RGB-D Images and Applications

Yao Lu

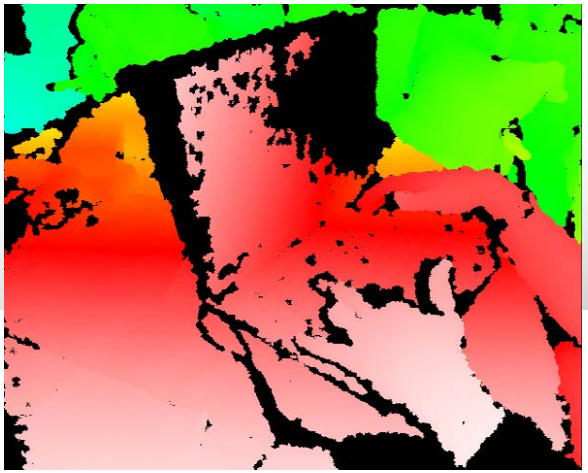
Outline

- Overview of RGB-D images and sensors
- Recognition: human pose, hand gesture
- Reconstruction: Kinect fusion

Outline

- **Overview of RGB-D images and sensors**
- Recognition: human pose, hand gesture
- Reconstruction: Kinect fusion

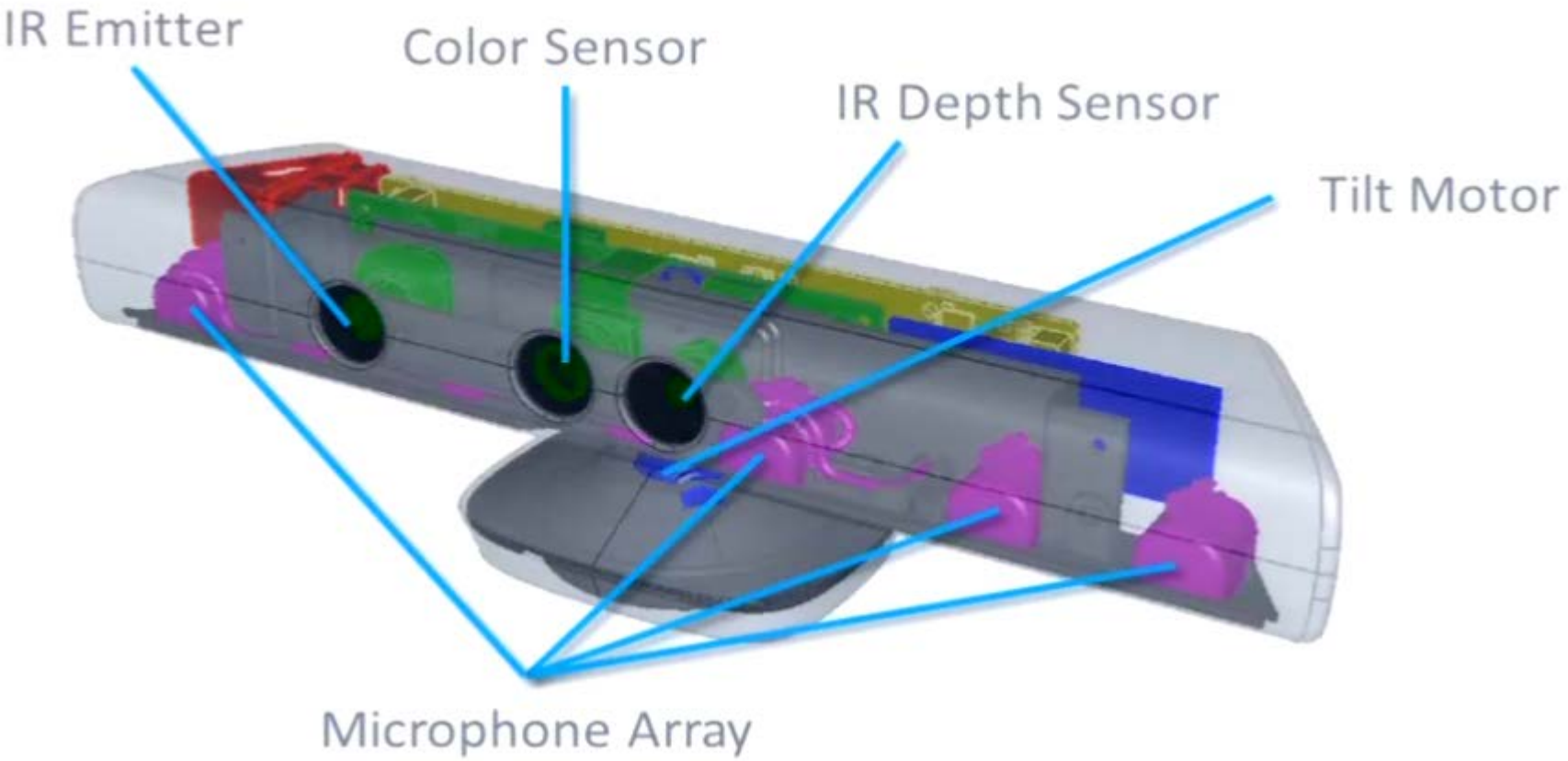

XBOX 360
KINECT



How does Kinect work?

Kinect has 3 components :-

- color camera (takes RGB values)
- IR camera (takes depth data)
- Microphone array (for speech recognition)



Depth Image

How Does the Kinect Compare?

- Distance Sensing
  - Alternatives Cheaper than Kinect
  - ~$2 Single-Point Close-Range Proximity Sensor

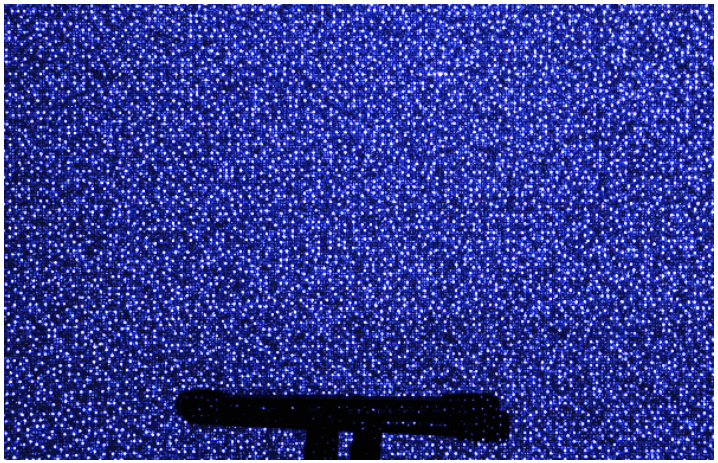
- Motion Sensing and 3D Mapping
  - High Performing Devices with Higher Cost

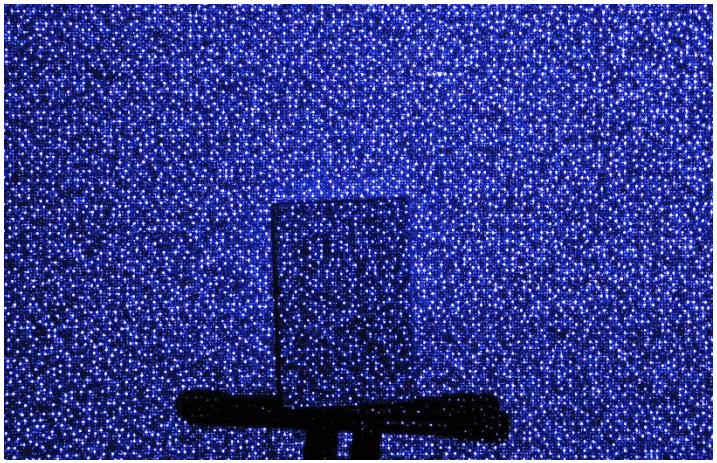
- Good Performance for Distance and Motion Sensing

- Provides a bridge between low cost and high performance sensors

Depth Sensor

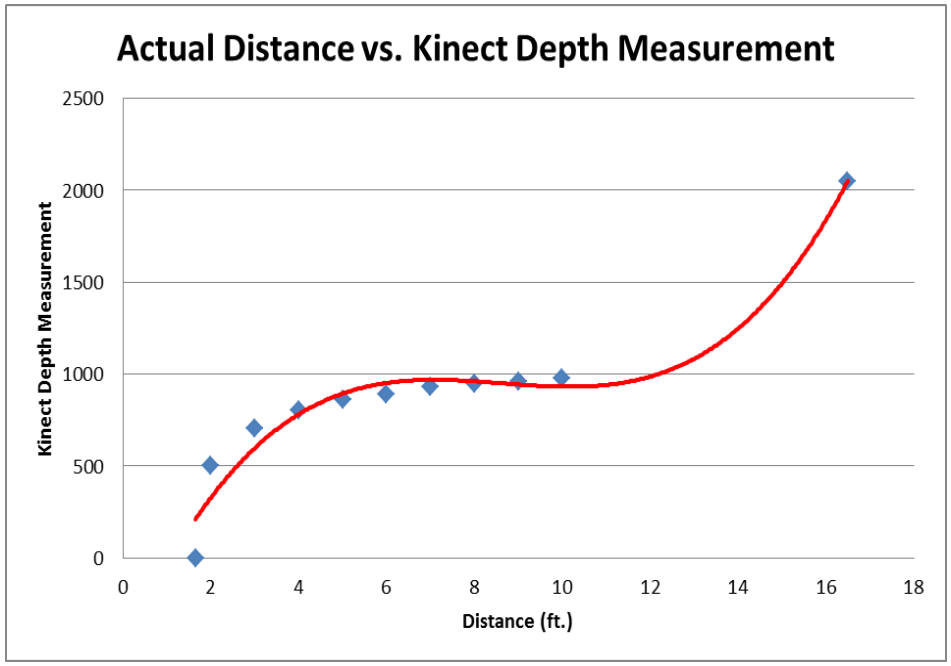
- IR projector emits predefined Dotted Pattern
- Lateral shift between projector and sensor
  - Shift in pattern dots
- Shift in dots determines Depth of Region





Kinect Accuracy

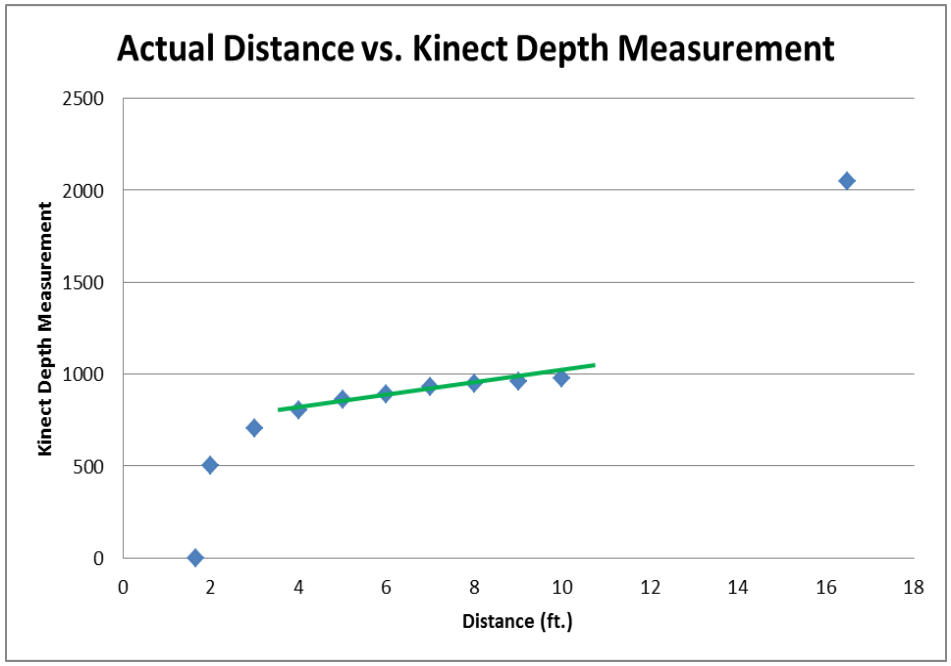
- OpenKinect SDK
- 11 Bit Accuracy
  - 2^{11} = 2048 possible values
- Measured Depth
  - Calculated 11 bit value
  - 2047 = maximum distance
    - Approx. 16.5 ft.
  - 0 = minimum distance
    - Approx. 1.65 ft.
- Reasonable Range
  - 4 – 10 feet
  - Provides Moderate Slope



