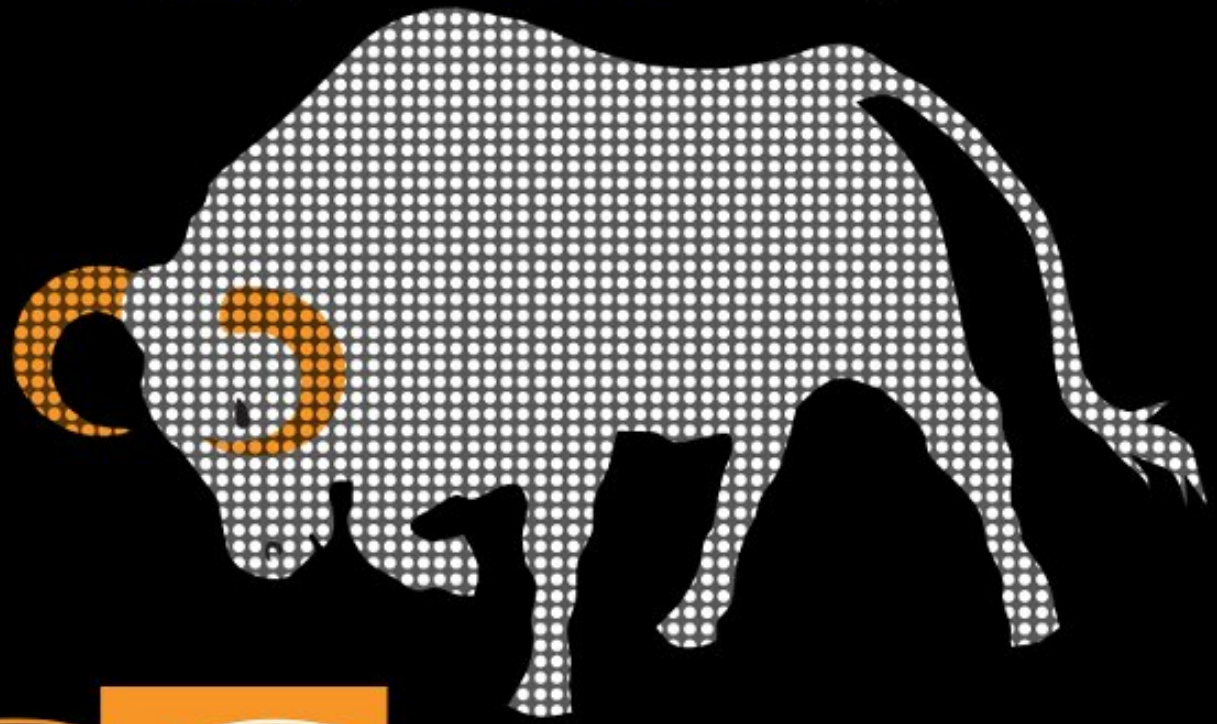
pointcloudlibrary



ICCV2011 BARCELONA. SPAIN.

Object Recognition, Classification and Evaluation with PCL

November 6, 2011

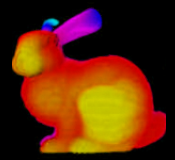
Outline

Object Recognition

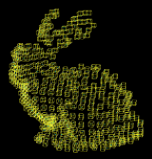
Can we use PCL for object recognition?

filters

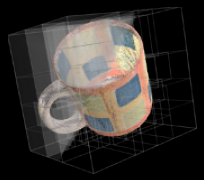
io



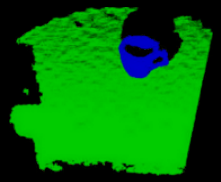
visualization



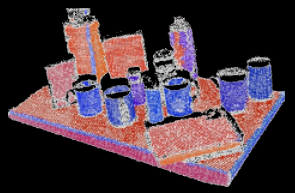
octree



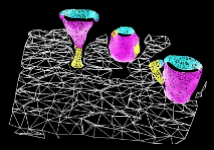
search



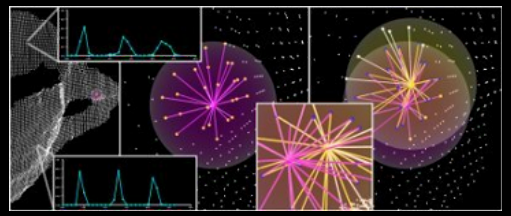
segmentation



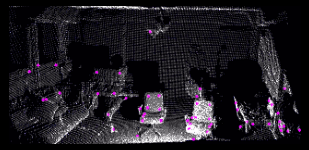
sample_consensus



surface



features



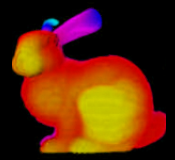
keypoints

Object Recognition

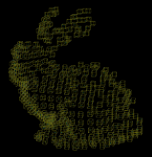
Can we use PCL for object recognition? Yes, we can

