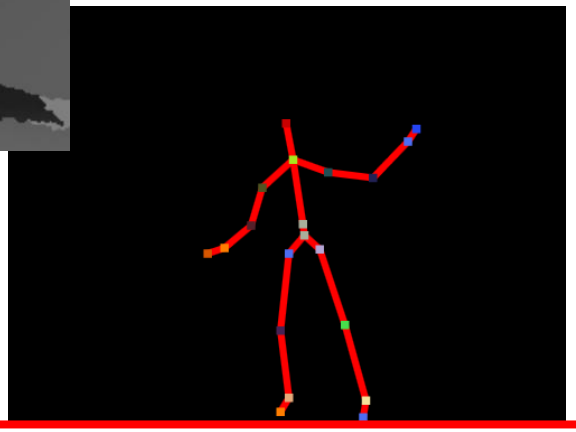
Inferring body position is a two-stage process: first compute a depth map, then infer body position

- Stage 1: The depth map is constructed by analyzing a speckle pattern of infrared laser light
- Stage 2: Body parts are inferred using a randomized decision forest, learned from over 1 million training examples

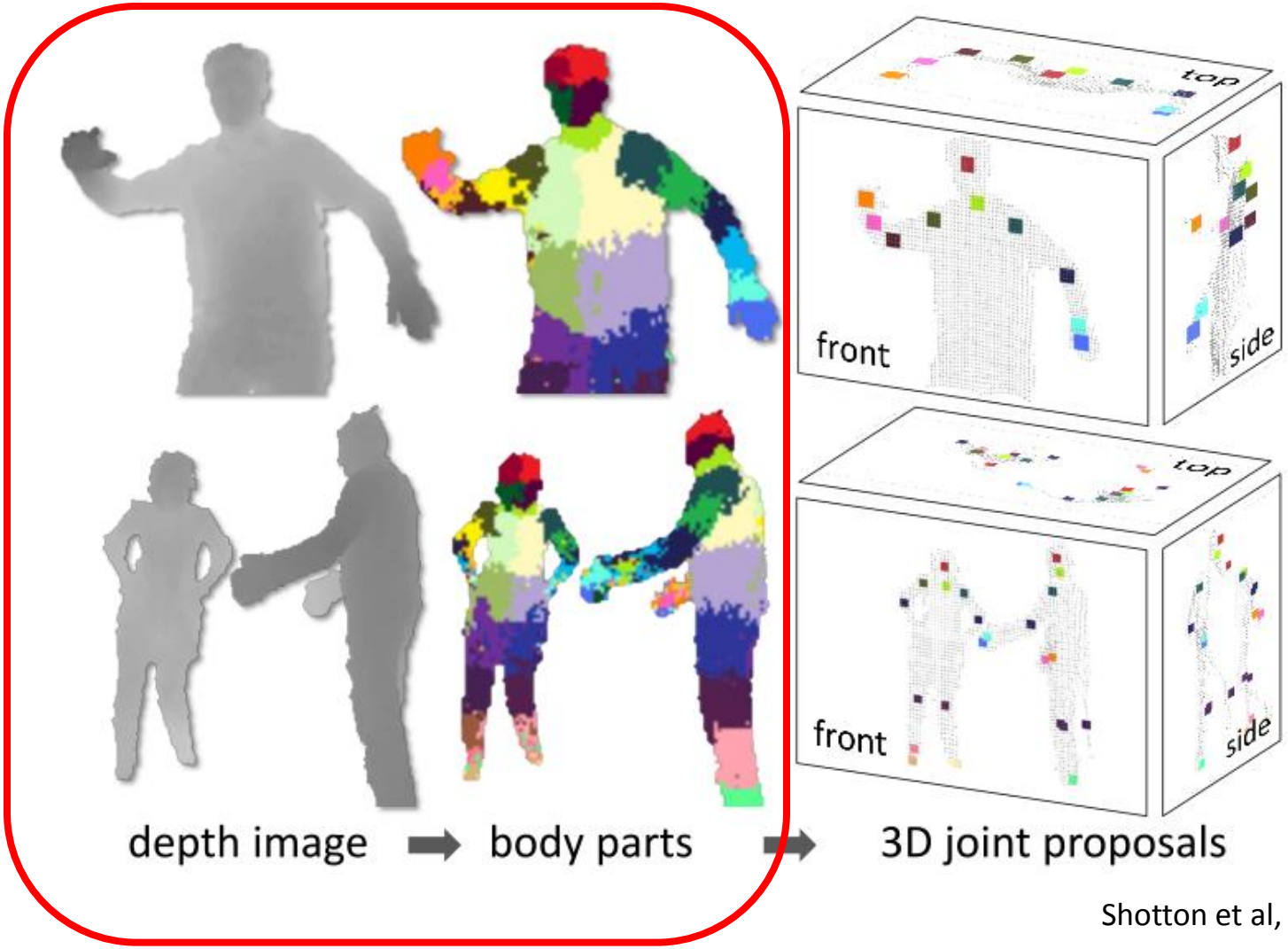
Note: details of Stage 2 were published in Shotton et al (CVPR 2011)

Inferring body position is a two-stage process: first compute a depth map, then infer body position





Stage 2 has 2 substages (use intermediate “body parts” representation)



Shotton et al, CVPR(2011)

Stage 2.1 starts with 100,000 depth images with known skeletons (from a motion capture system)

Shotton et al, CVPR(2011)

For each real image, render dozens more using computer graphics techniques

- Use computer graphics to render all sequences for 15 different body types, and vary several other parameters
- Thus obtain over a million training examples

For each real image, render dozens more using computer graphics techniques

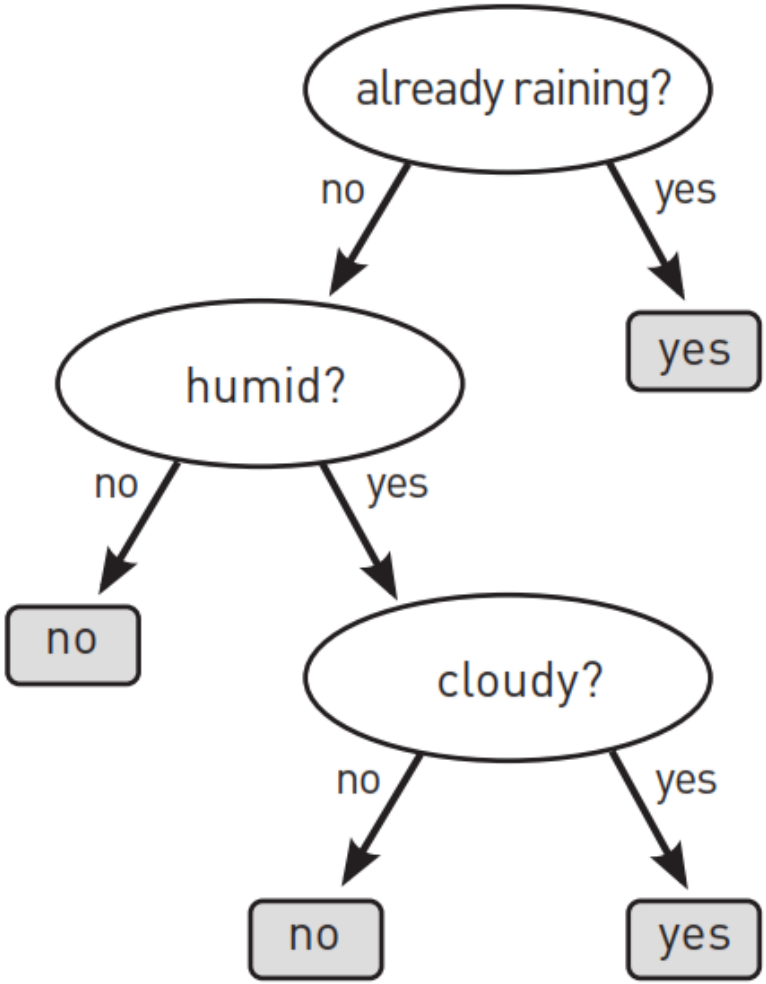
Shotton et al, CVPR(2011)