Values from: http://mathnathan.com/2011/02/depthvsdistance/

Kinect Accuracy

- OpenKinect SDK
- 11 Bit Accuracy
  - 2^{11} = 2048 possible values
- Measured Depth
  - Calculated 11 bit value
  - 2047 = maximum distance
    - Approx. 16.5 ft.
  - 0 = minimum distance
    - Approx. 1.65 ft.
- Reasonable Range
  - 4 – 10 feet
  - Provides Moderate Slope



Values from: http://mathnathan.com/2011/02/depthvsdistance/

Other RGB-D sensors

- Intel RealSense Series

- Asus Xtion Pro

- Microsoft Kinect V2

- Structure Sensor

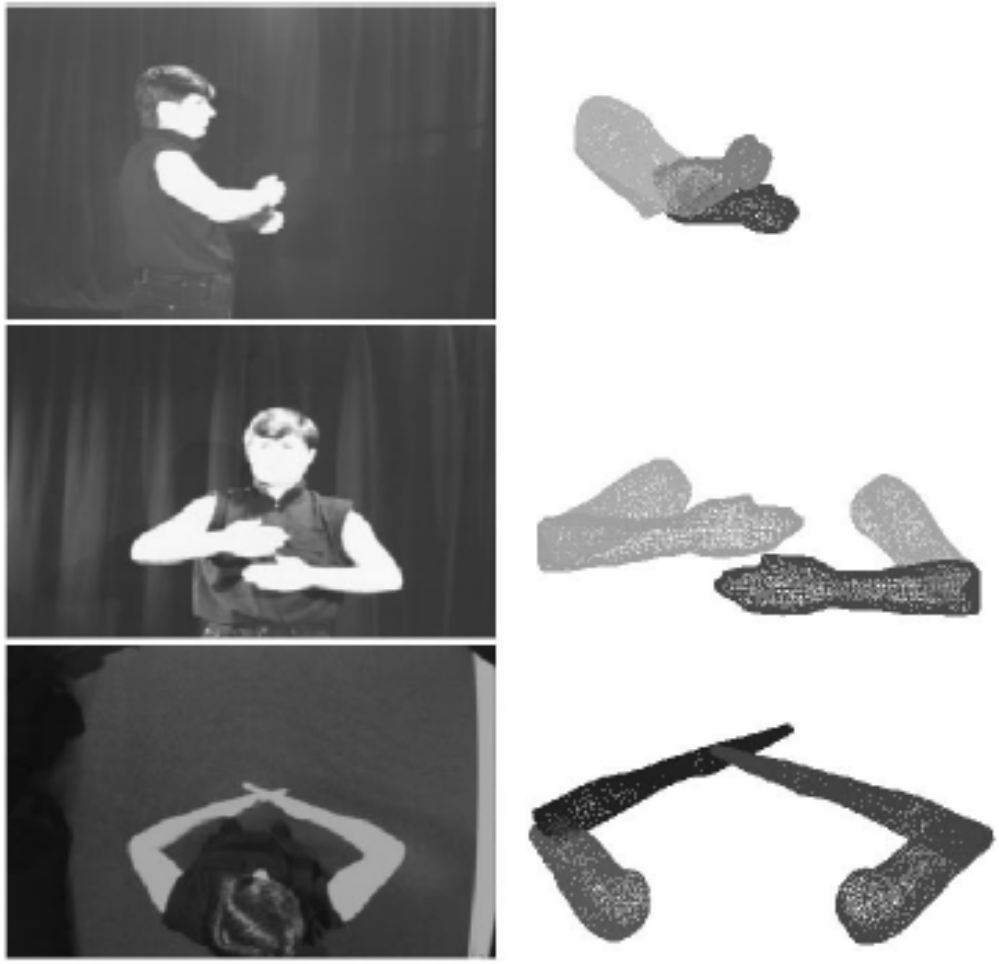
Outline

- Overview of RGB-D images and sensors
- Recognition: **human pose**, hand gesture
- Reconstruction: Kinect fusion

Recognition:
Human Pose recognition

- Research in pose recognition has been on going for 20+ years.

- Many assumptions: multiple cameras, manual initialization, controlled/simple backgrounds



Model-Based Estimation of 3D Human Motion, Ioannis Kakadiaris and Dimitris Metaxas, PAMI 2000

Fig. 22. Our automatic tracker on commercial sports footage with fast and extreme motions. On the **top**, we show results from a 300 frame sequence of a baseball pitch from the 2002 World Series. On the **bottom**, we show results from the *complete* medal-winning performance of Michelle Kwan from the 1998 Winter Olympics. We label frame numbers from the 7,600-frame sequence. For each sequence, our system first runs a walking pose finder on each frame; and then uses the single frame with the best score (shown in the **left insets**) to train the discriminative appearance models. In the baseball sequence, our system is able to track through frames with excessive motion blur and interlacing effects (the **center inset**). In the skating sequence, our system is able to track through extreme poses for thousands of frames. This means that our system can track long sequences with extreme poses, complex backgrounds, and fast movement *without manual intervention*.

Tracking People by Learning Their Appearance, Deva Ramanan, David A. Forsyth, and Andrew Zisserman, PAMI 2007

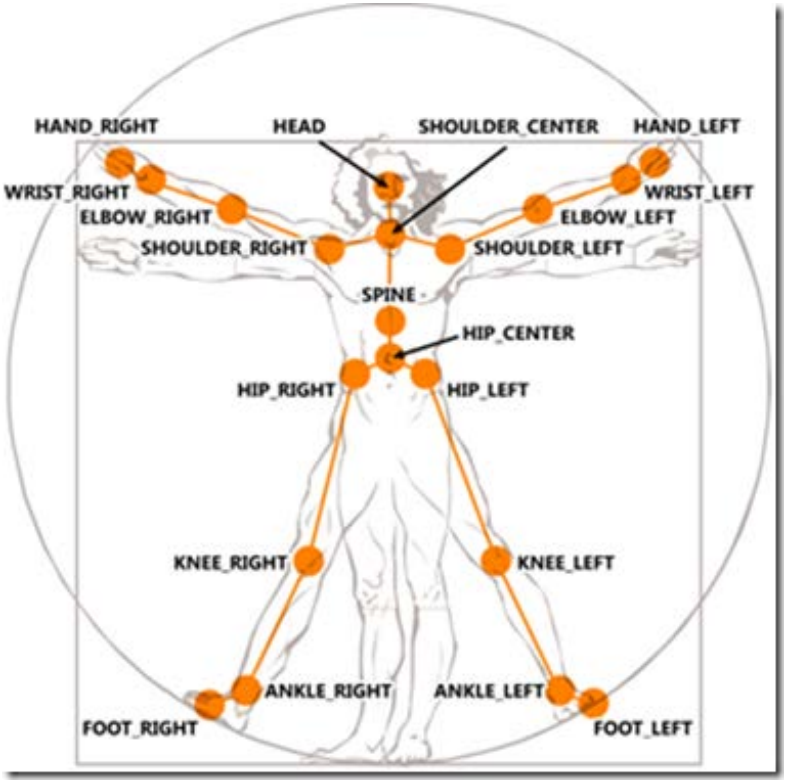
Kinect

- Why does depth help?

Algorithm design

Shotton et al. proposed two main steps:

1. Find body parts
2. Compute joint positions.



Real-Time Human Pose Recognition in Parts from Single Depth Images
Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake, CVPR 2011

Finding body parts

- What should we use for a feature?

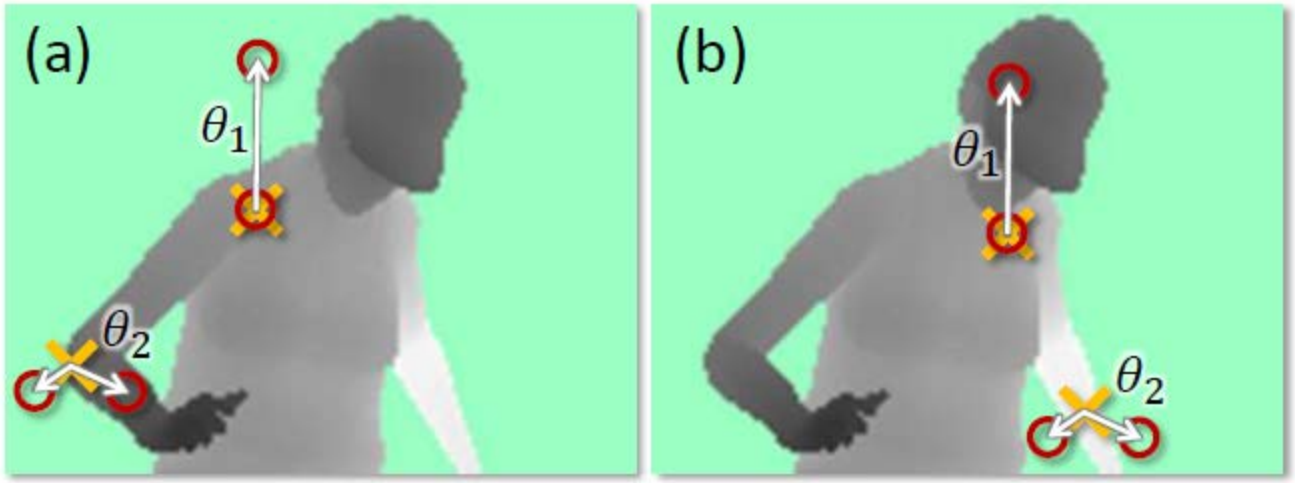
- What should we use for a classifier?

Finding body parts

- What should we use for a feature?
  - Difference in depth

- What should we use for a classifier?
  - Random Decision Forests
    - A set of decision trees

