3D Object Detection using Depth Images

Christol Barnes, Sachin Mohla, Pradeep Kumar

Team 6

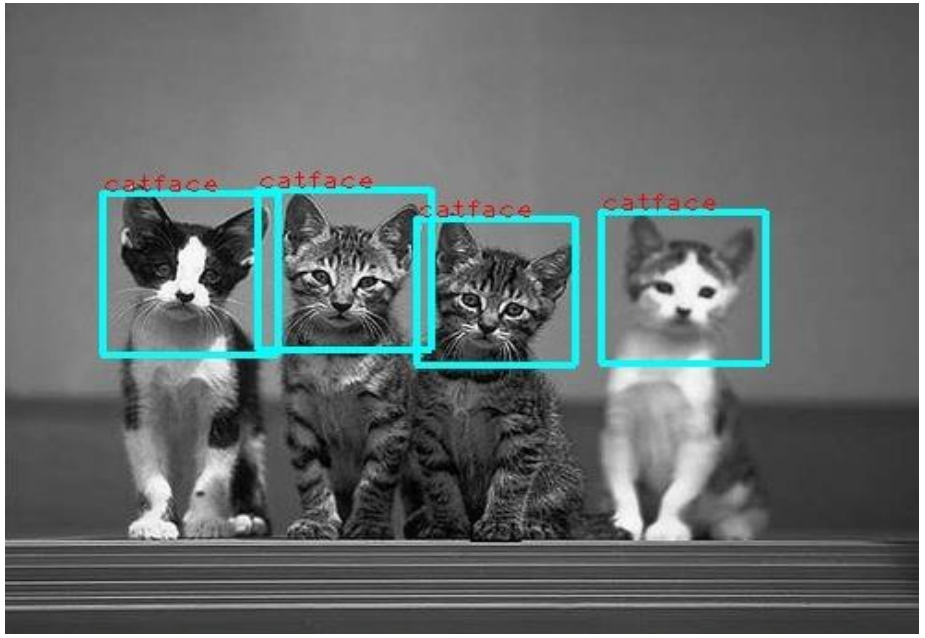
Introduction: Object Detection

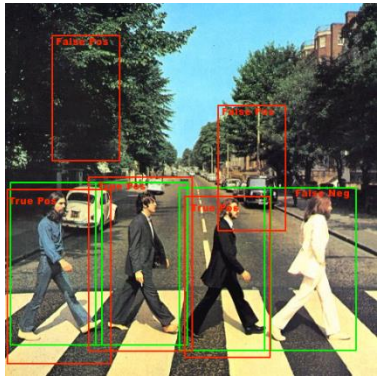
Need for object detection:

- Fundamental problem in CV!
- Application: Robotics, Image retrieval, Video surveillance

Traditional approaches

- Bounding box based
- Deep learning
  - ImageNet





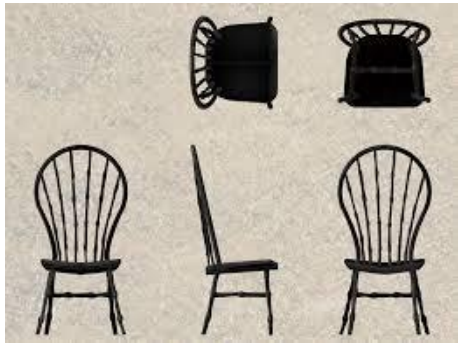
Object Detection is hard!

Variation

Object Detection is hard!

Variation

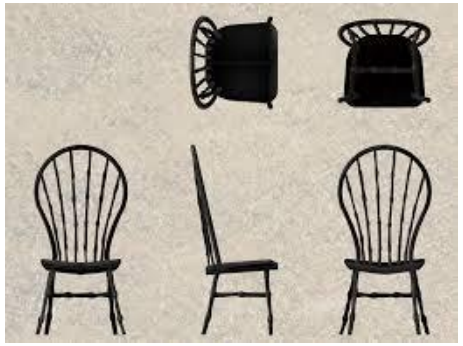
Viewpoint



Object Detection is hard!

Variation

Viewpoint

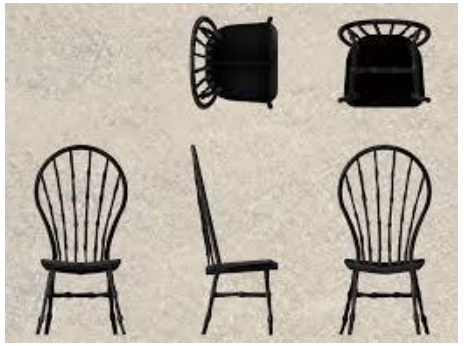


Illumination

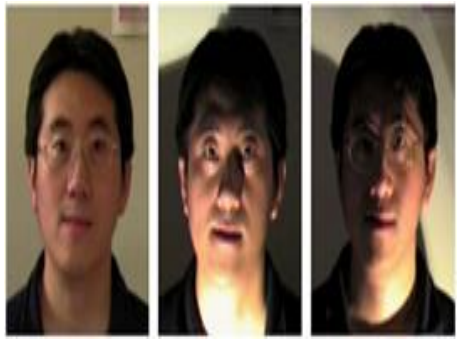
Object Detection is hard!

Variation

Viewpoint



Illumination

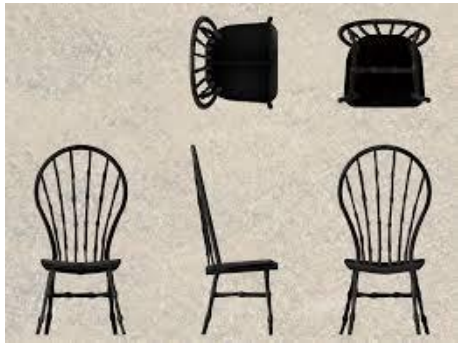


Clutter

Object Detection is hard!

Variation

Viewpoint



Illumination

Clutter

Occlusion

Depth Sensors

Asus Xtion Pro

Our problem statement: Detect objects in a scene using 3D depth images

3D Object Recognition: State of Art

Sliding Shapes

Suran Song, ECCV 2014

Used following features with exemplar SVM:

- Point density: no of points/voxel
- Shape: linearness, scatterness
- Normal: bin the orientation
- TSDF

Accuracy: 0.6 to 0.8

Kernel Descriptors

L. Bo, Dieter Fox, CVPR 2011

Uses 5 depth kernel descriptors that capture:

- Size
- Shape and edges
- Edges

Accuracy: 0.55

Algorithm: Training

- 3D Object detection is highly dependent on pose of the object.
- How do we obtain multiple poses?
- Solution??