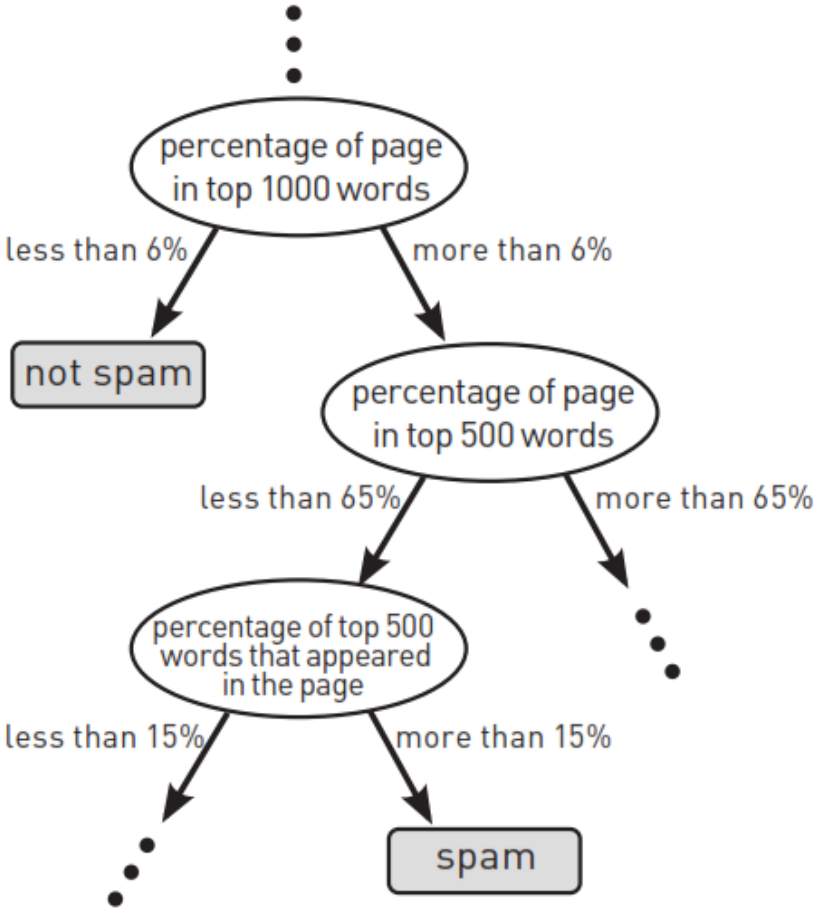
Stage 2.1 transforms depth image to body part image

- Start with 100,000 depth images with known skeleton (from a motion capture system)
- For each real image, render dozens more using computer graphics techniques.
- Learn a *randomized decision forest*, mapping depth images to body parts

A randomized decision forest is a more sophisticated version of the classic *decision tree*

A decision tree is like a pre-planned game of "twenty questions"

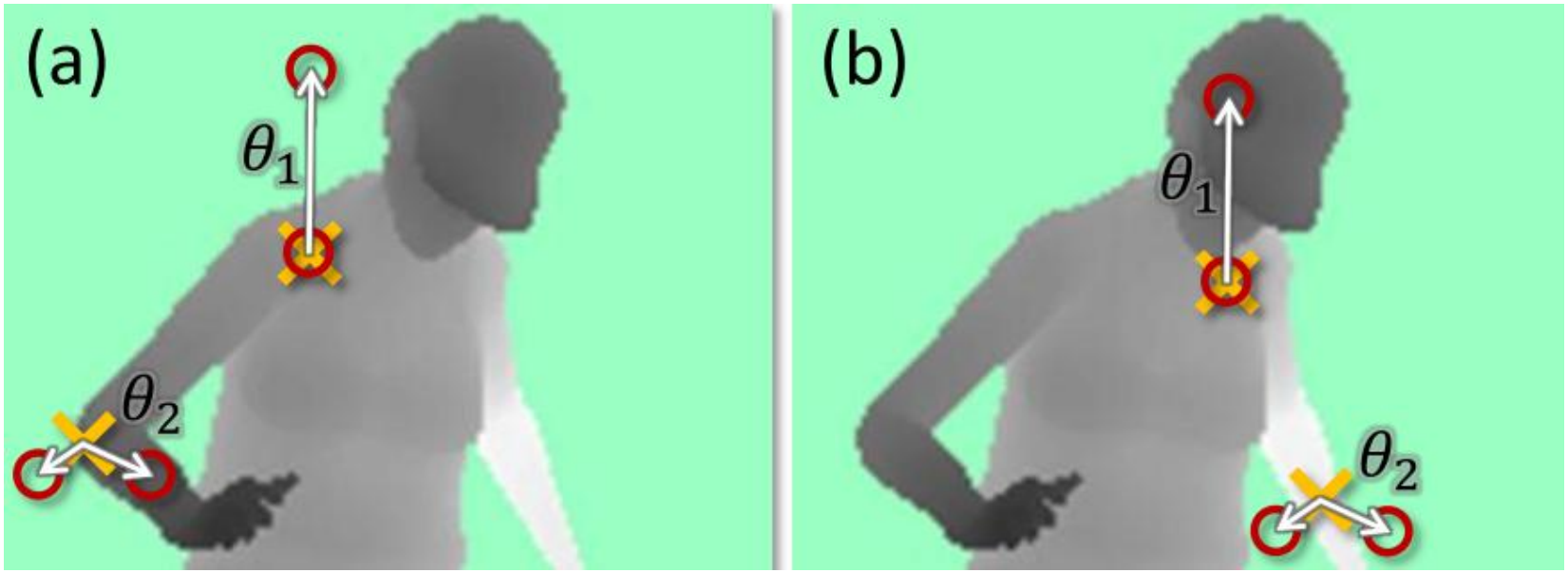




Ntoulas et al, WWW (2006)

What kind of "questions" can the Kinect ask in its twenty questions?

- Simplified version:
  - "is the pixel at *that* offset in the background?"
- Real version:
  - "how does the (normalized) depth at *that* pixel compare to *this* pixel?" [see Shotton et al, equation 1]



Shotton et al, CVPR(2011)

To learn a decision tree, you choose as the next question the one that is most "useful" on (the relevant part of) the training data

- E.g. for umbrella tree, is "raining?" or "cloudy?" more useful?
- In practice, "useful" = *information gain G* (which is derived from *entropy H*):