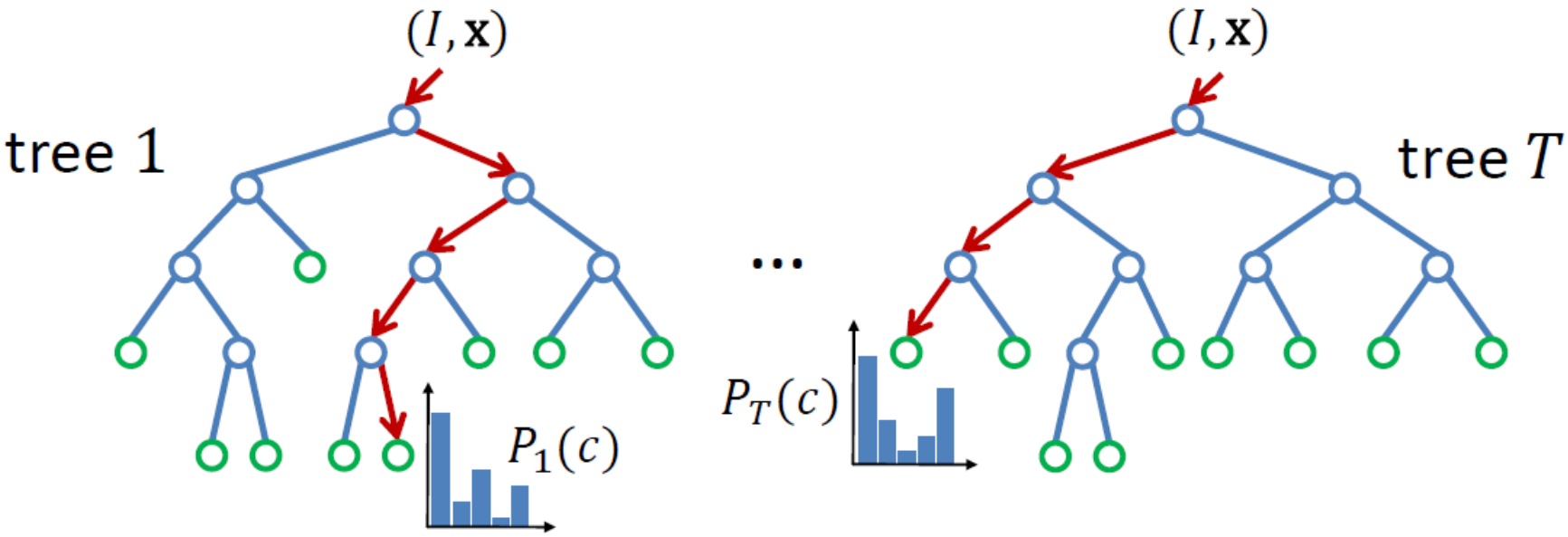
Features



$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

$d_I(x)$: depth at pixel x in image *I*
u, v: parameters describing offsets

Classification



$$P(c|I,\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,\mathbf{x})$$

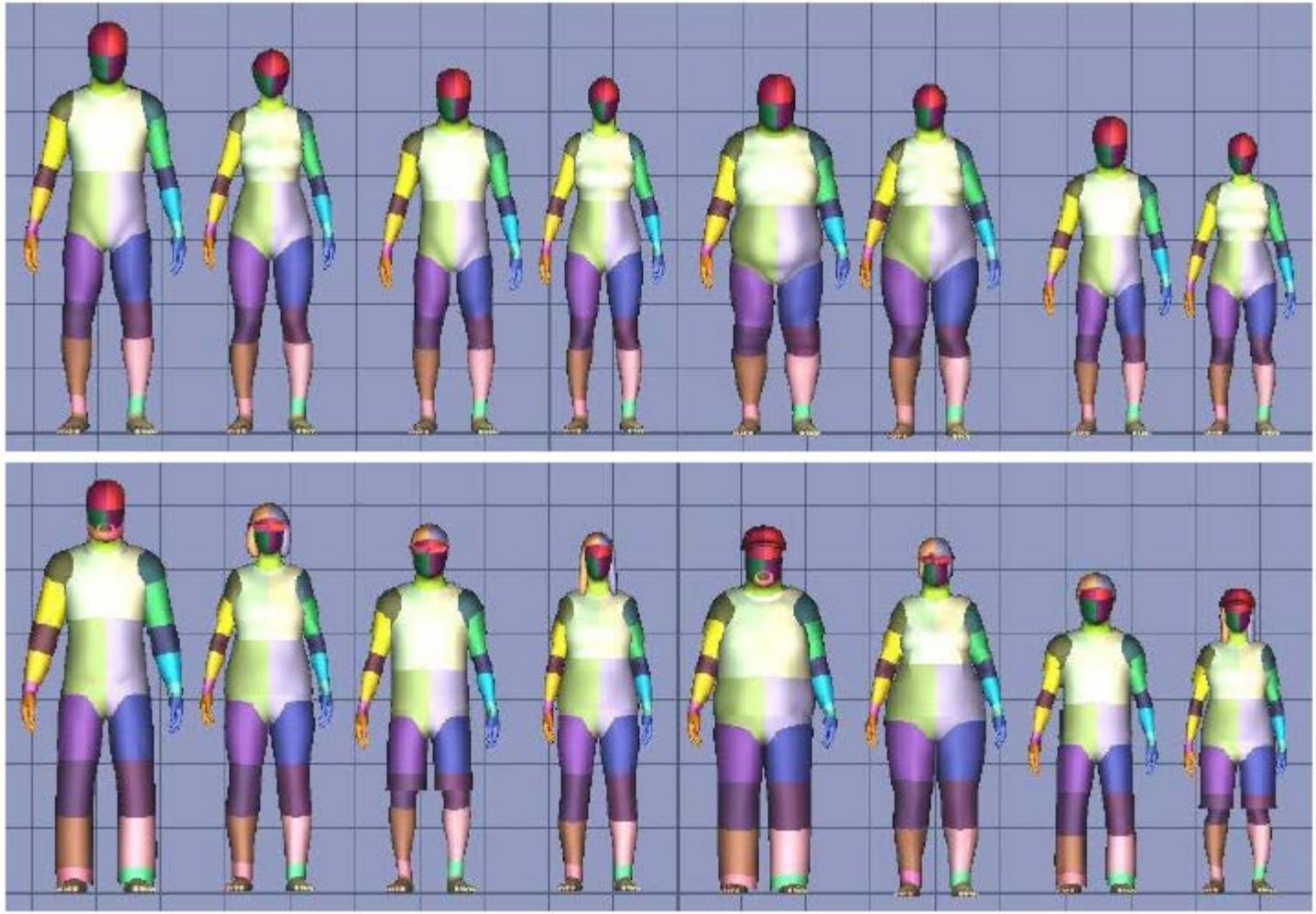
Learning:

1. Randomly choose a set of thresholds and features for splits.
2. Pick the threshold and feature that provide the largest information gain.
3. Recurse until a certain accuracy is reached or depth is obtained.

Implementation details

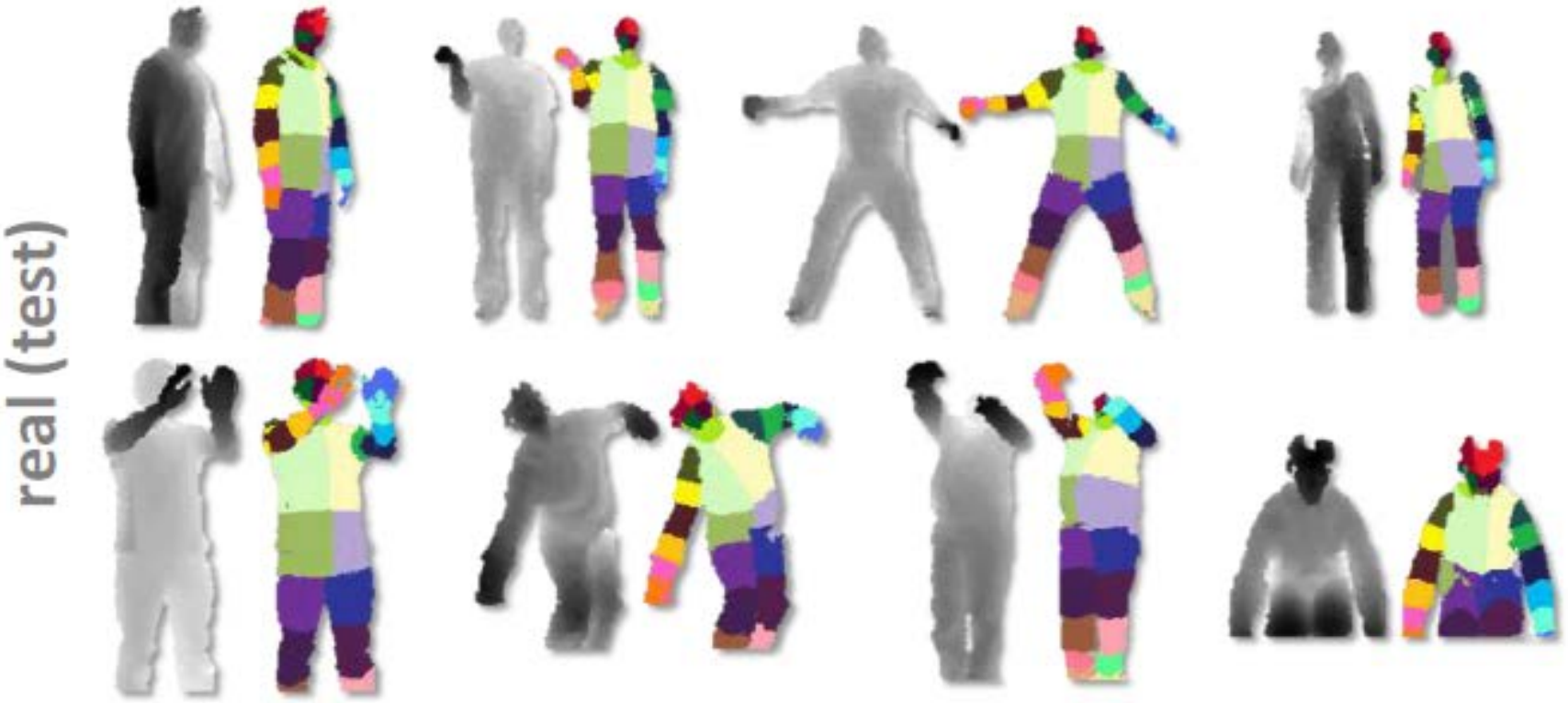
- 3 trees (depth 20)
- 300k unique training images per tree.
- 2000 candidate features, and 50 thresholds
- One day on 1000 core cluster.

Synthetic data



Synthetic training/testing

Real test



Results

Estimating joints

- Apply mean-shift clustering to the labeled pixels.
- “Push back” each mode to lie at the center of the part.

Results

Outline

- Overview of RGB-D images and sensors
- Recognition: human pose, **hand gesture**
- Reconstruction: Kinect fusion

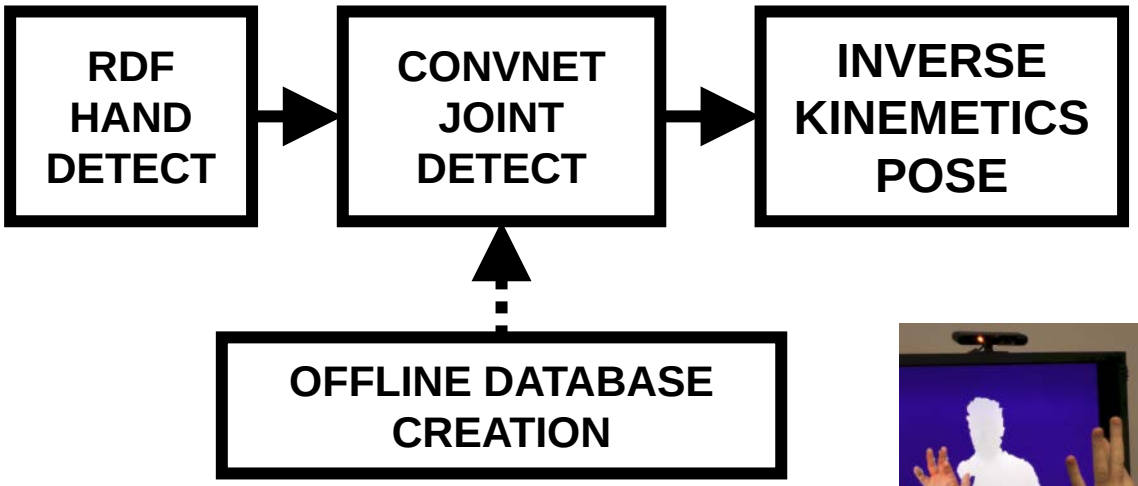
Hand gesture recognition

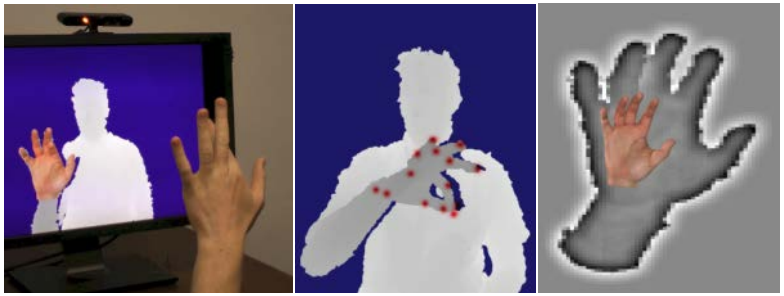
Hand Pose Inference

- Target: low-cost markerless mocap
  - Full articulated pose with high DoF
  - Real-time with low latency
- Challenges
  - Many DoF contribute to model deformation
  - Constrained unknown parameter space
  - Self-similar parts
  - Self occlusion
  - Device noise