filters

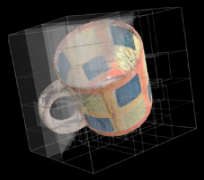
io



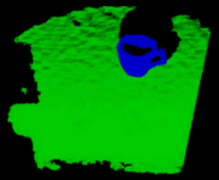
visualization



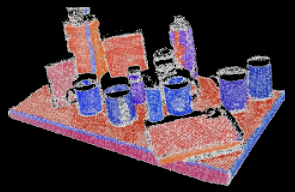
octree



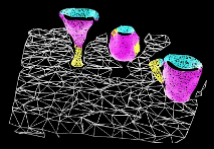
search



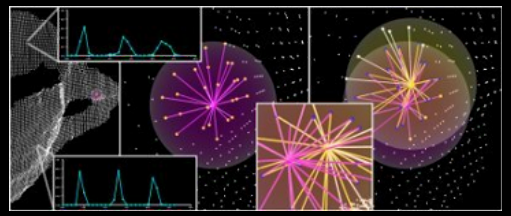
segmentation



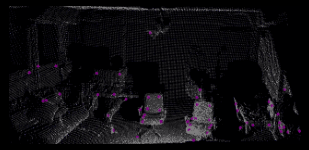
sample_consensus



surface



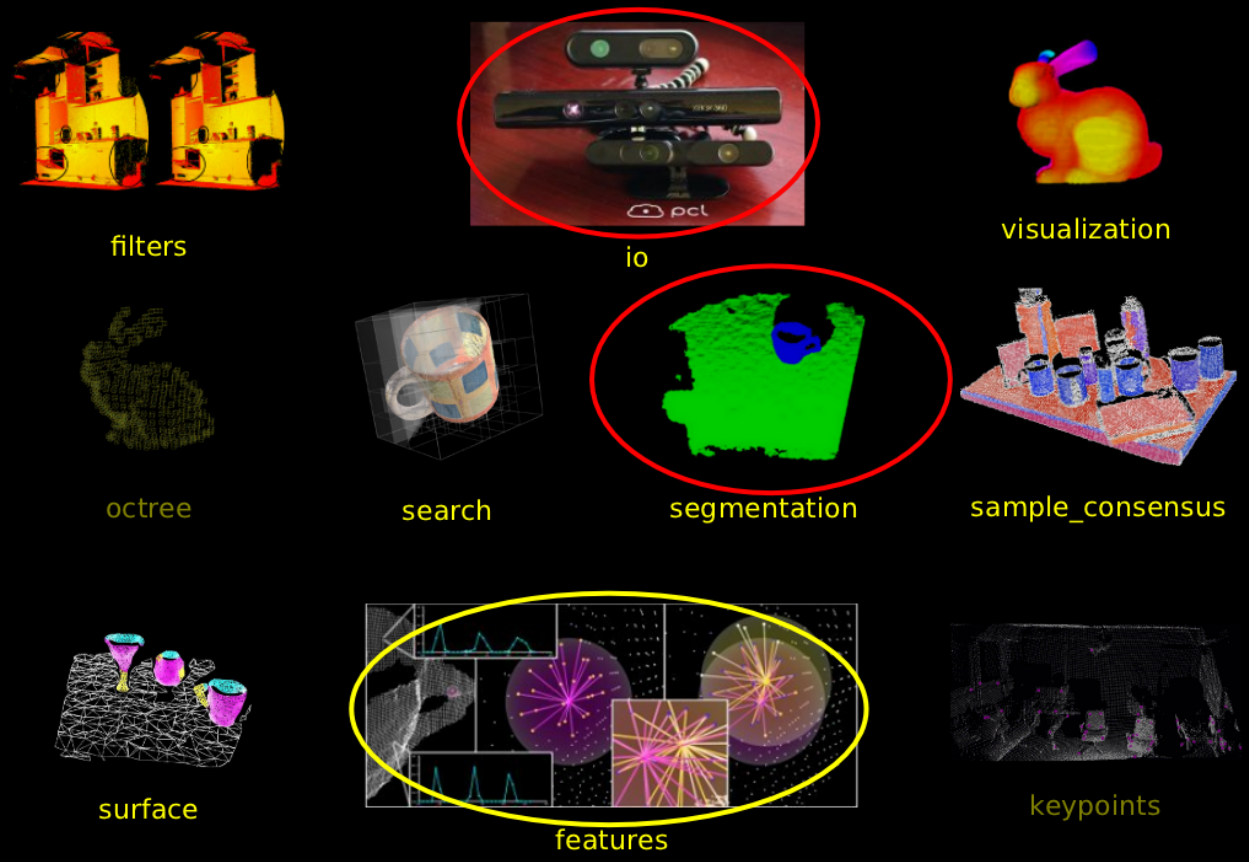
features



keypoints

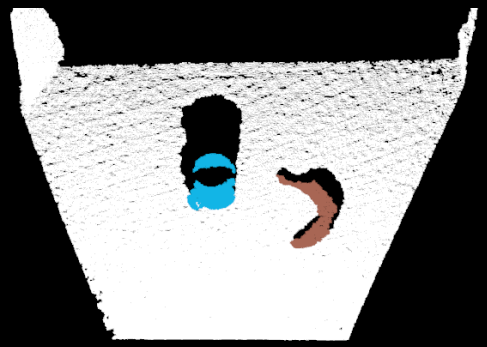
Object Recognition

Building a PCL based object recognition app



pcl::apps::DominantPlaneSegmentation

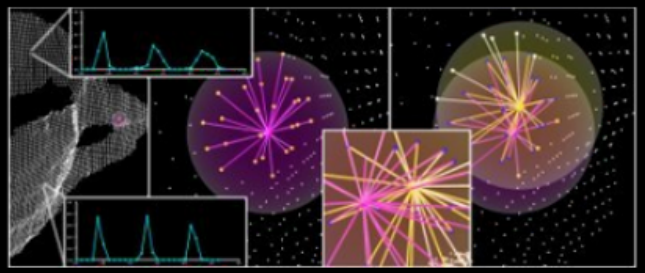
Segmentation of a Kinect scene



```