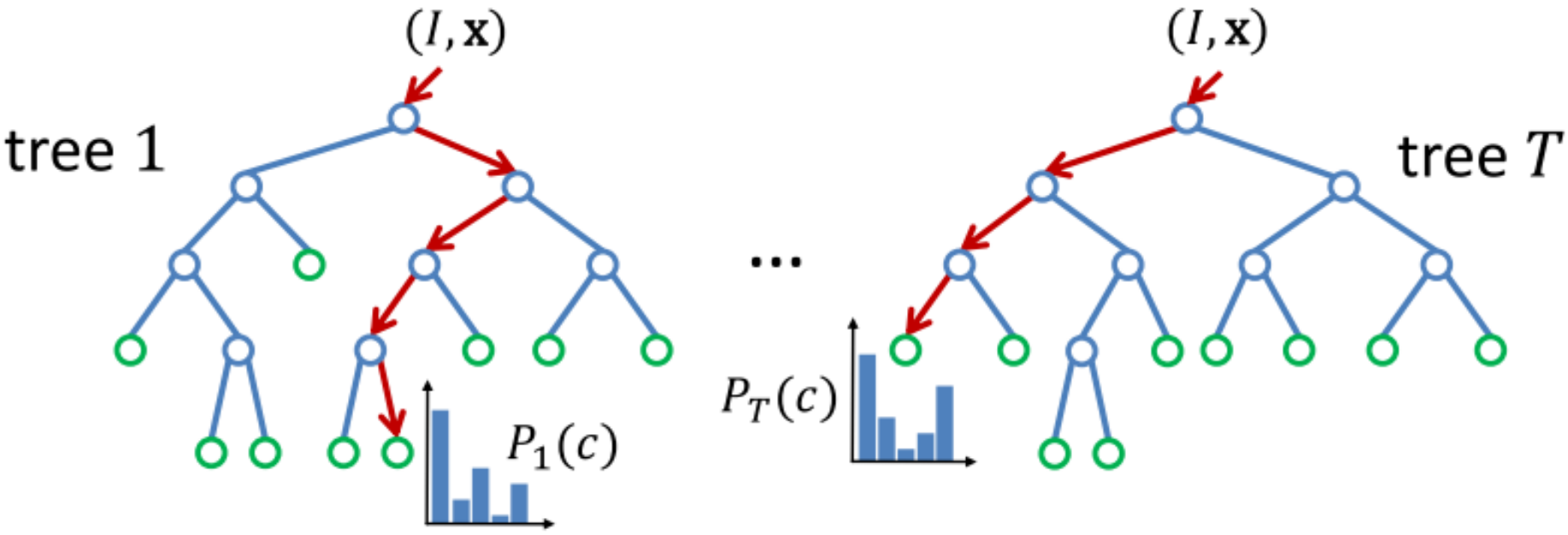
$$G(\phi) \quad = \quad H(Q) - \sum_{s \in \{\mathrm{l},\mathrm{r}\}} \frac{|Q_s(\phi)|}{|Q|} H(Q_s(\phi))$$

Shotton et al, CVPR(2011)

Kinect actually uses a *randomized* decision *forest*

- Randomized:
  - Too many possible questions, so use a random selection of 2000 questions each time
- Forest:
  - learn multiple trees
  - to classify, add outputs of the trees
  - outputs are actually probability distributions, not single decisions

Kinect actually uses a *randomized* decision *forest*



Shotton et al, CVPR(2011)

Learning the Kinect decision forest requires 24,000 CPU-hours, but takes only a day using hundreds of computers simultaneously

"To keep the training times down we employ a distributed implementation. Training 3 trees to depth 20 from 1 million images takes about a day on a 1000 core cluster."

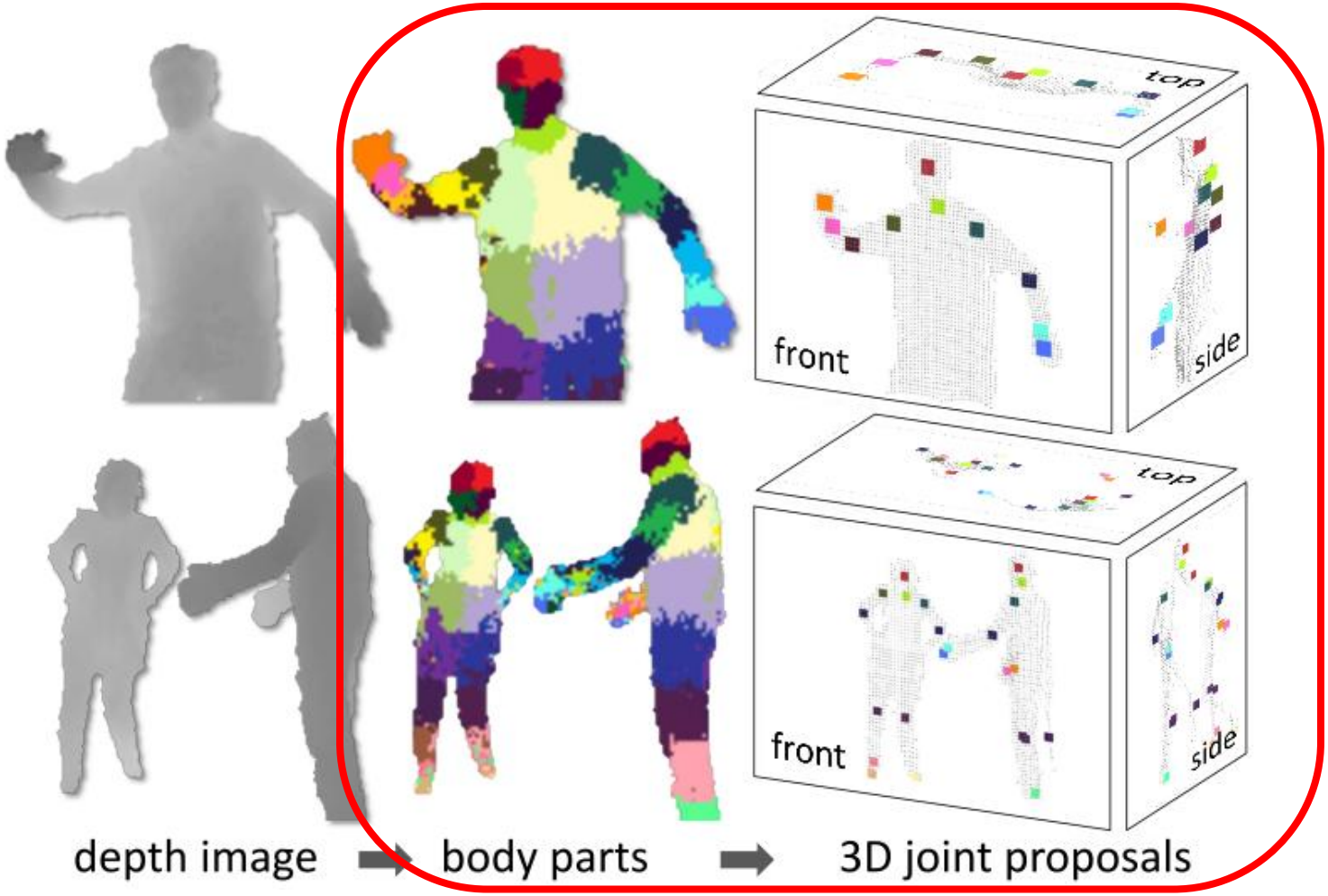
—Shotton et al, CVPR(2011)

Learning the Kinect decision forest requires 24,000 CPU-hours, but takes only a day using hundreds of computers simultaneously

"To keep the training times down we employ a distributed implementation. Training 3 trees to depth 20 from 1 million images takes about a day on a 1000 core cluster."