Pipeline Overview

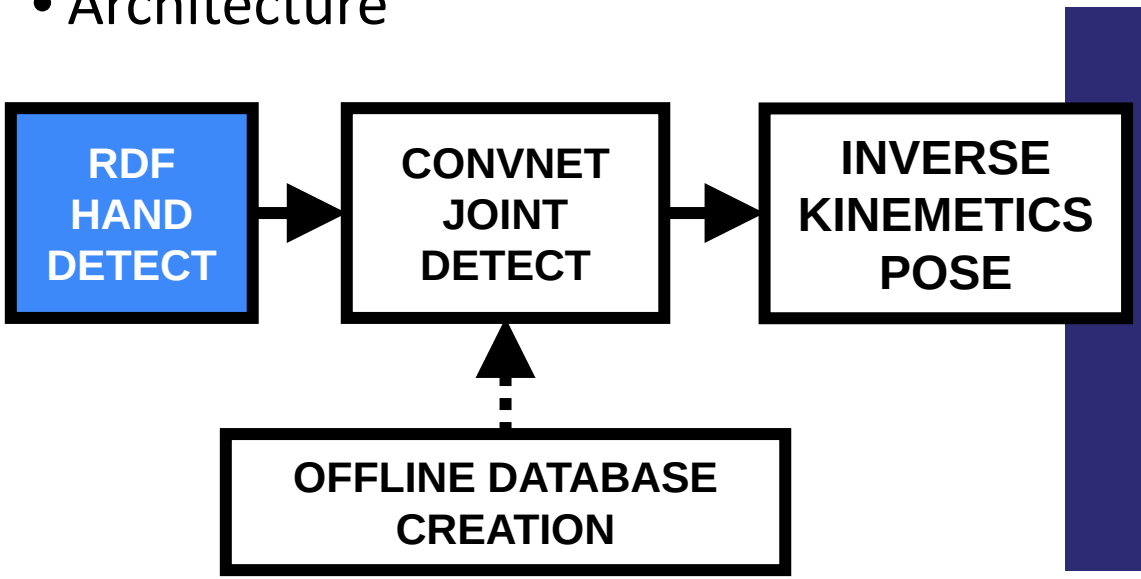
- Tompson et al. Real-time continuous pose recovery of human hands using convolutional networks. *ACM SIGGRAPH 2014.*
- Supervised learning based approach
  - Needs labeled dataset + machine learning
  - Existing datasets had limited pose information for hands
- Architecture





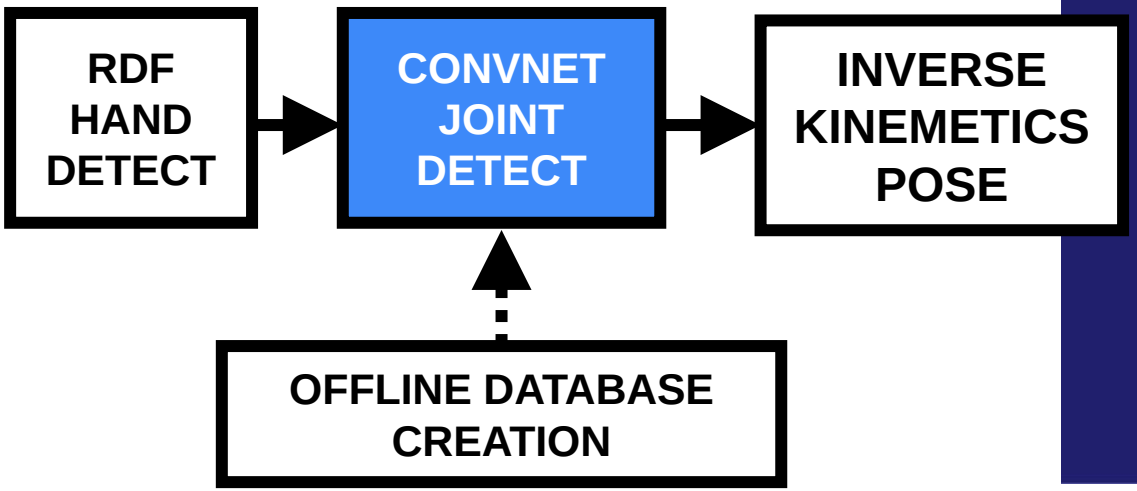
Pipeline Overview

- Supervised learning based approach
  - Needs labeled dataset + machine learning
  - Existing datasets had limited pose information for hands
- Architecture



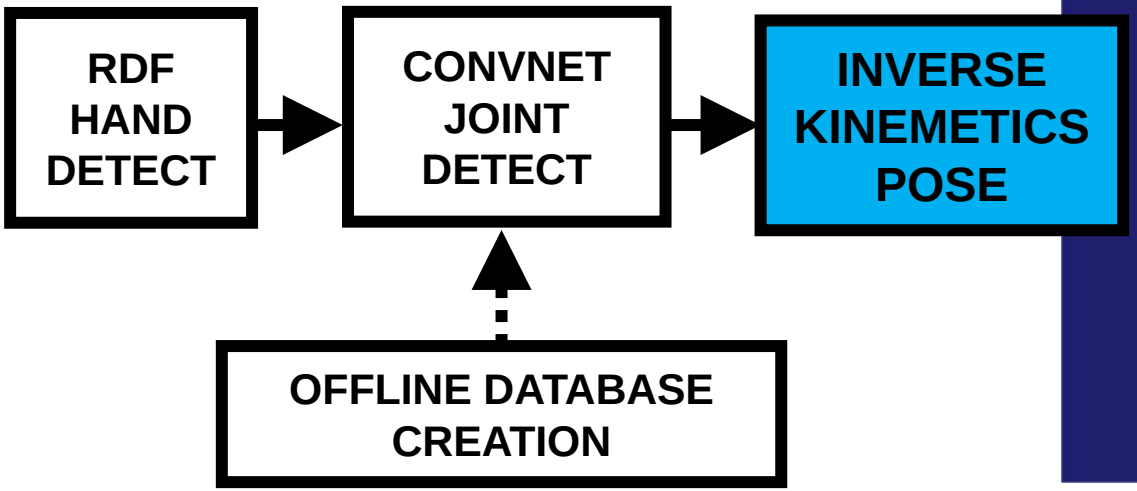
Pipeline Overview

- Supervised learning based approach
  - Needs labeled dataset + machine learning
  - Existing datasets had limited pose information for hands
- Architecture



Pipeline Overview

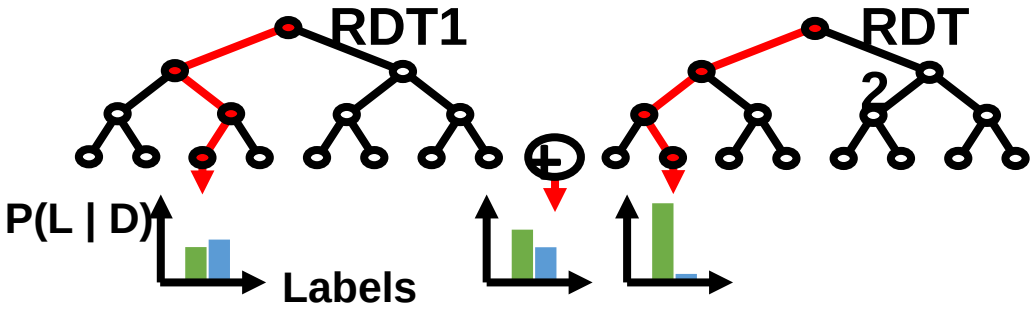
- Supervised learning based approach
  - Needs labeled dataset + machine learning
  - Existing datasets had limited pose information for hands
- Architecture



RDF Hand Detection

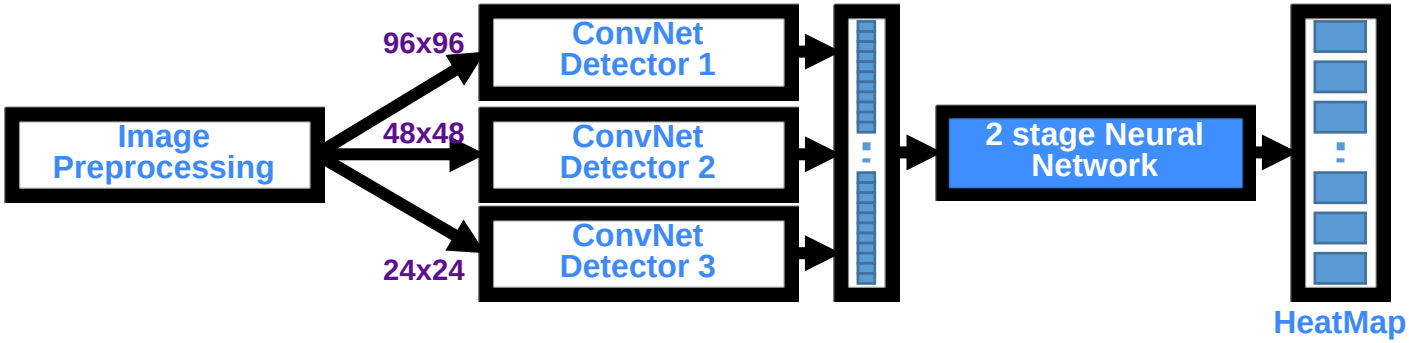
- Per-pixel binary classification → Hand centroid location

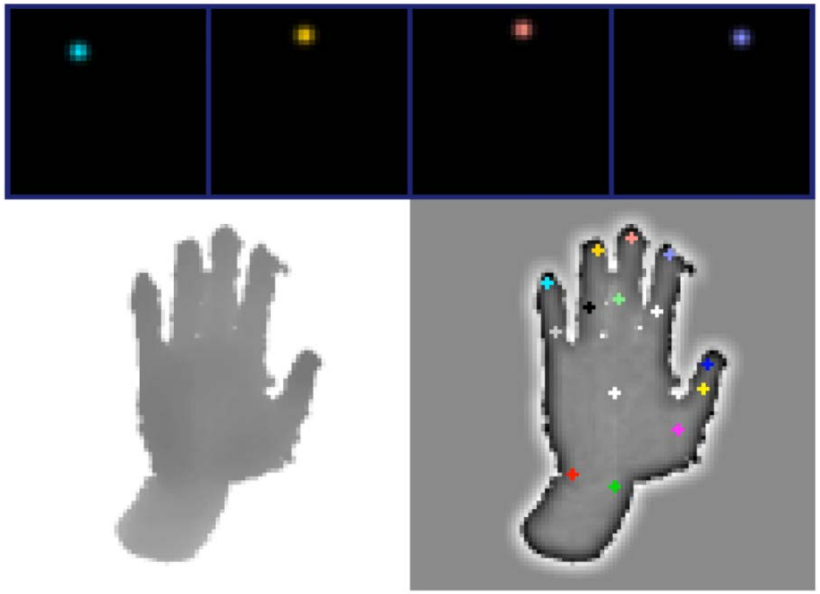
- Randomized decision forest (RDF)
  - Shotton et al.[1]
  - Fast (parallel)
  - Generalize



[1] J. Shotten et al., Real-time human pose recognition in parts from single depth images, CVPR 11