typedef pcl::PointCloud<OpenNIFrameSource::PointT> Cloud;
std::vector<typename Clout::Ptr>, ... > clusters;
frame = camera.snap();

pcl::apps::DominantPlaneSegmentation<OpenNIFrameSource::PointT> dps;
dps.setInputCloud (frame);
dps.setMaxZBounds (1.0);
dps.setObjectMinHeight (0.005);
dps.setMinClusterSize (1000);
dps.setWSize(3);
dps.setDistanceBetweenClusters(0.1);
dps.setDownsamplingSize (0.02);
dps.compute_fast (clusters);
```

Object Recognition

What else do we need? Descriptors



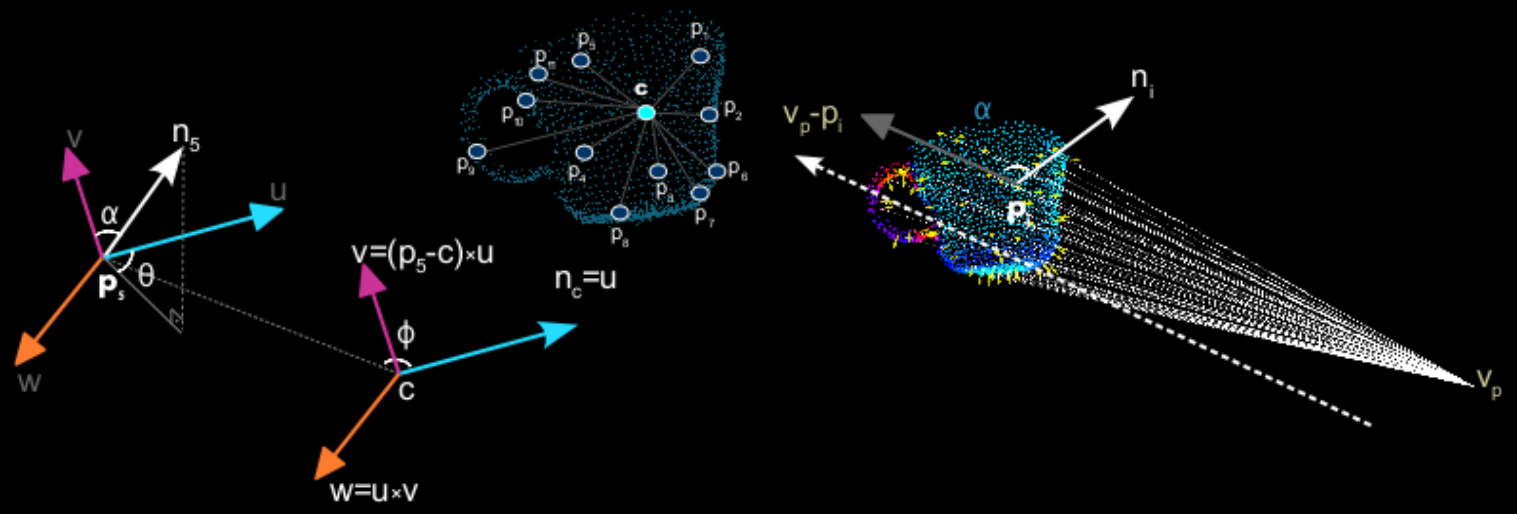
features

- Encode whole object with descriptor (globally)
  - VFH
  - CVFH
  - SHOT
  - SpinImage
  - ShapeContext
  - ShapeDistributions
  - ...