—Shotton et al, CVPR(2011)

Stage 2 has 2 substages (use intermediate “body parts” representation)

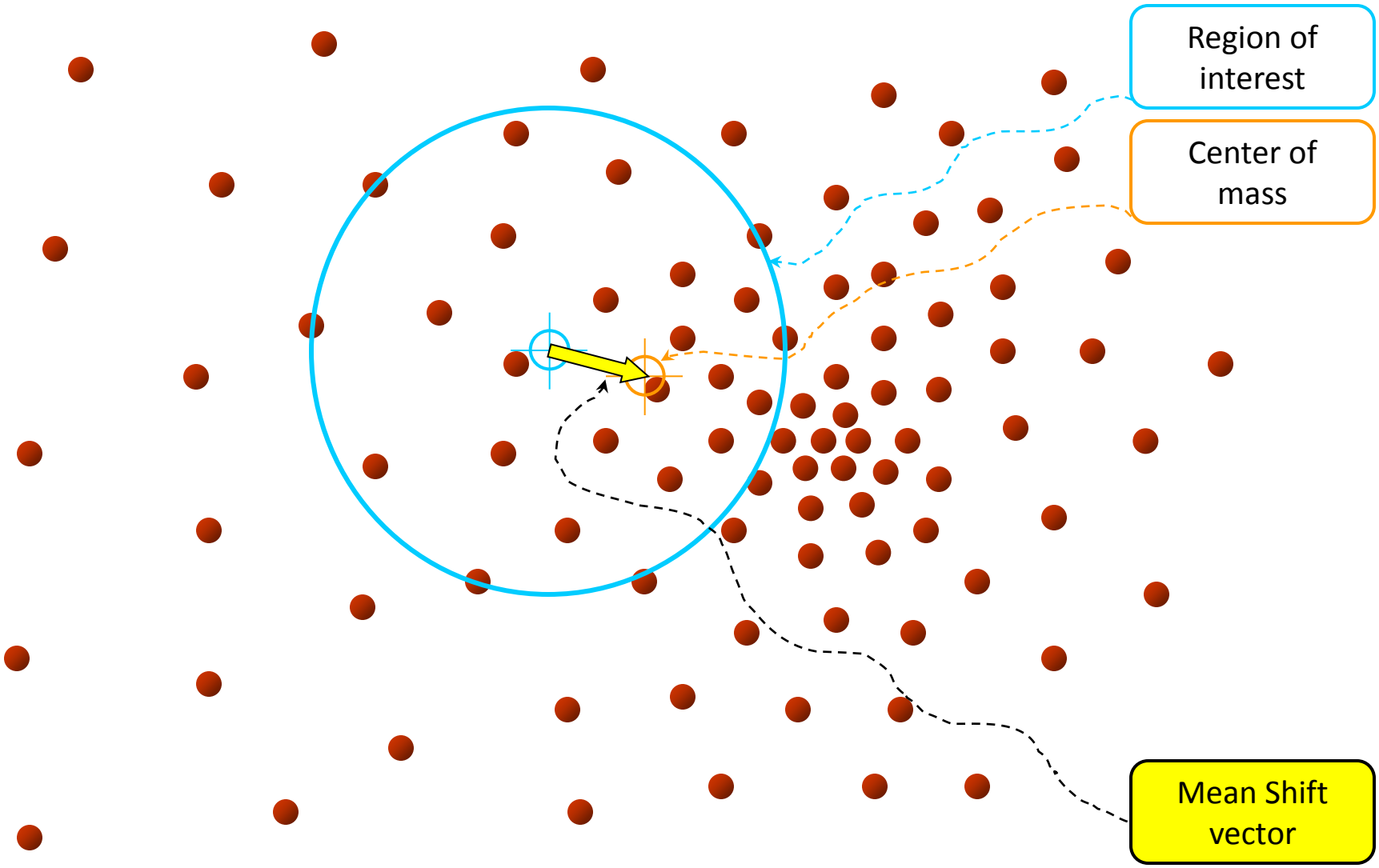


Shotton et al, CVPR(2011)

Stage 2.2 transforms the body part image into a skeleton

- The *mean shift* algorithm is used to robustly compute modes of probability distributions
- Mean shift is simple, fast, and effective