Inferring Joint Positions





PrimeSense Depth

ConvNet Depth

Hand Pose Inference

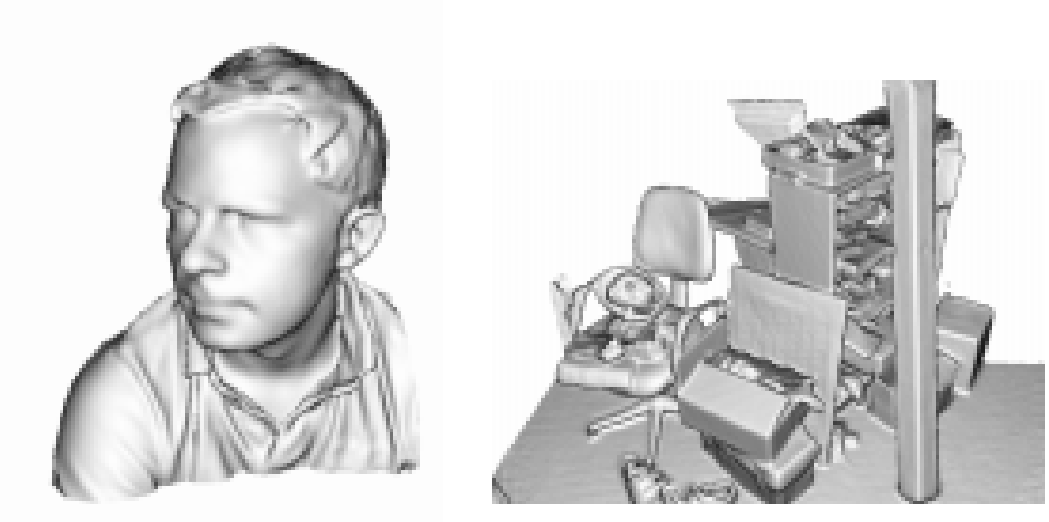
- Results

Outline

- Overview of RGB-D images and sensors
- Recognition: human pose, hand gesture
- **Reconstruction: Kinect fusion**

Reconstruction: Kinect Fusion

- Newcombe et al. KinectFusion: Real-time dense surface mapping and tracking. *2011 IEEE International Symposium on Mixed and Augmented Reality.*
- https://www.youtube.com/watch?v=quGhaggn3cQ



Motivation

Augmented Reality

Robot Navigation

3d model scanning

Etc..

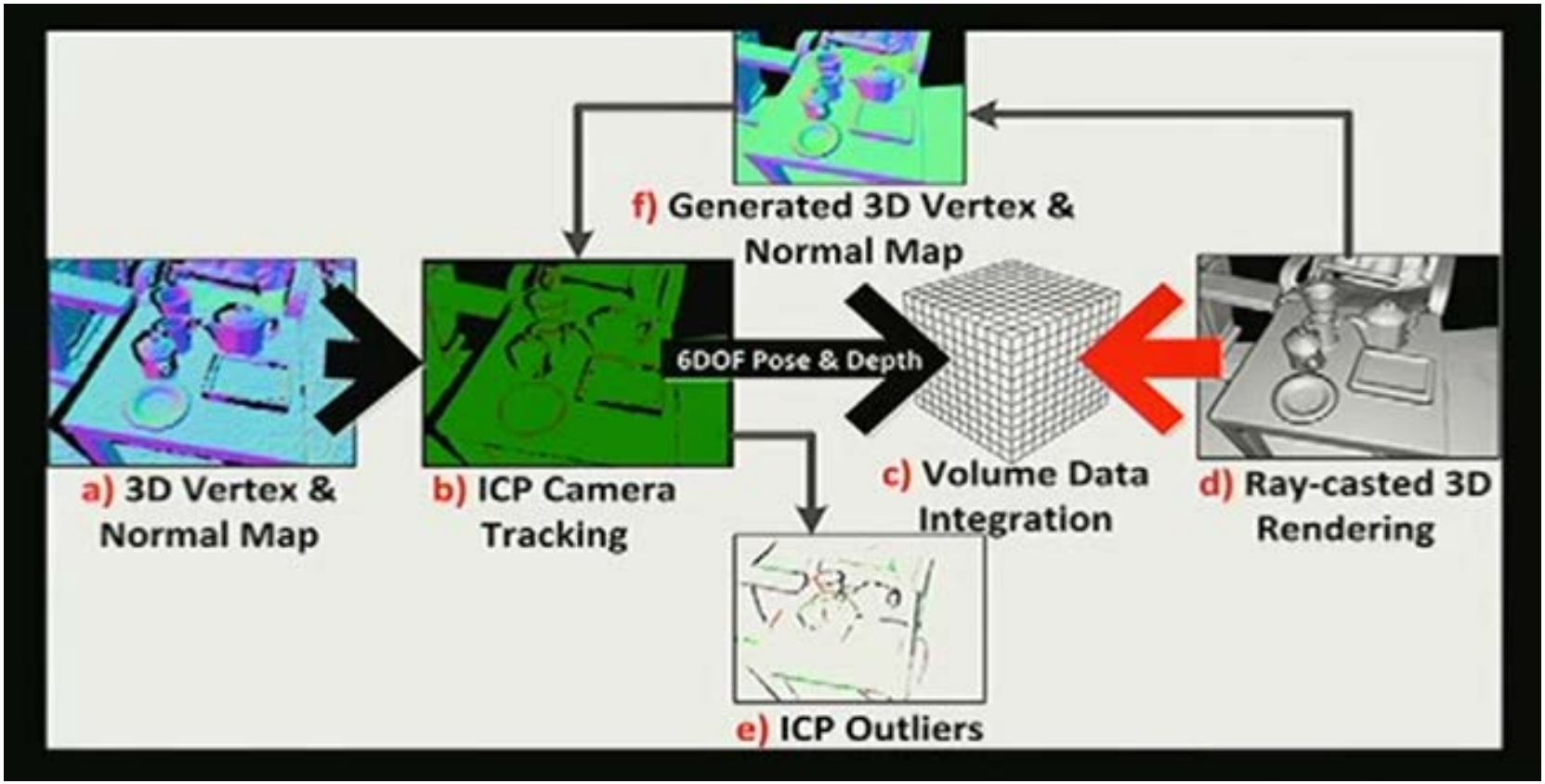
Challenges

- Tracking Camera Precisely
- Fusing and De-noising Measurements
- Avoiding Drift
- Real-Time
- Low-Cost Hardware

Proposed Solution

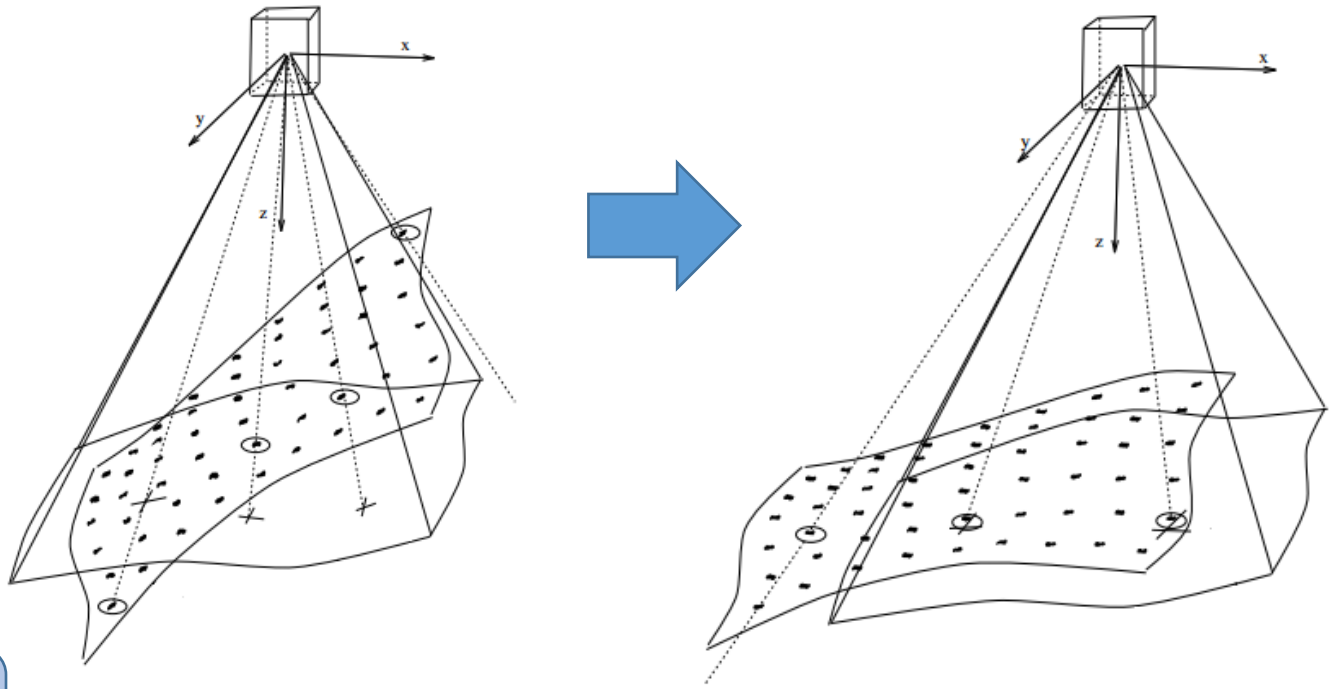
- Fast Optimization for Tracking, Due to High Frame Rate.
- Global Framework for fusing data
- Interleaving Tracking & Mapping
- Using Kinect to get Depth data (low cost)
- Using GPGPU to get Real-Time Performance (low cost)

Method


f) Generated 3D Vertex &
Normal Map
6DOF Pose & Depth
a) 3D Vertex &
Normal Map
b) ICP Camera
Tracking
c) Volume Data
Integration
d) Ray-casted 3D
Rendering
e) ICP Outliers