Algorithm: Training with 3D CAD Models

- 3D Object detection is highly dependent on pose of the object.
- How do we obtain multiple poses?
- Solution??
- **Use CAD Models**
- Downloaded from Trimble 3D warehouse* and other freely available models
- Render the models(.ply) and capture point clouds of partial views

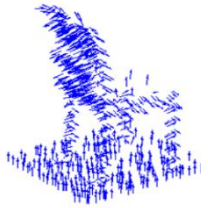
*https://3dwarehouse.sketchup.com/

Algorithm: Features

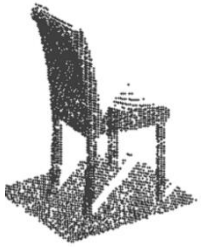
- Required: Set of correspondences between test point cloud and known cloud
- Features:
    - Compact, but rich representation of 3D data
    - Needs to be invariant to
        - Transformations and disturbances
        - Noise
        - Resolution invariant

Algorithm: Features

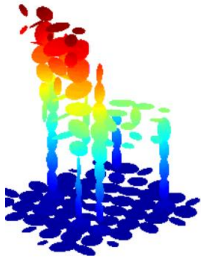
- Required: Set of correspondences between test point cloud and known cloud
- Features:
  - Compact, but rich representation of 3D data
  - Needs to be invariant to
    - Transformations and disturbances
    - Noise
    - Resolution invariant
- Local Descriptors v/s Global Descriptors?



Normal



Points



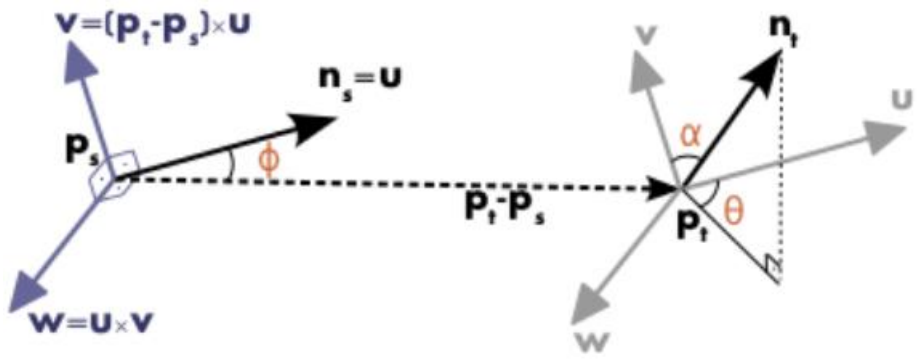
Shape

Algorithm: Point Feature Histogram

- PFH computes 3 values for each pair in the neighborhood
  - Captures information of the geometry surrounding the point
  - Finds the difference between directions of normals
- For each pair, computes a local reference frame(LRF) centered on p_s as:
  - The normal $u = n_s$
  - Cross product between n_s and the vector $(p_t - p_s)$ $v = n_s \times (p_t - p_s)$
  - Cross product between previous vectors $w = u \times v$
- Computes and accumulates

$$\alpha = cos^{-1}(v.n_t) \;\; ; \;\; \phi = cos^{-1}\left(u.\frac{(p_t - p_s)}{\|p_t - p_s\|_2}\right)$$

$$\theta = tan^{-1}(w.n_t, u.n_t)$$



Algorithm: Point Feature Histogram

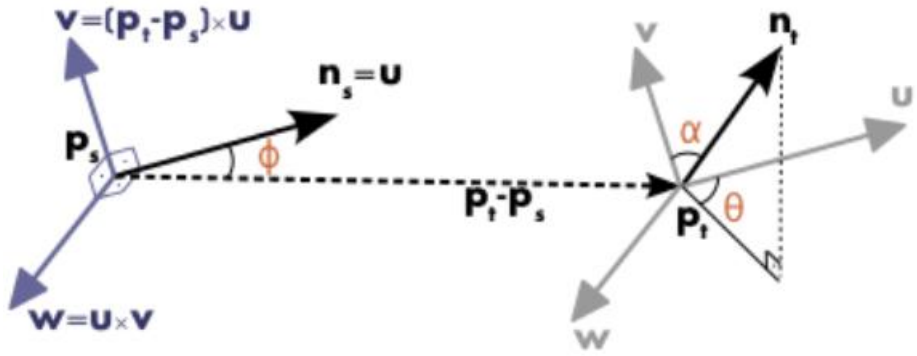
- PFH computes 3 values for each pair in the neighborhood
  - Captures information of the geometry surrounding the point
  - Finds the difference between directions of normals
- For each pair, computes a local reference frame(LRF) centered on p_s as:
  - The normal $u = n_s$
  - Cross product between n_s and the vector (p_t - p_s) $v = n_s \times (p_t - p_s)$
  - Cross product between previous vectors $w = u \times v$
- Computes and accumulates

$$\alpha = cos^{-1}(v.n_t) \;\; ; \;\; \phi = cos^{-1}\left(u.\frac{(p_t - p_s)}{\|p_t - p_s\|_2}\right)$$

$$\theta = tan^{-1}(w.n_t, u.n_t)$$

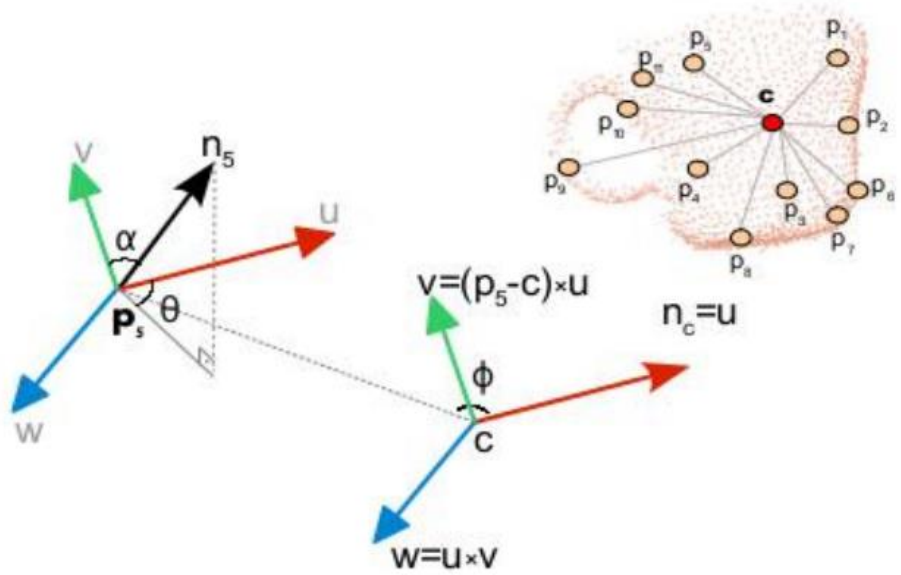
Extremely slow!