Intuitive Description

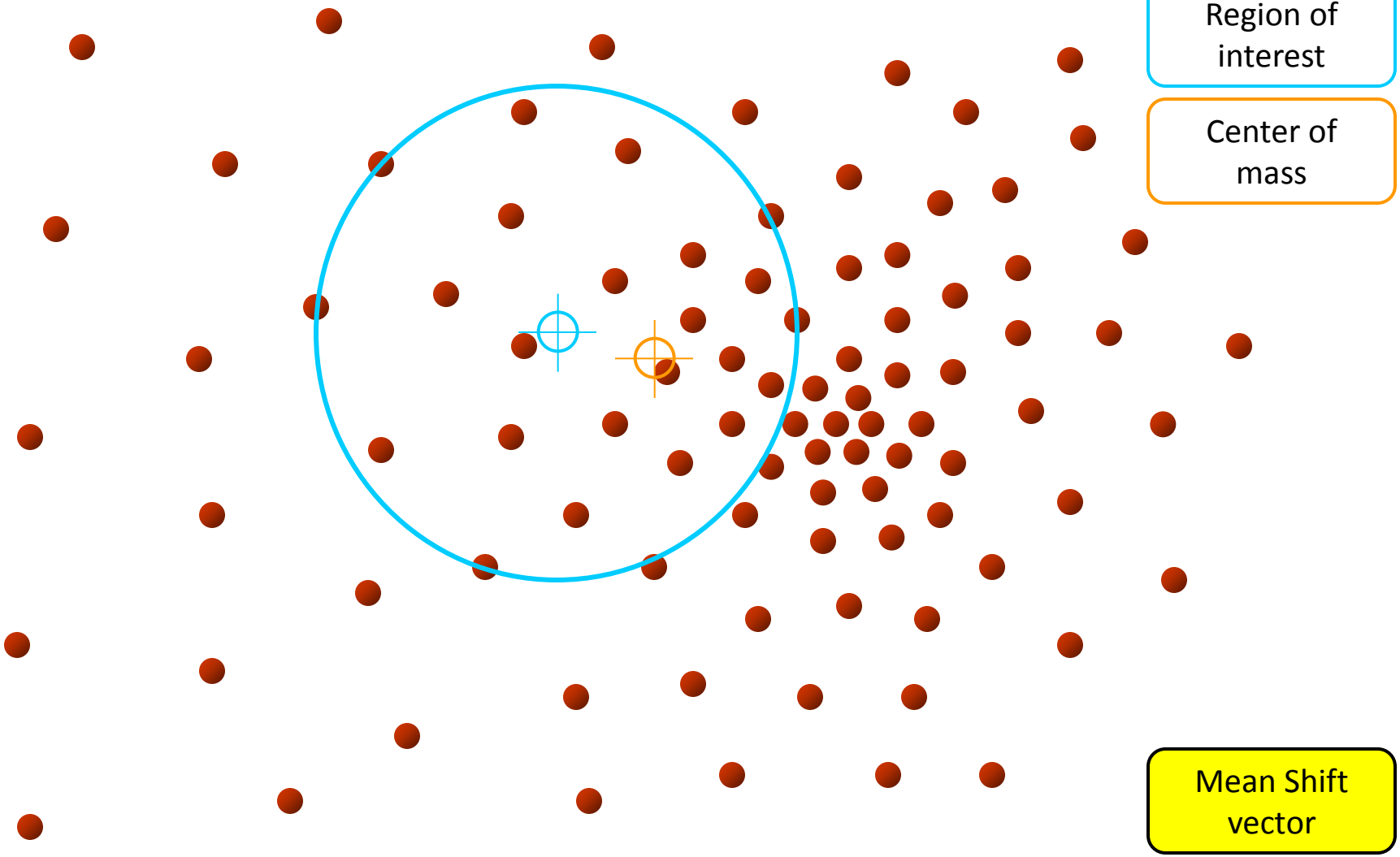


Objective : Find the densest region

Distribution of identical billiard balls

Slide taken from Ukrainitz & Sarel

Intuitive Description

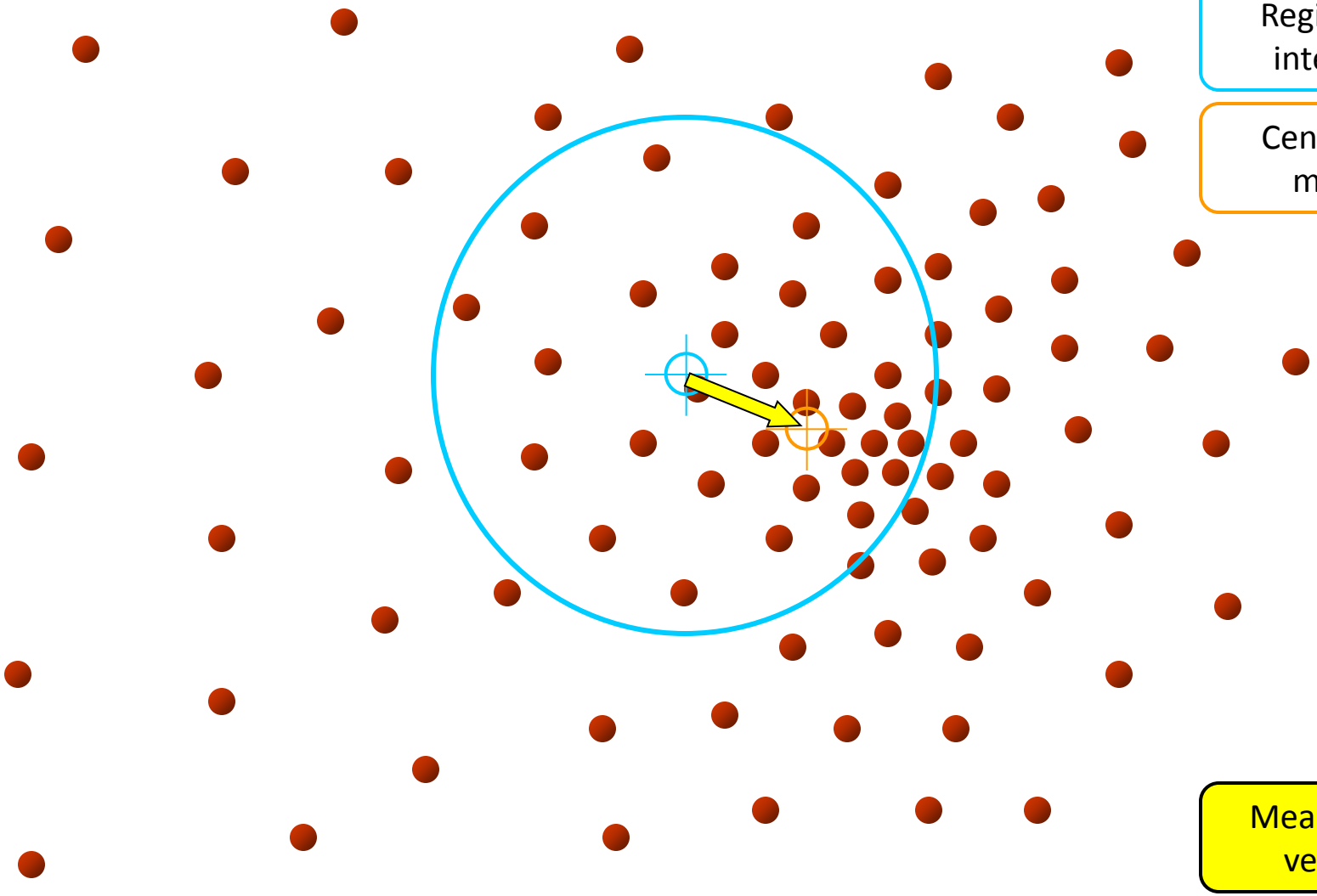


Objective : Find the densest region

Distribution of identical billiard balls

Slide taken from Ukrainitz & Sarel

Intuitive Description

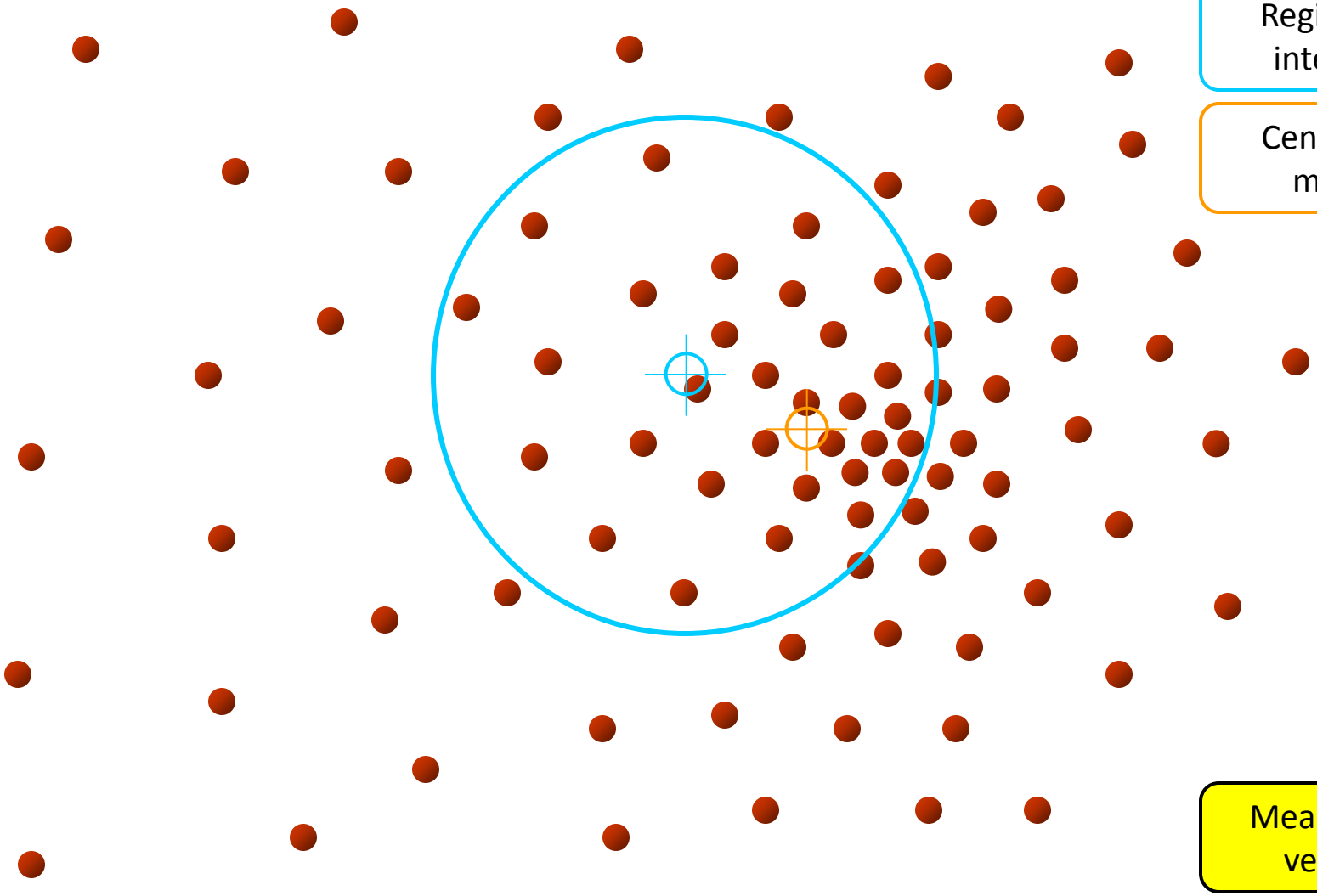