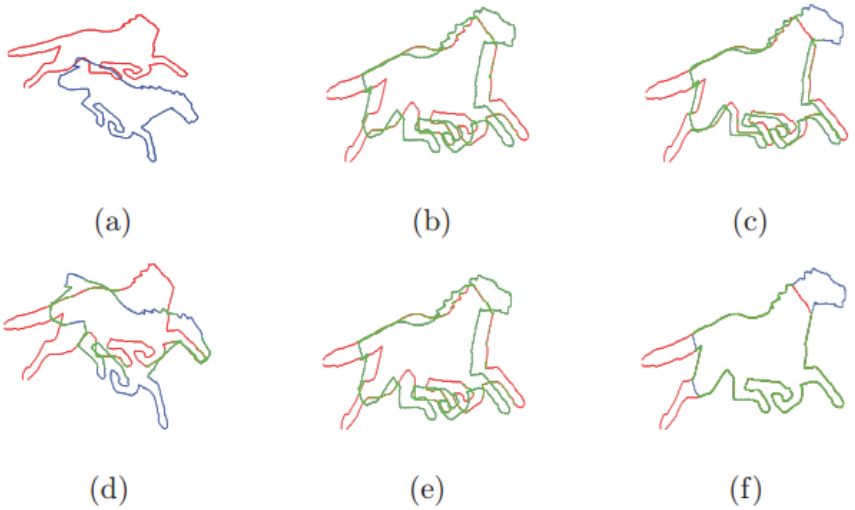
Tracking

- Finding Camera position is the same as fitting frame's Depth Map onto Model



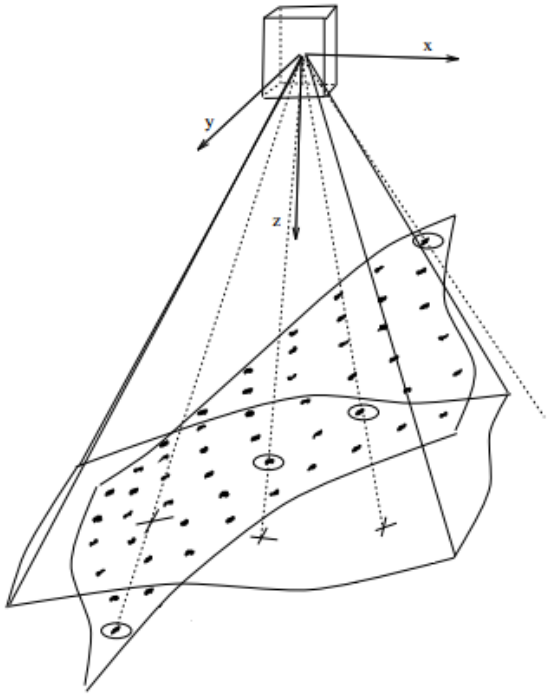
Tracking
Mapping

Tracking – ICP algorithm



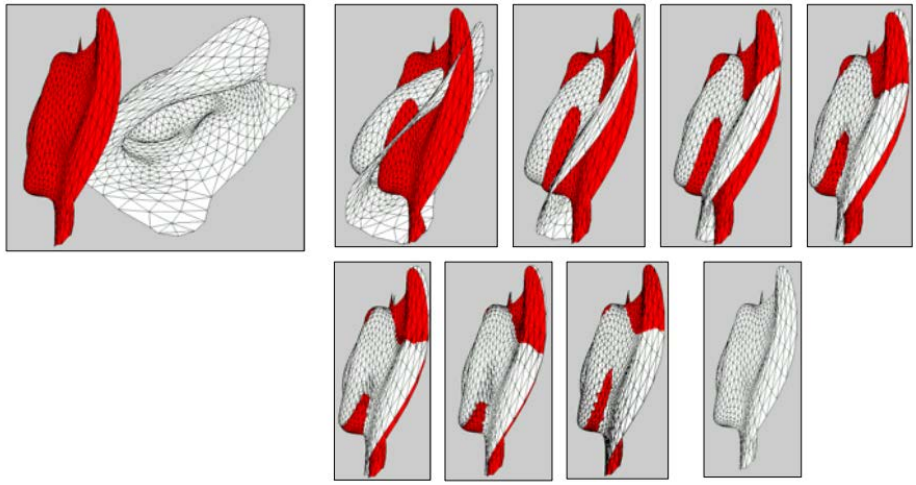
- icp = iterative closest point
- Goal: fit two 3d point sets
- Problem: What are the correspondences?

- Kinect fusion chosen solution:
  1) Start with T_0
  2) Project model onto camera
  3) Correspondences are points with same coordinates
  4) Find new T with Least - Squares
  5) Apply T, and repeat 2-5 until convergence



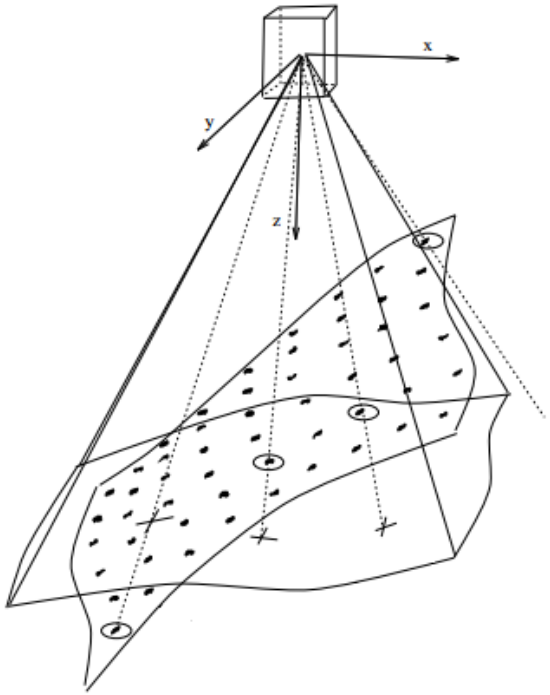
Tracking
Mapping

Tracking – ICP algorithm



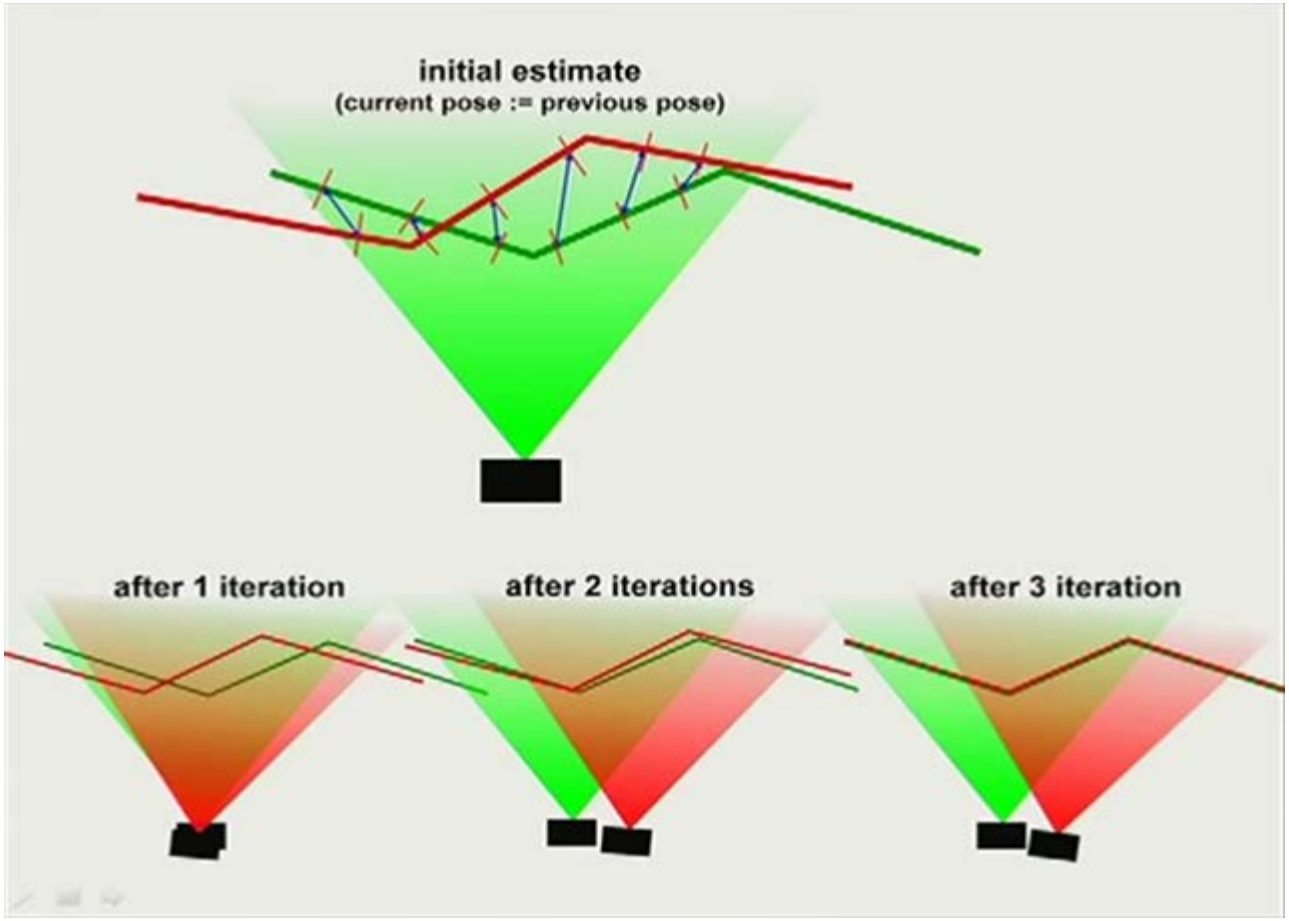
- icp = iterative closest point
- Goal: fit two 3d point sets
- Problem: What are the correspondences?

- Kinect fusion chosen solution:
  1) Start with T_0
  2) Project model onto camera
  3) Correspondences are points with same coordinates
  4) Find new T with Least - Squares
  5) Apply T, and repeat 2-5 until convergence



Tracking
Mapping

Tracking – ICP algorithm

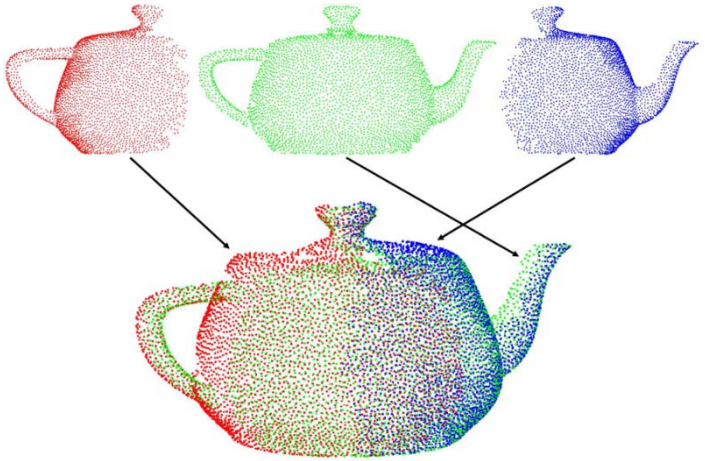


Tracking
Mapping

- Assumption: frame and model are roughly aligned.
- True because of high frame rate

Mapping

- Mapping is Fusing depth maps <u>when camera poses are known</u>
  - Model from existing frames
  - New frame

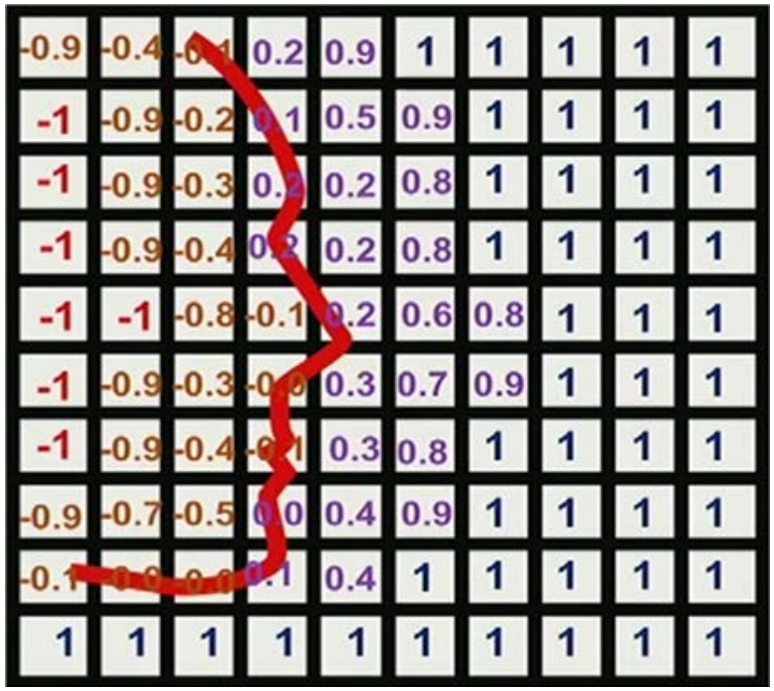


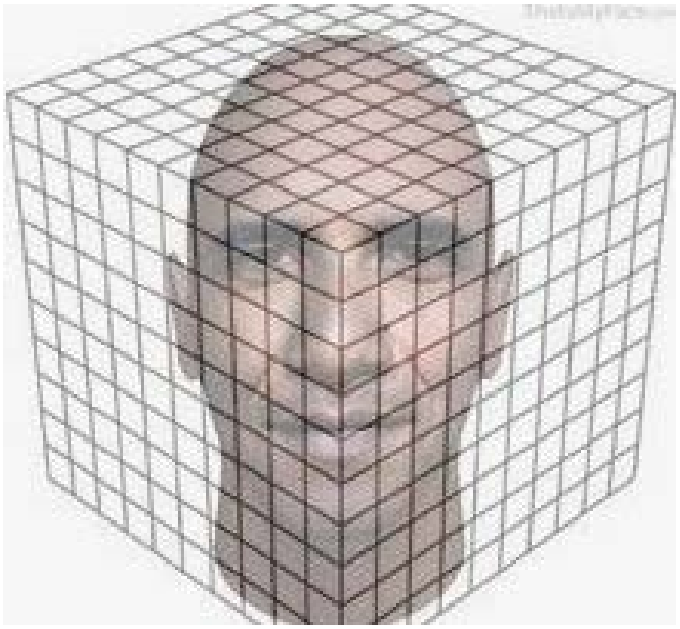
- Problems:
  - measurements are noisy
  - Depth maps have holes in them
- Solution:
  - using implicit surface representation
  - Fusing = estimating from all frames relevant