For *n* key points with *k* neighbors, complexity = $O(nk^2)$



Algorithm: Viewpoint Feature Histogram

- **Global Descriptor**
- Described by Rusu *et al.**
- Each view of the object provides a different descriptor
  - **Scale Invariant,** but encodes the viewpoint from where surface was captured
- For each pair ($\boldsymbol{p_c}$, $\boldsymbol{p_i}$):
  - Compute LRF for the centroid:
    - $u = n_s$
    - $v = \frac{p_i - p_c}{\|p_i - p_c\|} \times u$
    - $w = u \times v$
  - $\alpha = cos^{-1}(v.n)$
  - $\phi = cos^{-1}\left(u.\frac{(p_i - p_s)}{\|p_i - p_s\|_2}\right)$
  - $\theta = tan^{-1}(w.n, u.n)$



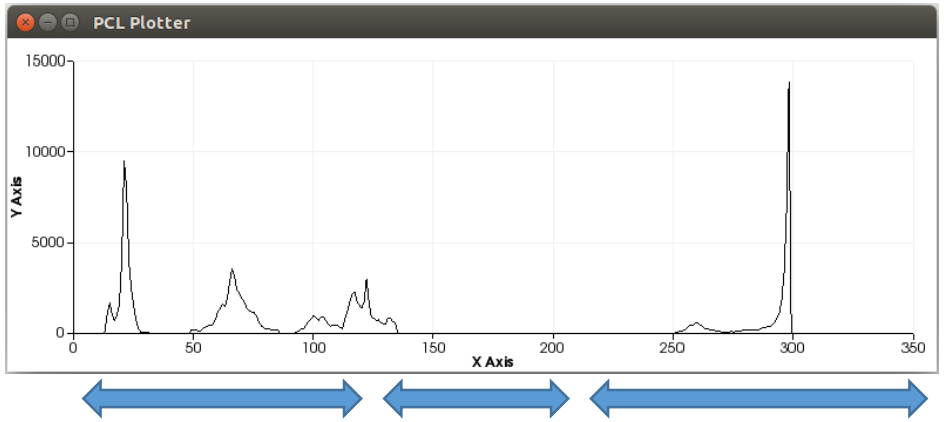
*Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram, Rusu *et al.*, in Proc. Int. Conf. Intelligent Robots and Systems, 2010, pp. 2155–2162.

Algorithm: Viewpoint Feature Histogram

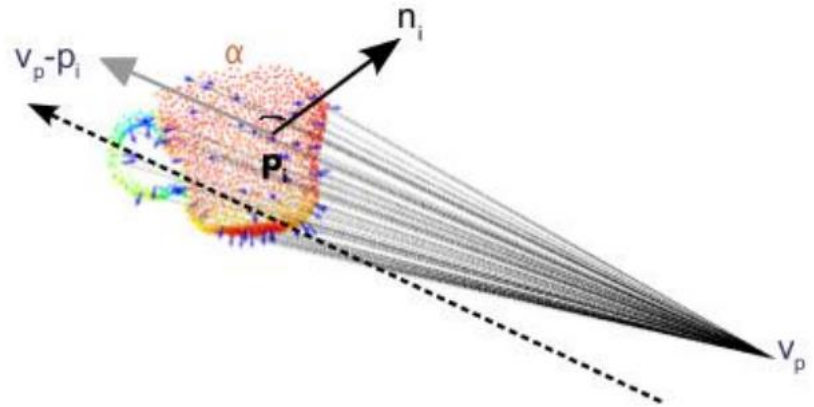
- Descriptor composition
  - 3 PFH angular values (α, θ, ϕ) with respect to centroid of object(45 bins each)
  - 1 shape distribution component with respect to centroid (45 bins)

$$SDC = \frac{(p_c - p_i)^2}{max((p_c - p_i)^2)}$$

  - 1 angular value which encodes angle b/w normal and central view direction α (128 bins)



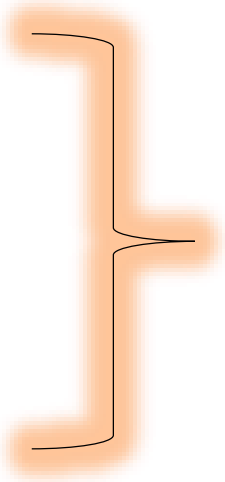
FPFH (α, θ, ϕ) Shape dist Viewpoint α



Algorithm: Global Radius-Based Surface Descriptor

- Described by Marton *et al.**
- Global feature is computed from each cluster by counting transitions between surface types
- Categorizes the point clouds into following shapes:
  - Cylindrical : minRadius = radius of cylinder, maxRadius = ∞
  - Spherical/Corner : minRadius and maxRadius are same
  - Planar
  - Noise/Corner
  - Edge