Objective : Find the densest region

Distribution of identical billiard balls

Slide taken from Ukrainitz & Sarel

Intuitive Description

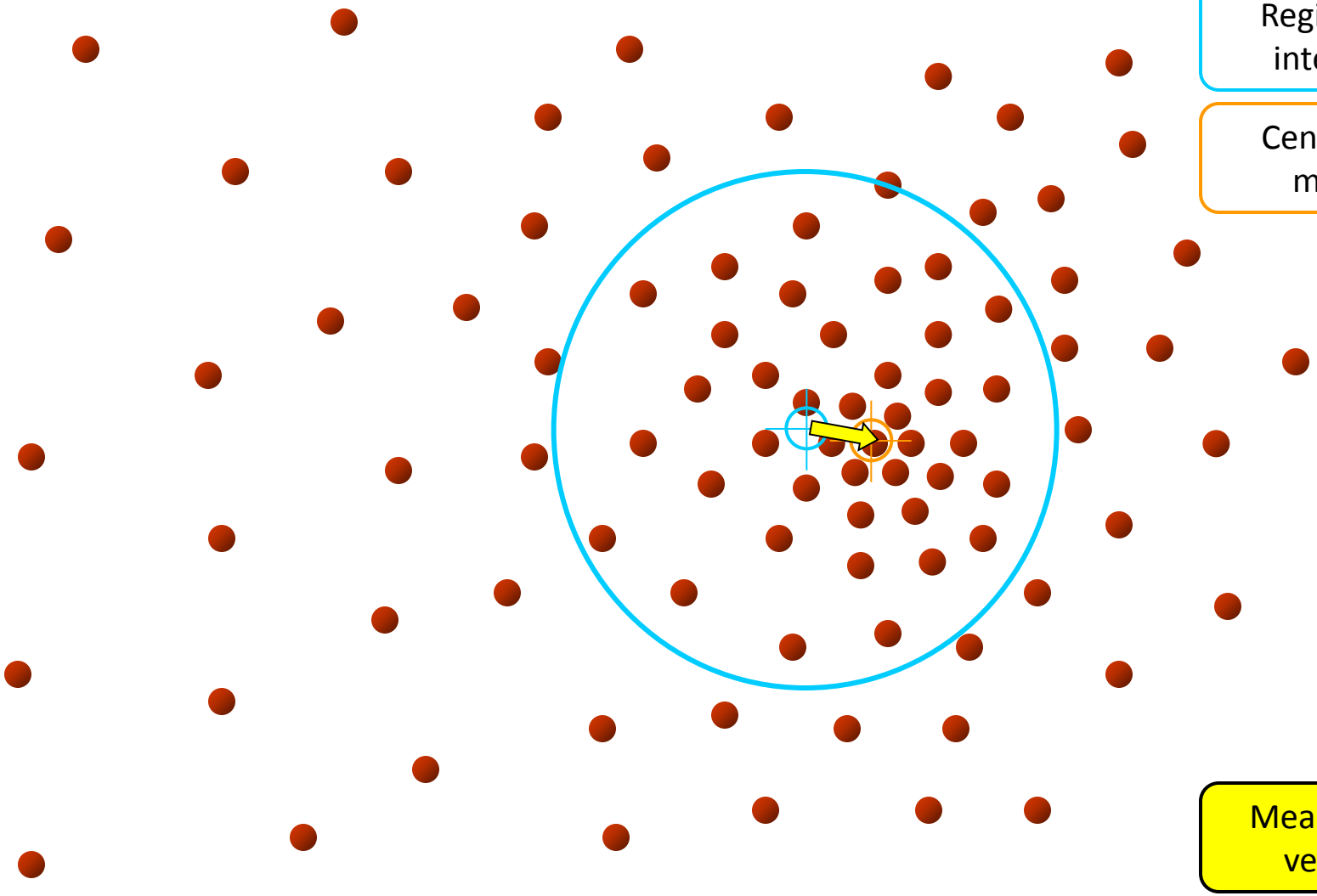


Objective : Find the densest region

Distribution of identical billiard balls

Slide taken from Ukrainitz & Sarel

Intuitive Description

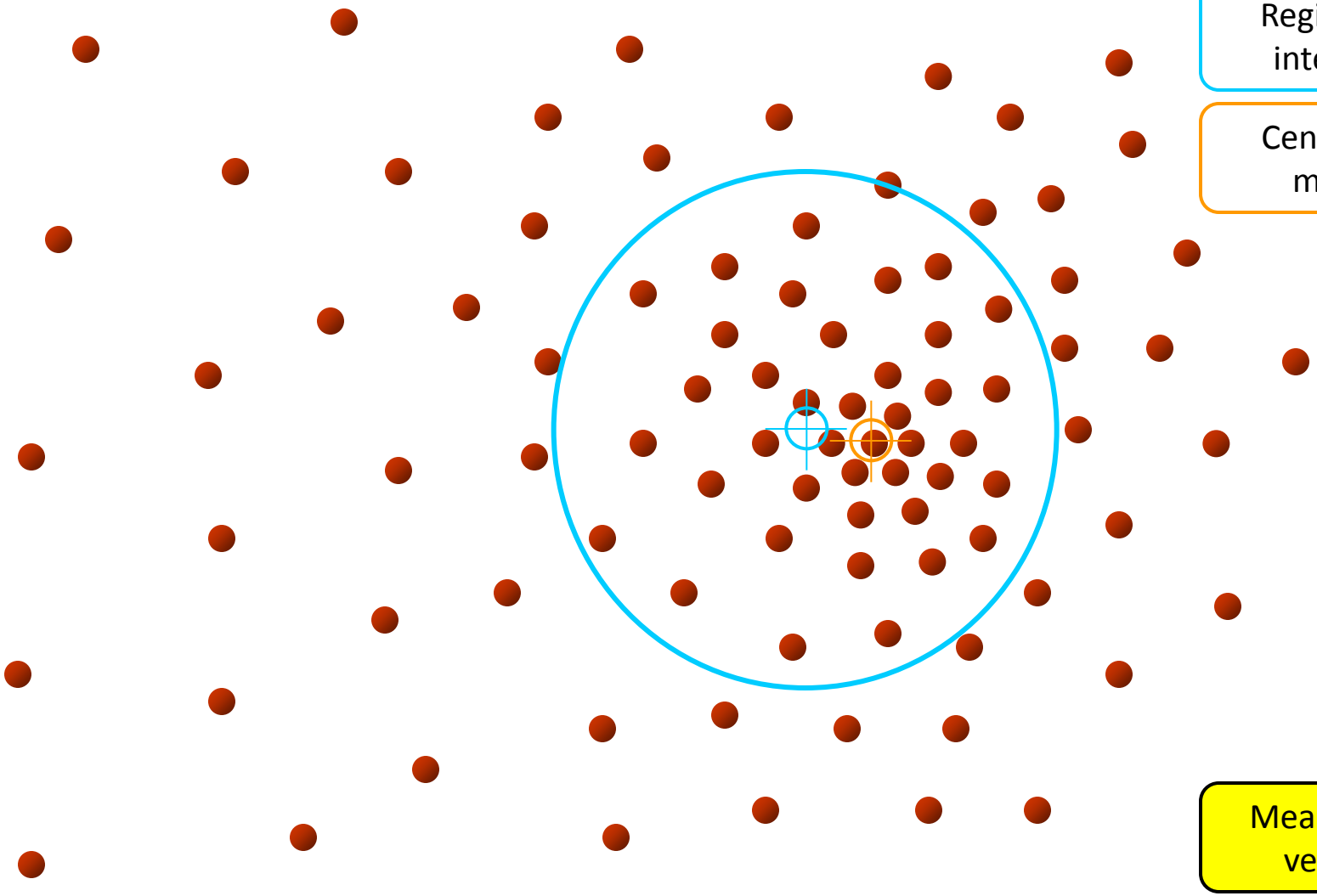


Objective : Find the densest region