Tracking
➢ Mapping

Mapping – surface representation

- Surface is represented implicitly - using Truncated Signed Distance Function (TSDF)



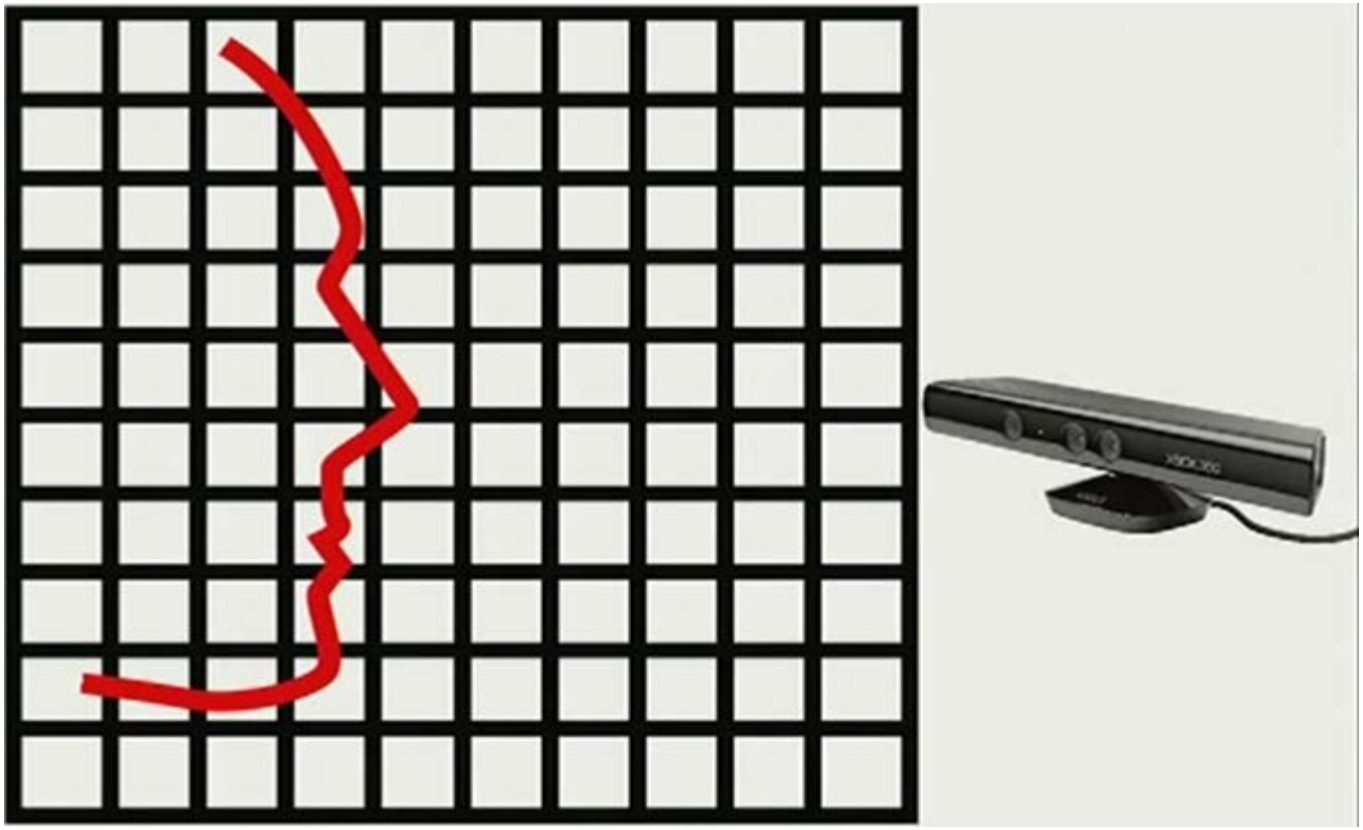


Voxel grid

Tracking
Mapping

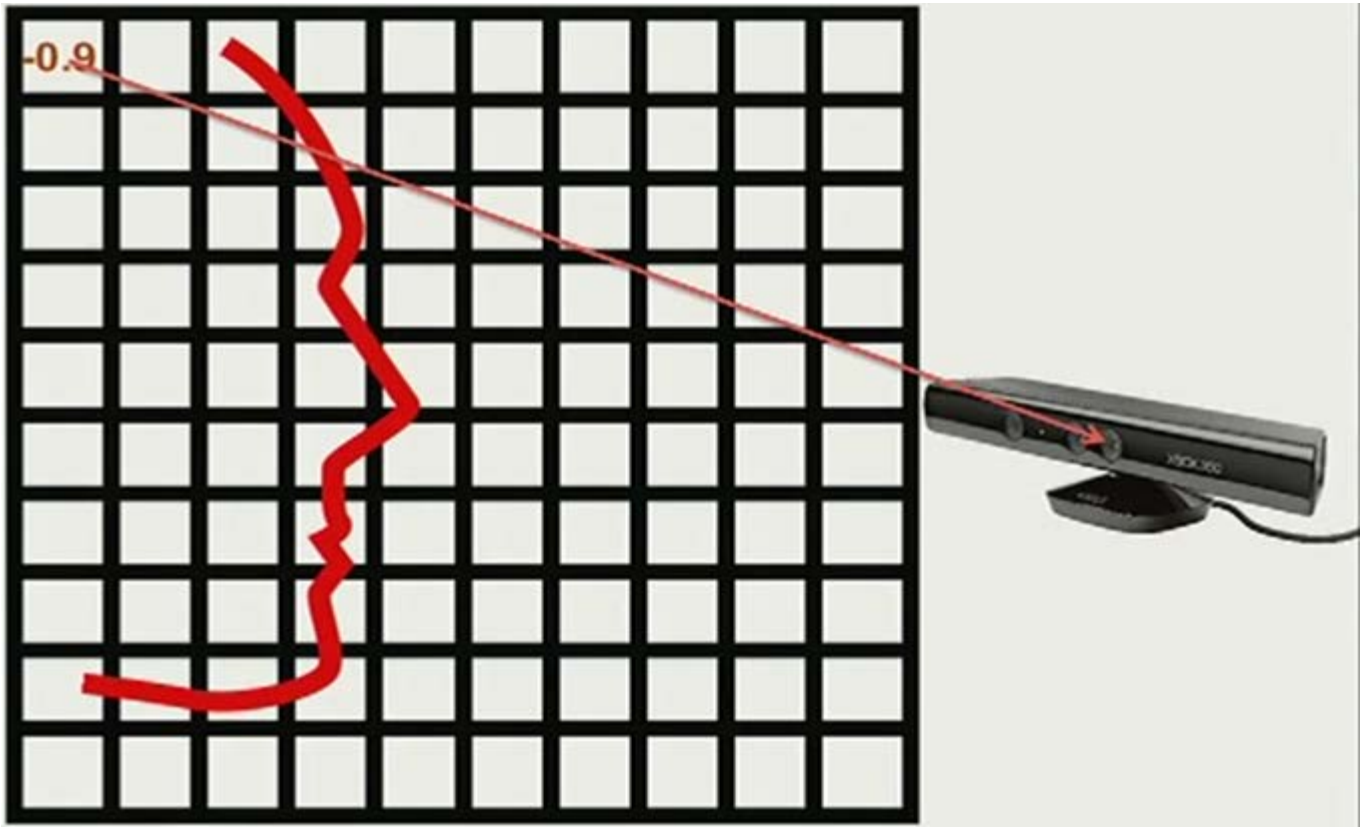
- Numbers in cells measure voxel distance to surface – D