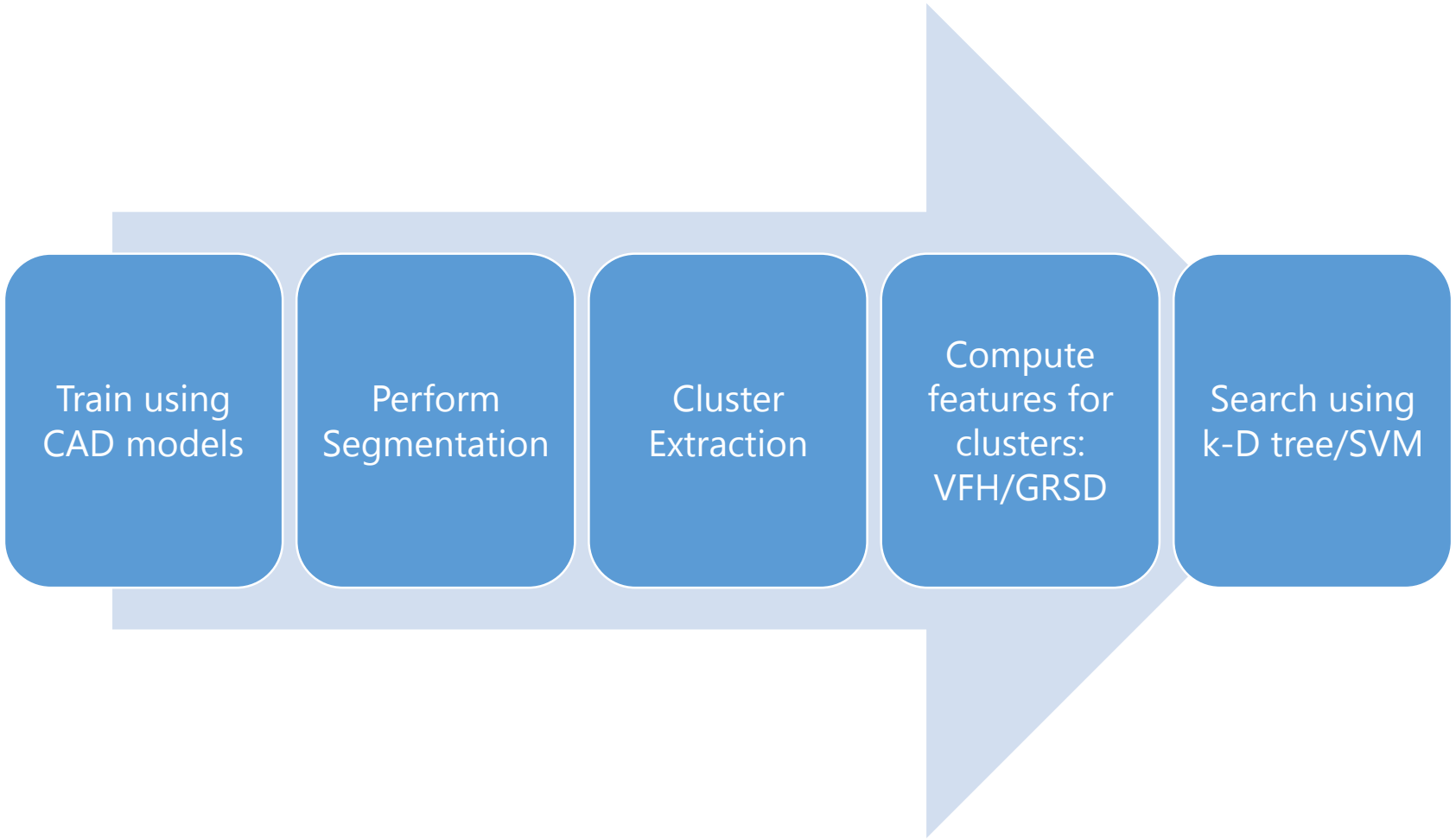
Based on minRadius, maxRadius



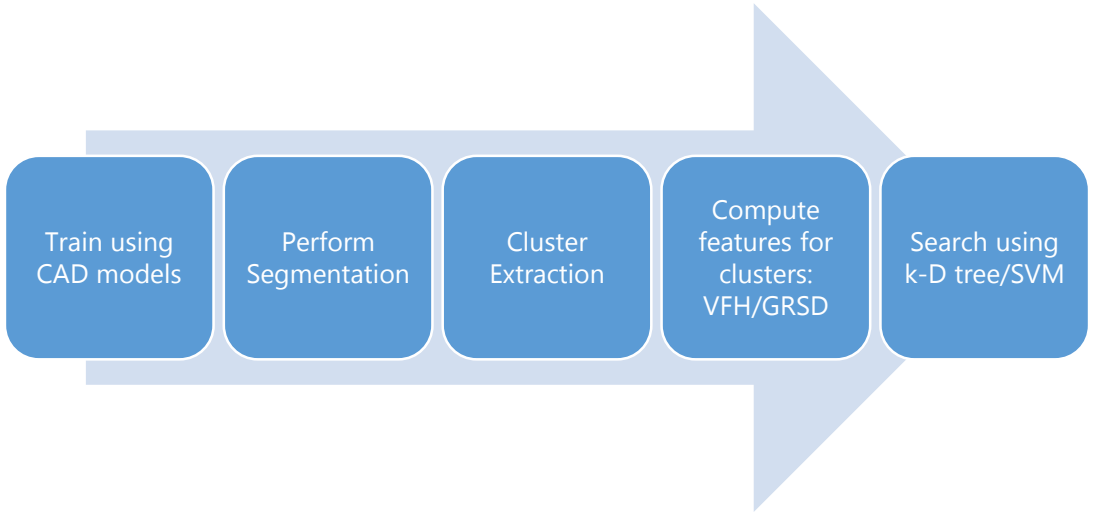
Use geometric class to decide on what model to fit.

*General 3D Modelling of Novel Objects from a Single View, Marton *et al.*, in IROS, 2010