Distribution of identical billiard balls

Slide taken from Ukrainitz & Sarel

Intuitive Description

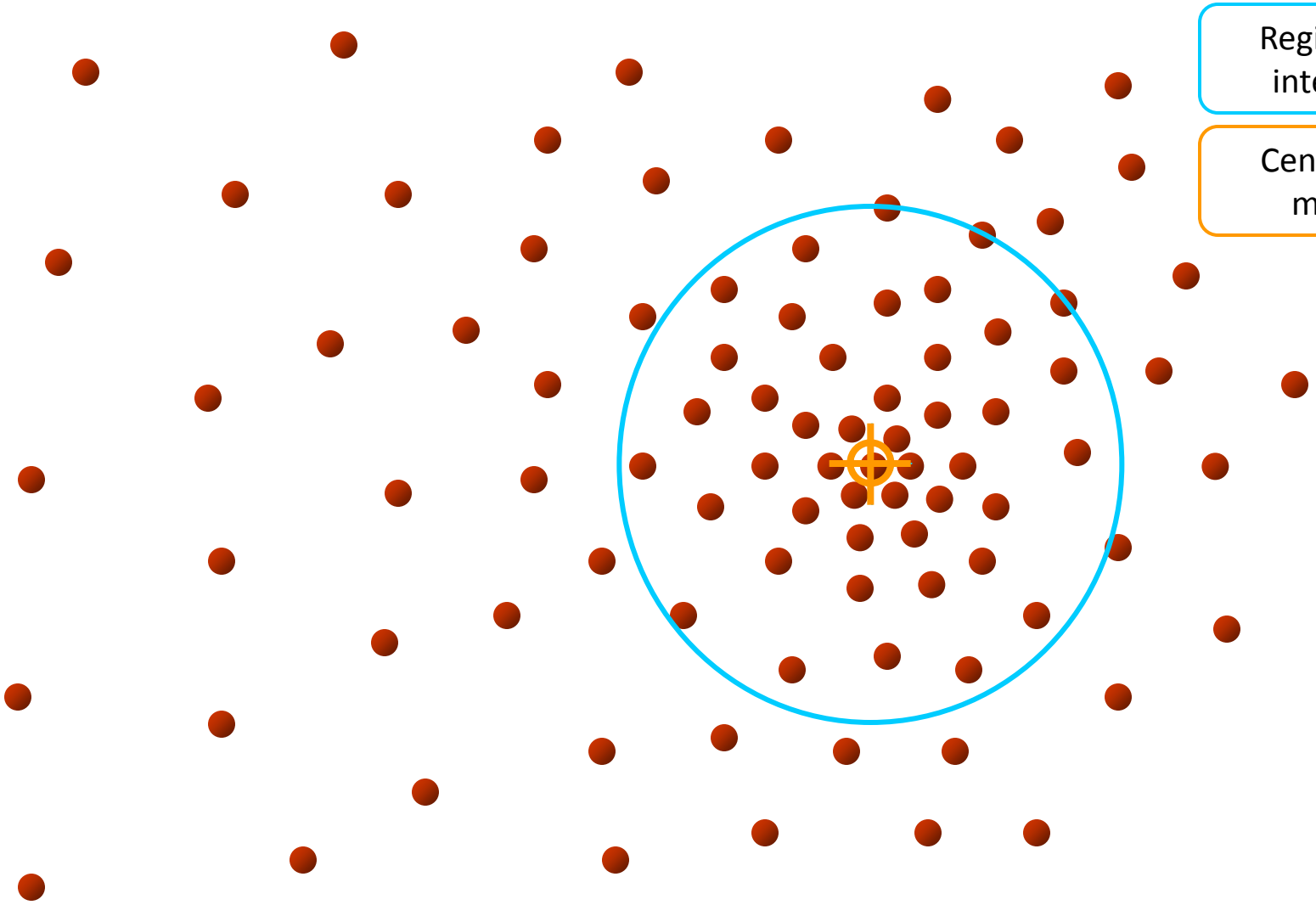


Objective : Find the densest region

Distribution of identical billiard balls

Slide taken from Ukrainitz & Sarel

Intuitive Description



Objective : Find the densest region

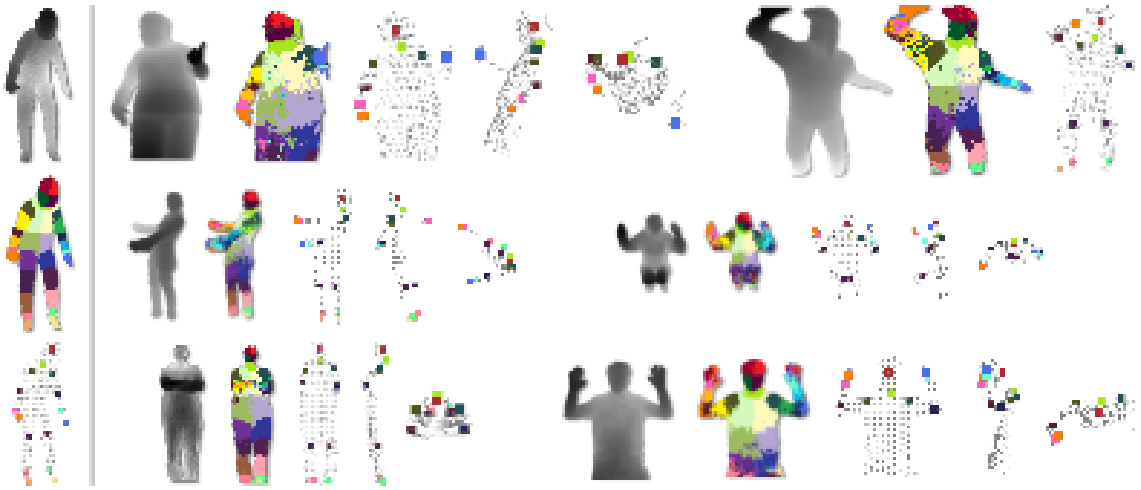
Distribution of identical billiard balls