Mapping



Tracking
Mapping

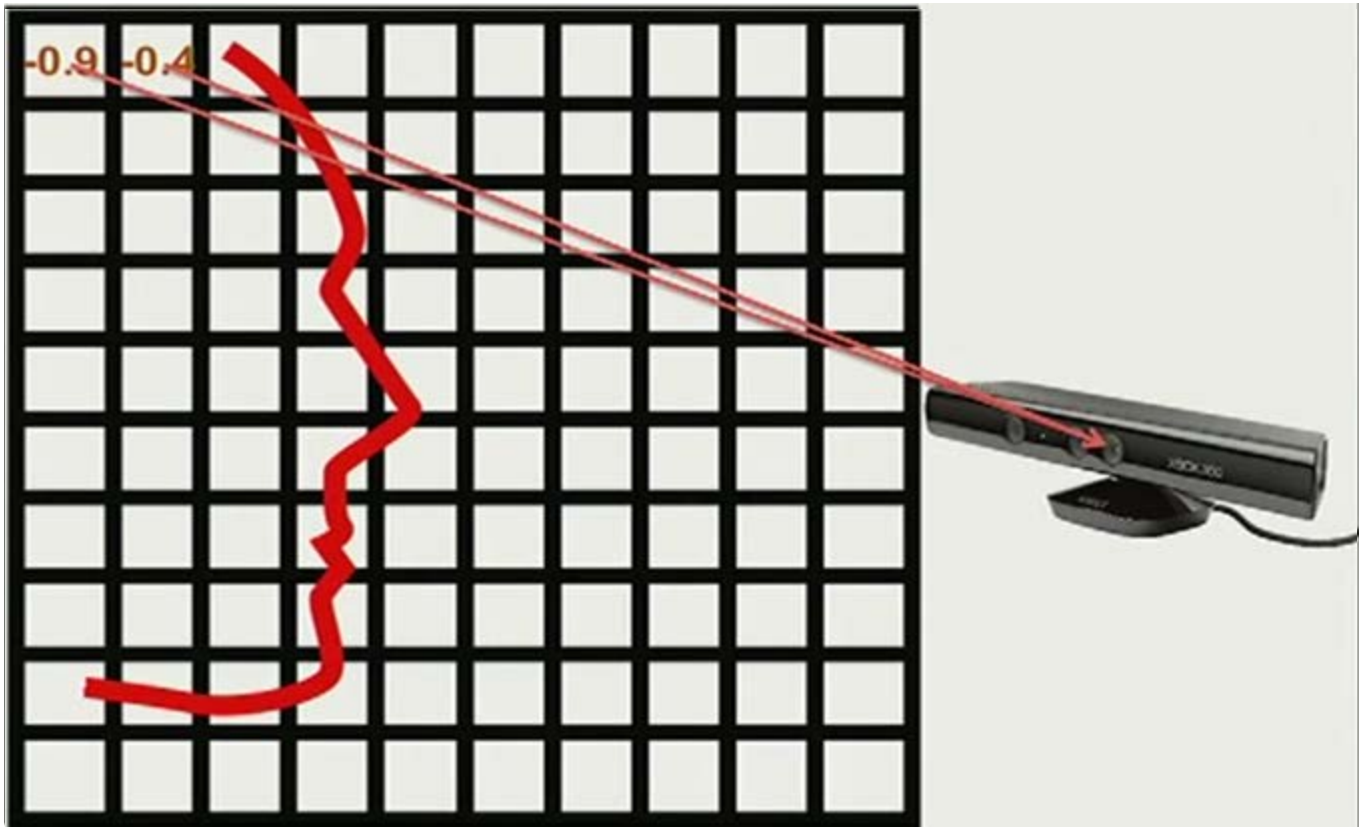
Mapping



Tracking
➢ Mapping

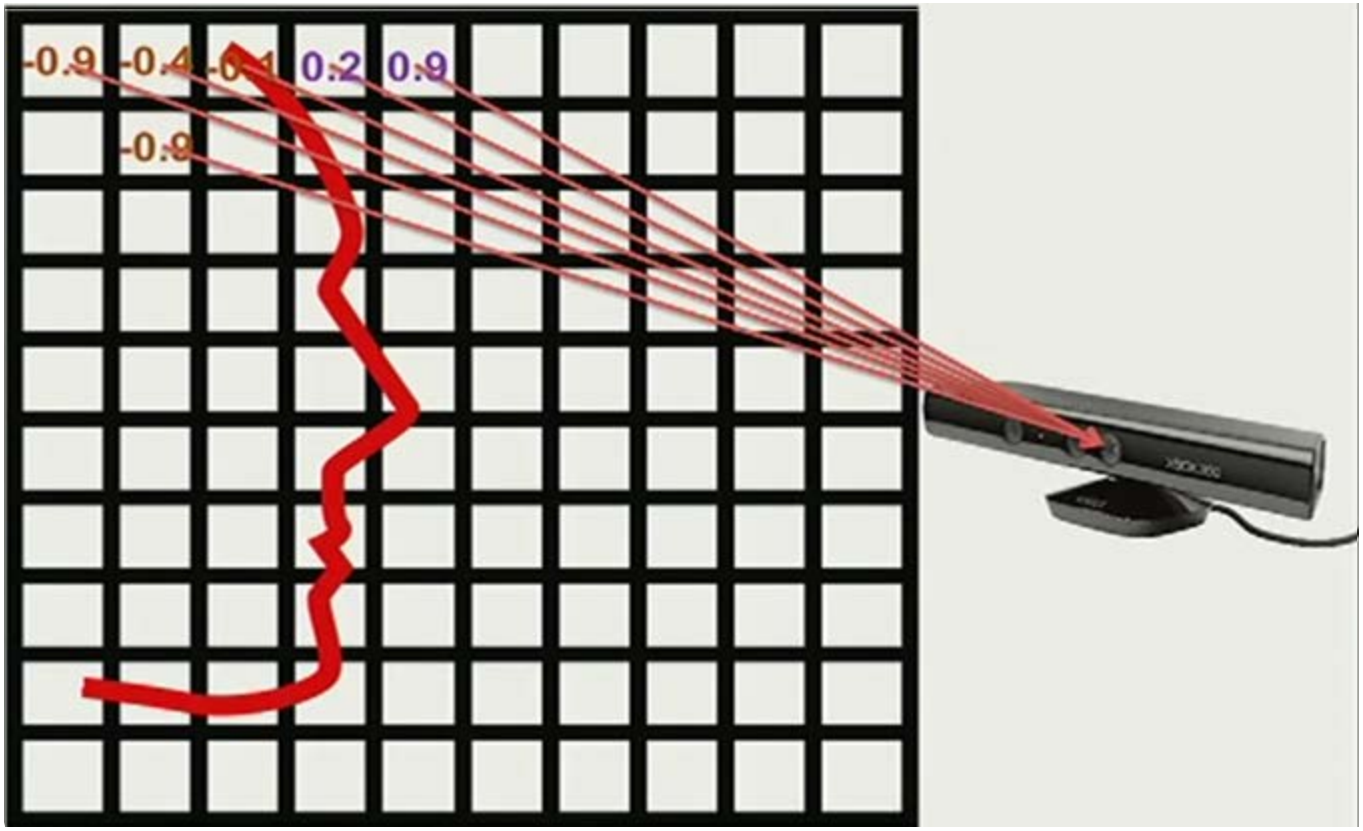
d= [pixel depth] – [distance from sensor to voxel]

Mapping



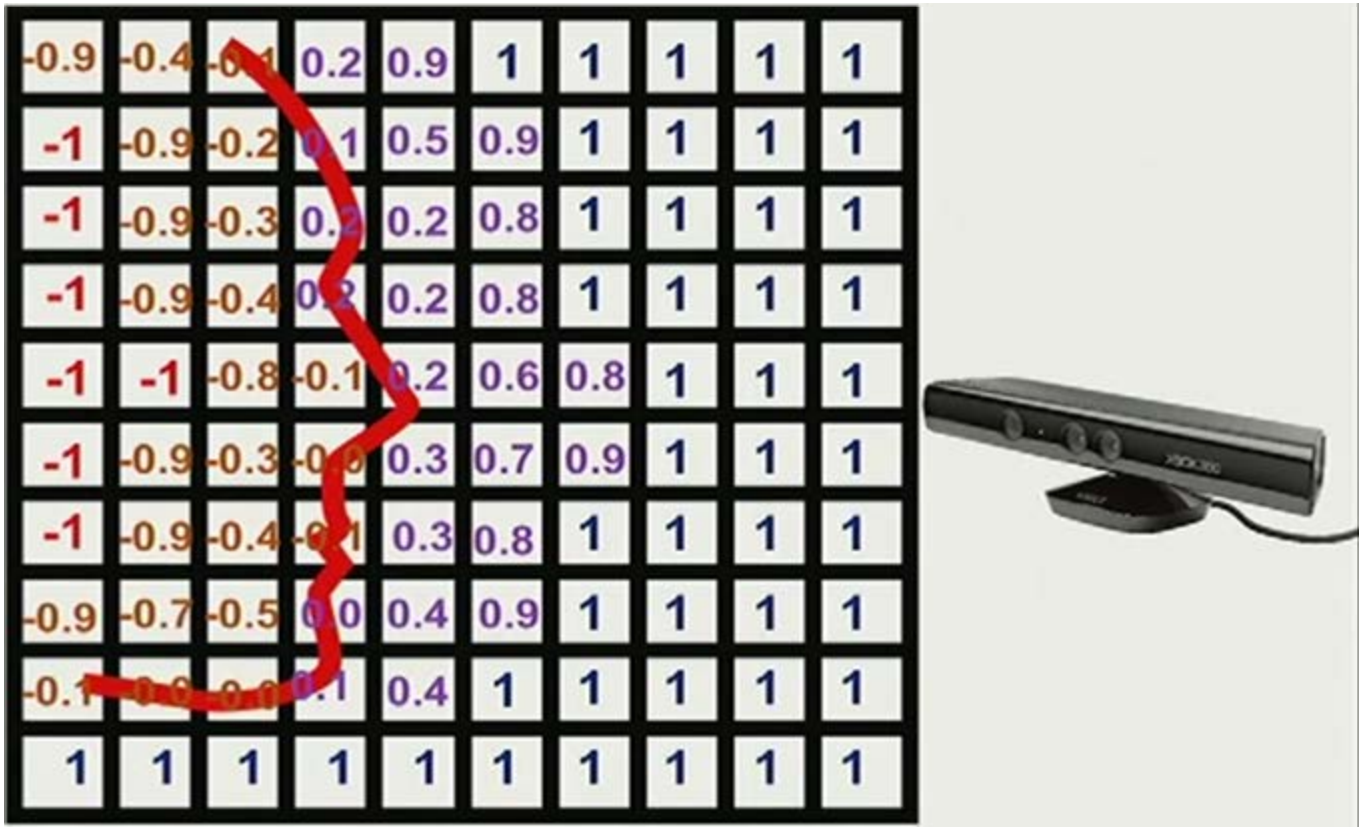
Tracking
Mapping

Mapping



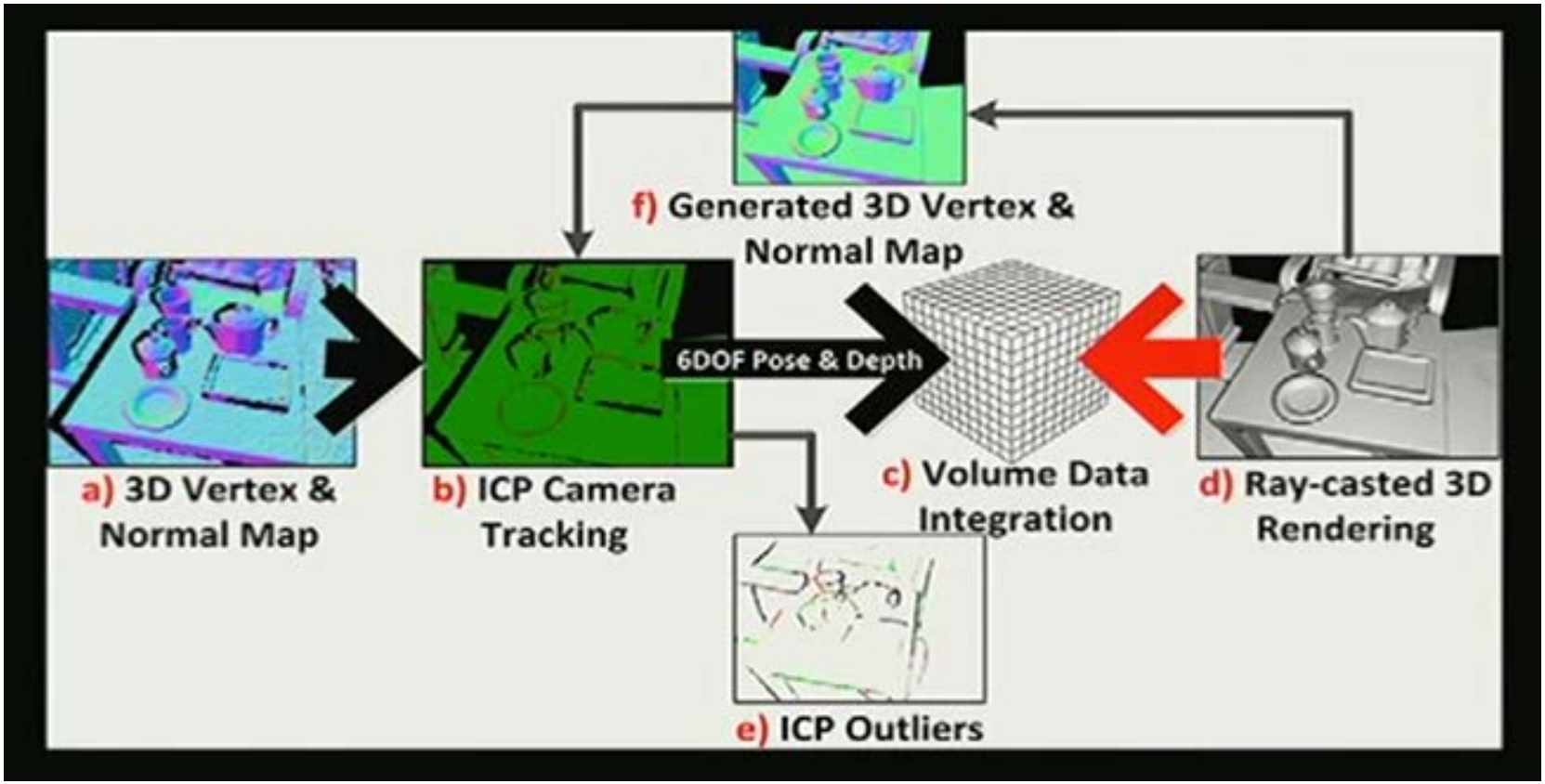
Tracking
Mapping

Mapping



Tracking
Mapping

Method


f) Generated 3D Vertex &
Normal Map
6DOF Pose & Depth
a) 3D Vertex &
Normal Map
b) ICP Camera
Tracking
c) Volume Data
Integration
d) Ray-casted 3D
Rendering
e) ICP Outliers

Pros & Cons

- Pros:
  - Really nice results!
    - Real time performance (30 HZ)
    - Dense model
    - No drift with local optimization
    - Robust to scene changes
  - Elegant solution
- Cons :
  - 3d grid can’t be trivially up-scaled

Limitations

- doesn’t work for large areas (Voxel-Grid)
- Doesn’t work far away from objects (active ranging)
- Doesn’t work out-doors (IR)
- Requires powerful Graphics card
- Uses lots of battery (active ranging)
- Only one sensor at a time