Object Recognition Pipeline

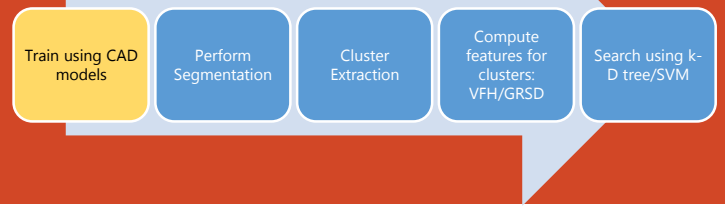


Object Recognition Pipeline: Platform

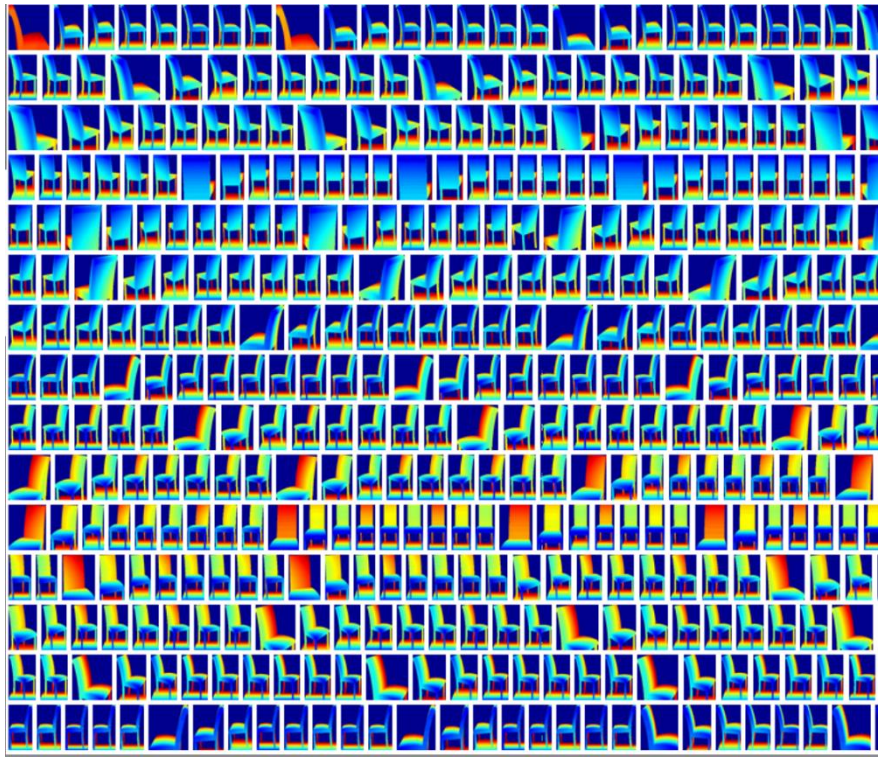


- Point Cloud Library implemented in C++
- Hardware: XBOX 360 Kinect
- OpenNI for Kinect image capture
- KD Tree : FLANN implementation

Object Recognition Pipeline: Training



- Generated 80 views using different angles
- Object categories comprised:
  - Chair
  - Coffee Mug
  - Bottle
  - Ball
  - Bowl
- Used Blender and Meshlab for editing and creating CAD dataset



Object Recognition Pipeline: Training

Segmentation process:

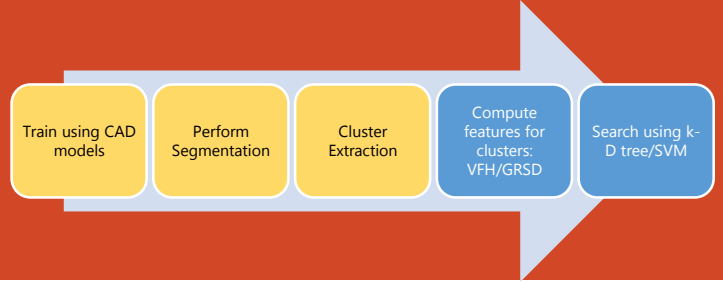
- SAC Segmentation: (**SA**mple **C**onsensus based)
  - Fit plane model
  - With constraints = orientation

- Once, plane is extracted:
  - Find objects standing on tables, shelves
  - Protruding objects such as door handles

  By

- Polygonal prism method:
  - Using convex hull of planar points
  - Extruding this outline along plane normal

Object Recognition Pipeline: Cluster Extraction



- Once the objects are detected
  - Need to segment remaining point cloud into clusters
  - Idea: Use region growing approach that cannot grow two points with a high distance
    - This merges locally dense areas and splits separate clusters
  - Basically, **Euclidean distance** based clustering
  - Up-sample and down-sample
    - to obtain clusters with different point cloud densities