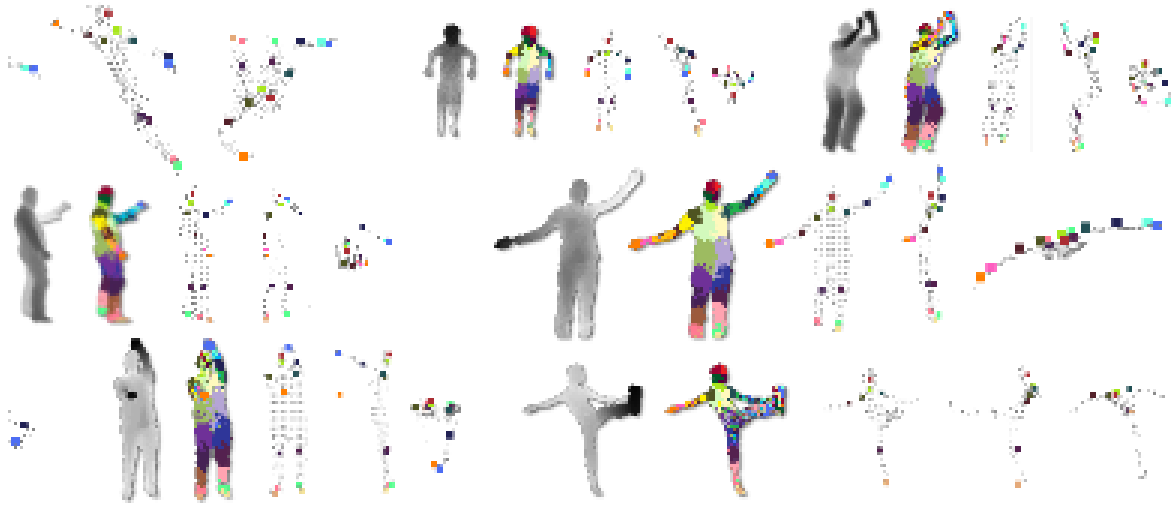
Slide taken from Ukrainitz & Sarel

The Kinect uses *structured light* and *machine learning*

- Inferring body position is a two-stage process: first compute a depth map (using structured light), then infer body position (using machine learning)
- The results are great!
- The system uses many college-level math concepts, and demonstrates the remarkable advances in computer vision in the last 20 years

The results are great!





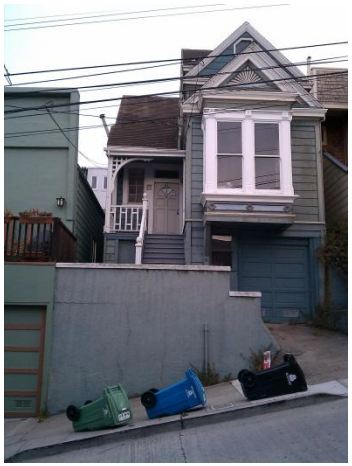
The results are great!

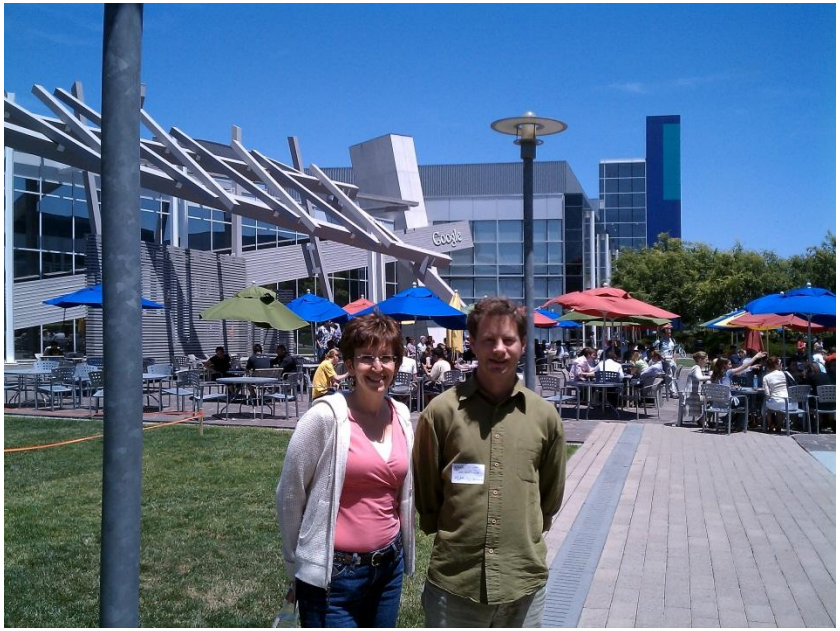
- But how can we make the results even better?

How can we make the results even better?

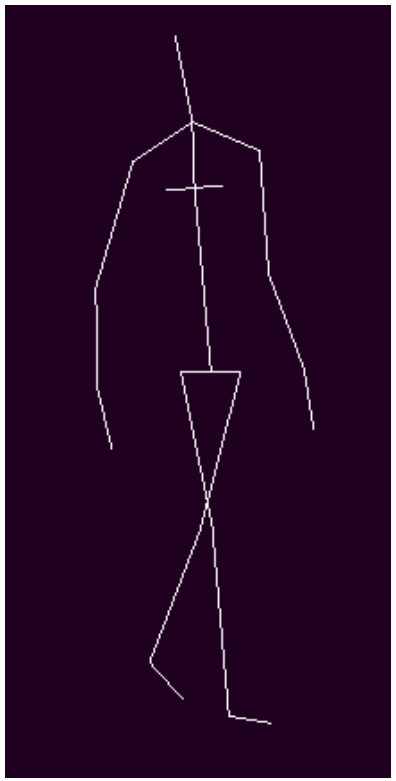
- Use temporal information
  - The Xbox team wrote a tracking algorithm that does this
- Reject “bad” skeletons and accept “good” ones
  - This is what I worked on last summer
- Of course there are plenty of other ways (next summer???)

Last summer (1)

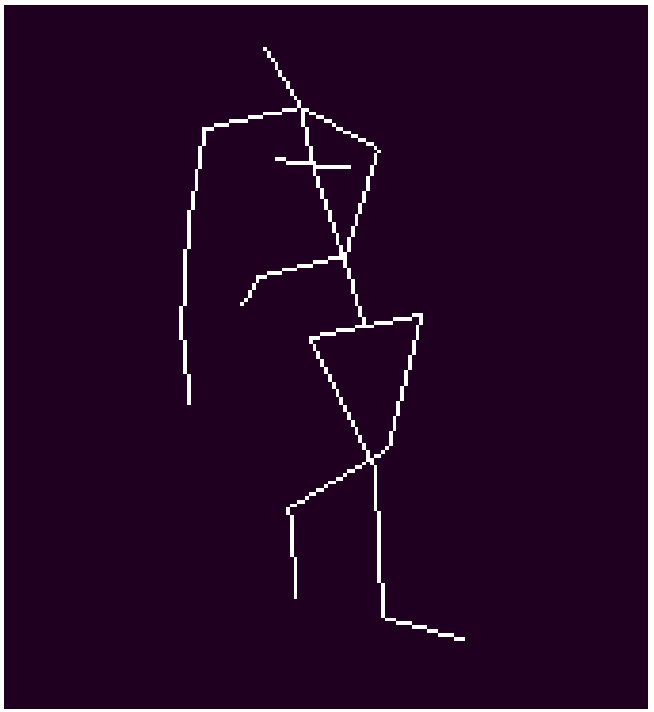




Last summer (2): Use the same ground truth dataset to infer skeleton likelihoods



likely



unlikely

Next summer: ????

- (working with Andrew Fitzgibbon at MSR Cambridge)

The Kinect uses *structured light* and *machine learning*

- Inferring body position is a two-stage process: first compute a depth map (using structured light), then infer body position (using machine learning)
- The results are great!
- The system uses many college-level math concepts, and demonstrates the remarkable advances in computer vision in the last 20 years

What kind of math is used in this stuff?

- PROBABILITY and STATISTICS
- MULTIVARIABLE CALCULUS
- LINEAR ALGEBRA
- Complex analysis, combinatorics, graph algorithms, geometry, differential equations, topology

Personal reflection: computer vision has come a long way since I started working on it in 1996.

- Key enablers of this progress:
  - Use of probabilistic and statistical models
  - Use of machine learning
  - Vastly faster hardware
  - Parallel and distributed implementations (clusters, multicore, GPU)
  - Cascades of highly flexible, simple features (not detailed models)

The Kinect uses *structured light* and *machine learning*