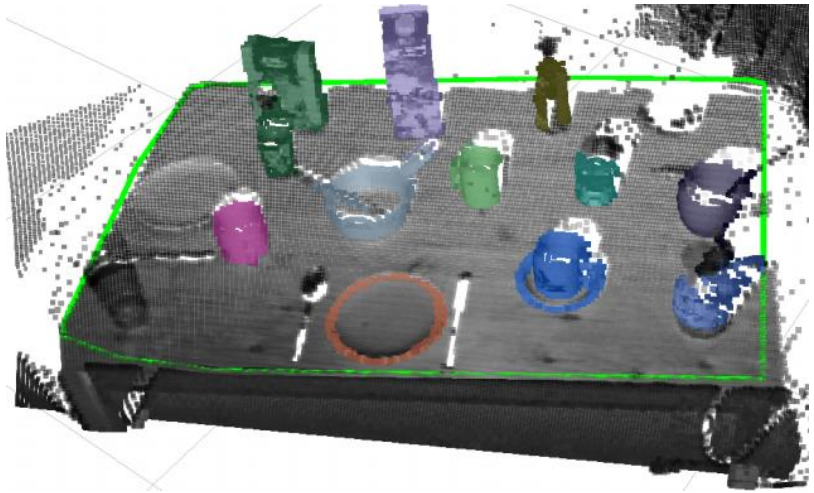
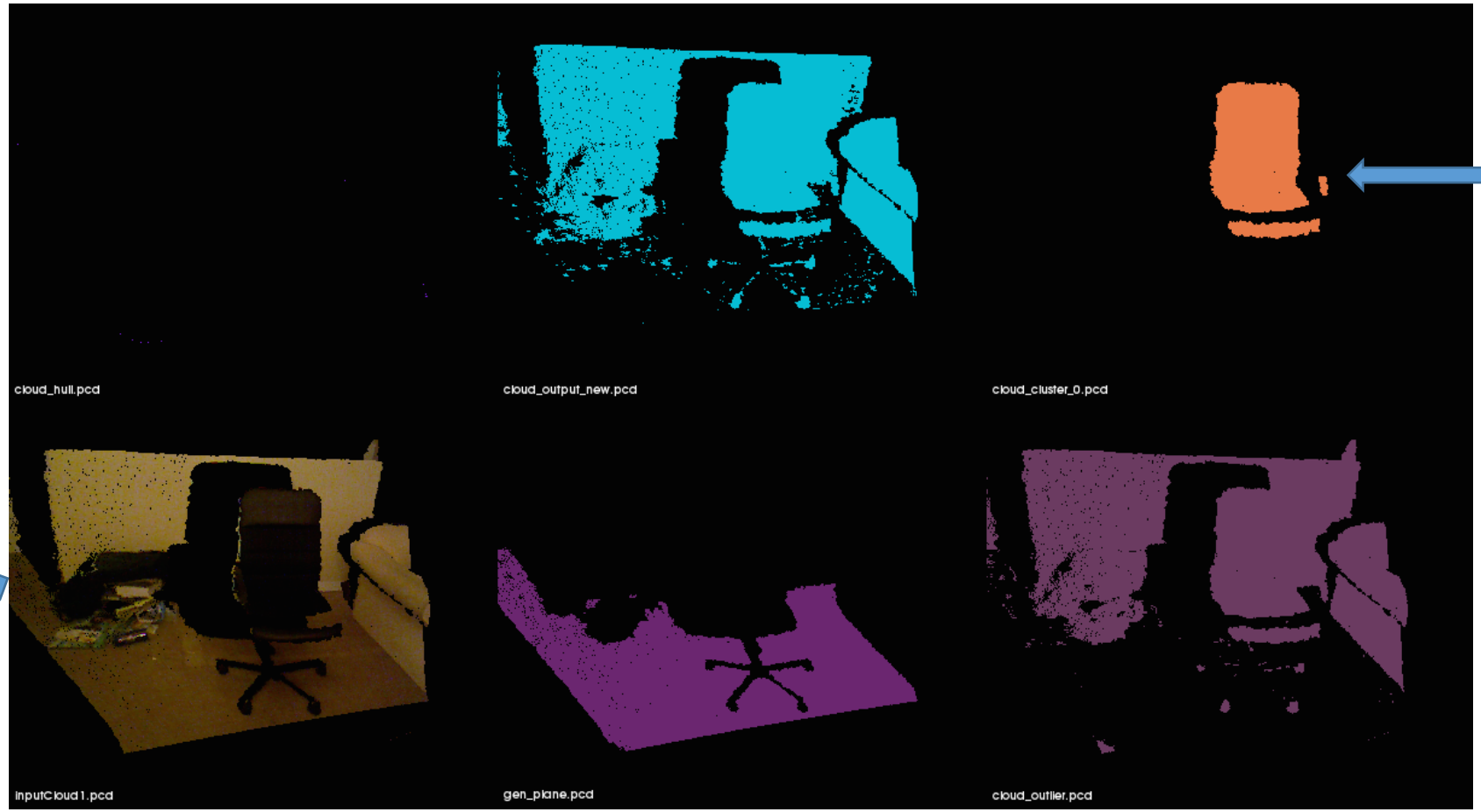


Table top clusters obtained after plane extraction

Object Recognition Pipeline: Cluster Extraction

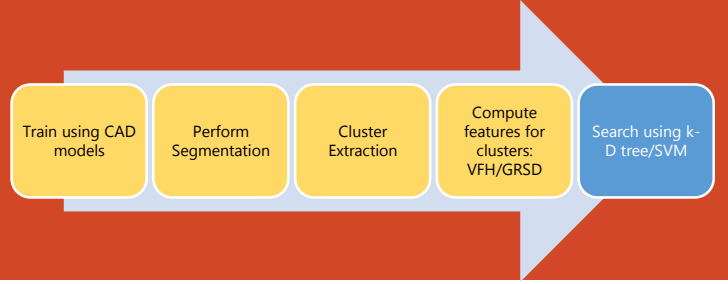
- Result



Final Segmented cluster

Input depth image

Object Recognition Pipeline: Compute Features

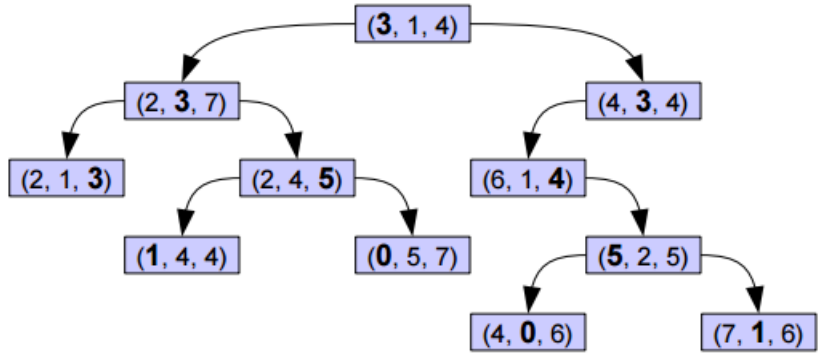


- Compute for each point cloud:
  - VFH/GRSD features
  - Single view might not provide differentiating features:
    - Perform feature extraction/recognition at regular intervals

Object Recognition Pipeline: Compute Features

- Once feature descriptors are computed for each clustered point cloud:
  - **Nearest neighbor search**
    - Run k-D tree based nearest match based on histogram distance
    - Histogram distance is calculated based on:
      - Chi-square metric

$$d(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i)}$$



  - **One v/s all -SVM**
    - Computed using libSVM library
    - Perform this for all classes

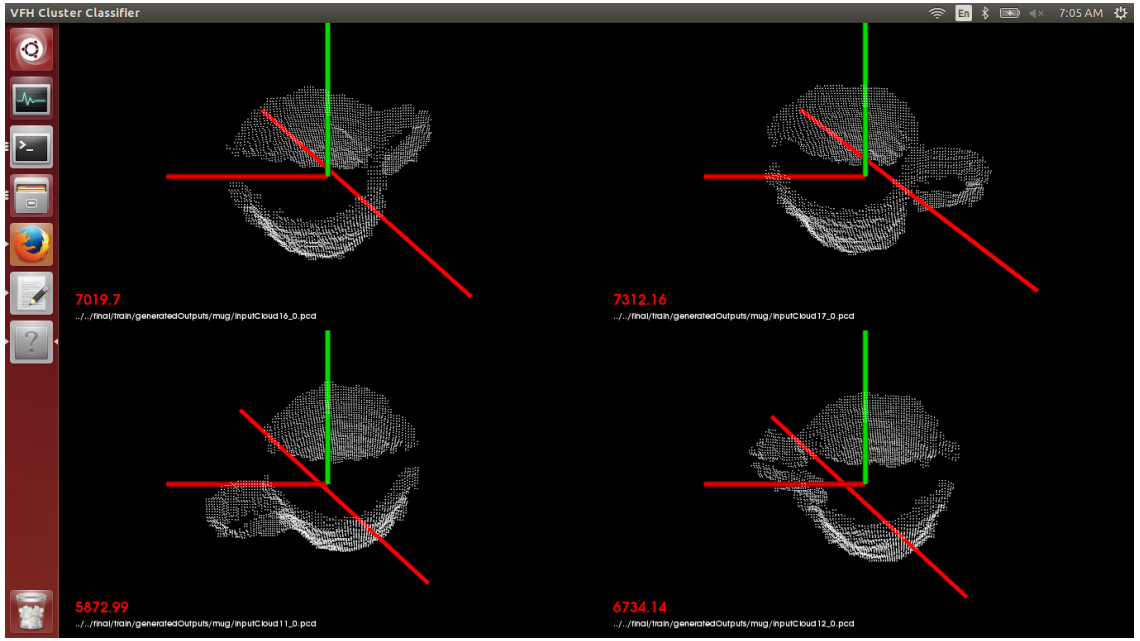
Demo: Segmentation

- Play video!

Demo: Object Recognition

- Using Kinect

Results

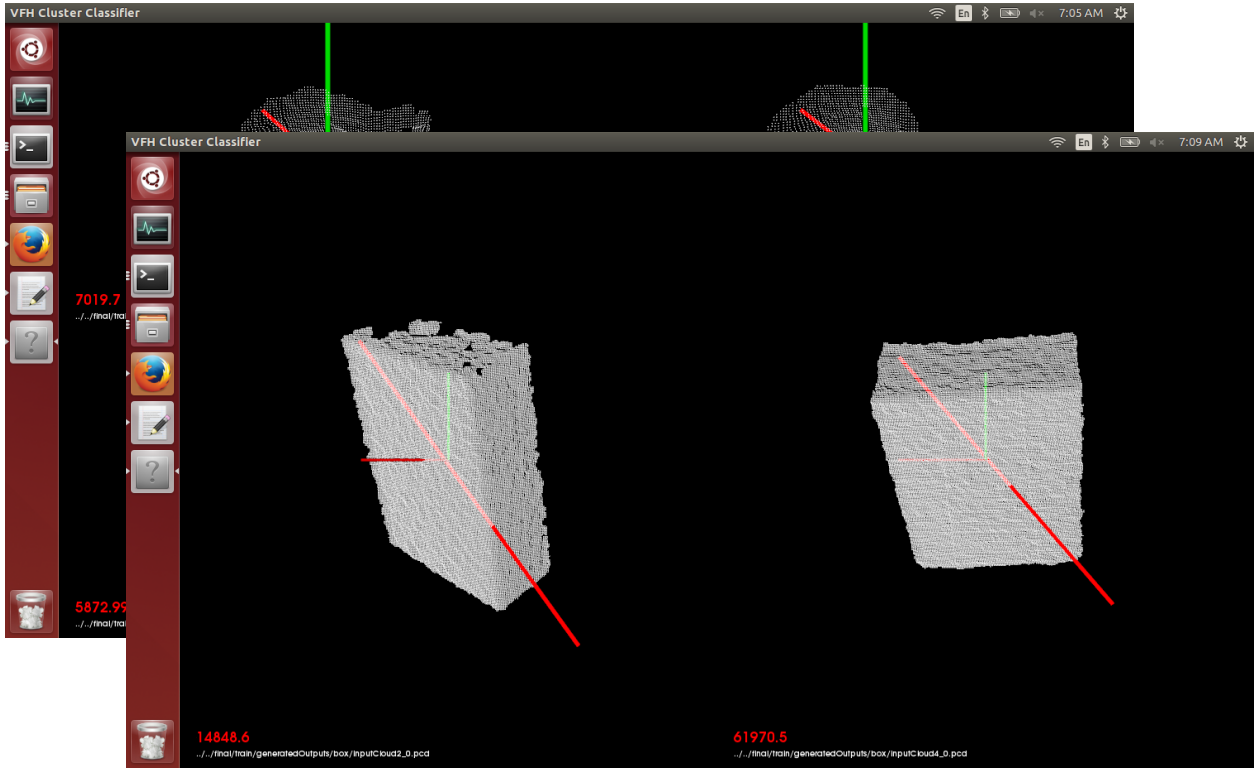


Matching obtained for Coffee Mug

Challenges

- Kinect dataset is too small
  - Compared to 2D datasets available online
- Kinect Data: Too noisy
- Need exact 3D models
- Need powerful segmentation
- One v/s all SVM wasn't good enough classifier
  - Only polynomial kernel provided good results

Results

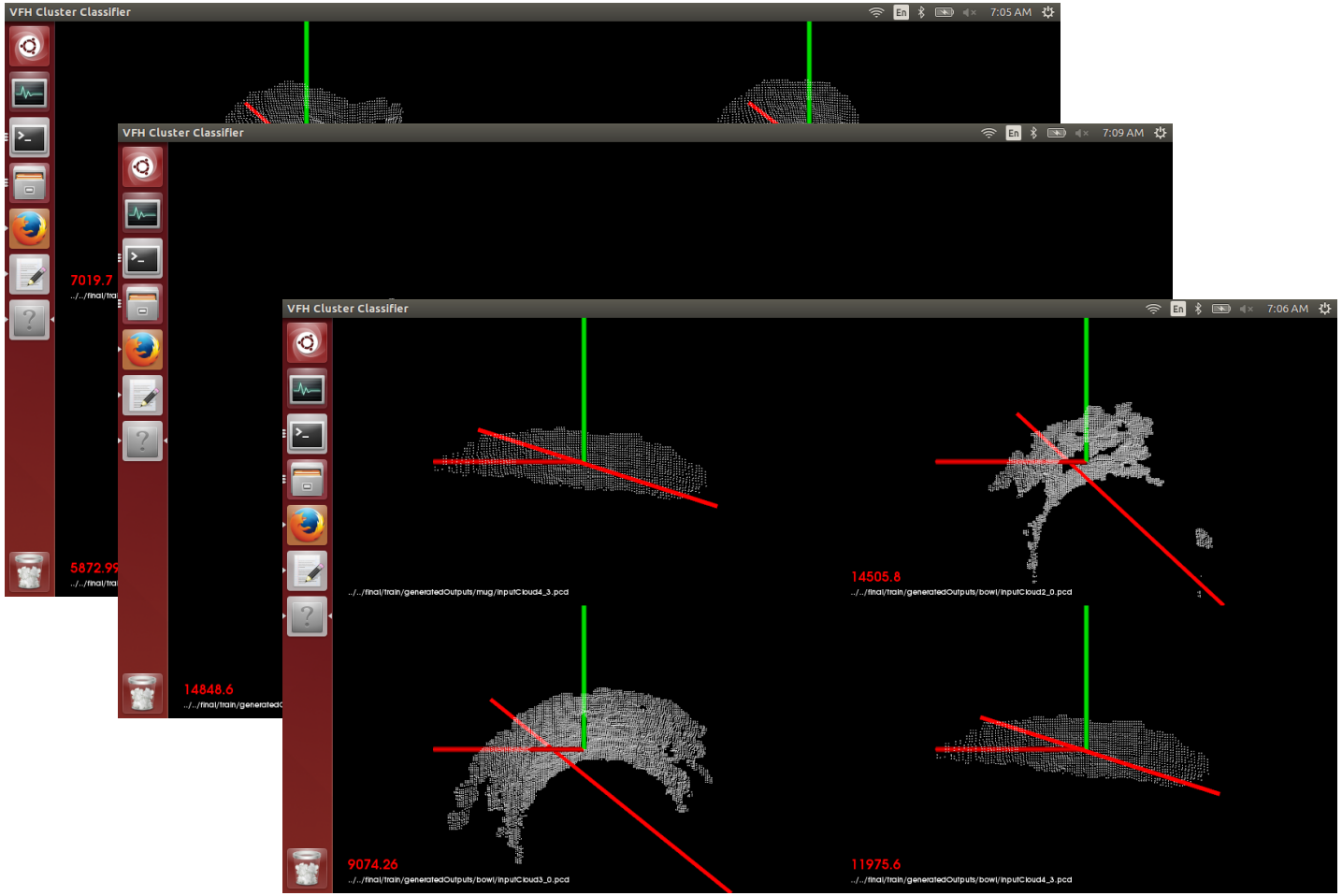


Matching obtained for Cereal Box

Challenges

- Kinect dataset is too small
  - Compared to 2D datasets available online
- Kinect Data: Too noisy
- Need exact 3D models
- Need powerful segmentation
- One v/s all SVM wasn't good enough classifier
  - Only polynomial kernel provided good results

Results



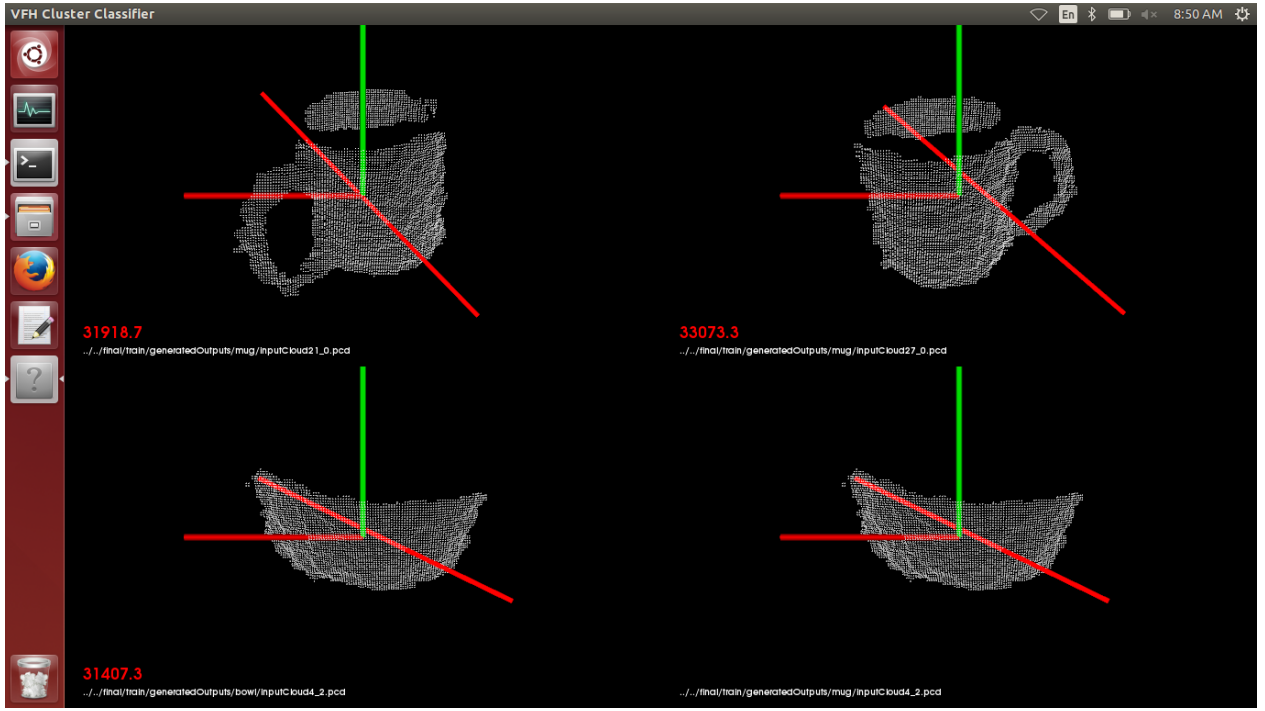
Matching obtained for Bowl

Challenges

- Kinect dataset is too small
  - Compared to 2D datasets available online
- Kinect Data: Too noisy
- Need exact 3D models
- Need powerful segmentation
- One v/s all SVM wasn't good enough classifier
  - Only polynomial kernel provided good results

Results: False positives

- Objects with similar shapes were wrongly classified
- Example: Bowl and Coffee Mug



Future Work

- Use exemplar-SVM for classification as it has been proven
- Use better 3D dataset
- Use texture(color) information
- Use a sliding bounding box(as proposed by Song)
- Perform correspondence grouping
    - To fit best available model available

Thank you!