VFH

Viewpoint Feature Histogram

- Encodes the surface of the object and the viewpoint
  - Relative to the centroid and the average of the normals
- Efficient to compute



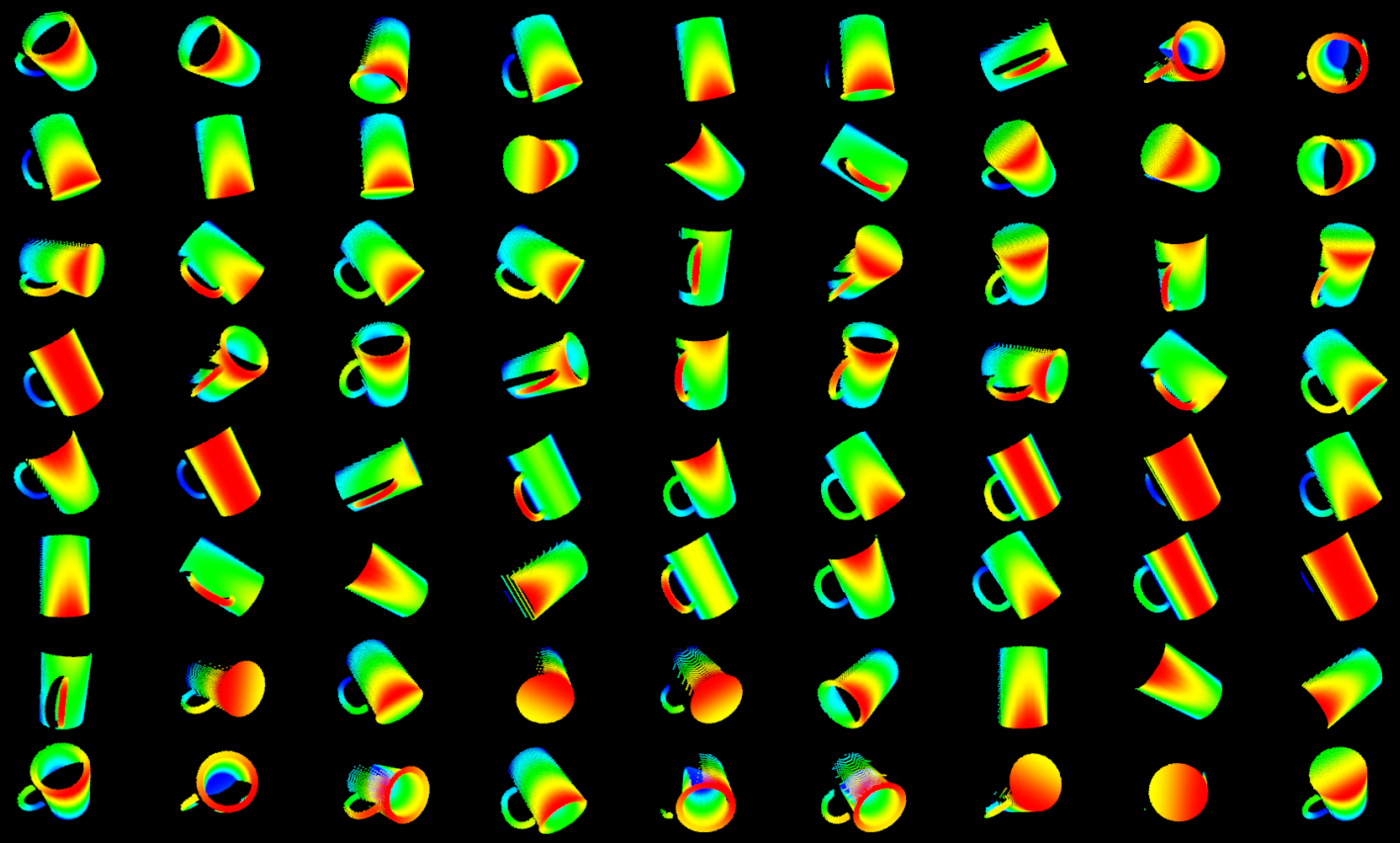
VFH Estimation in PCL

```
using namespace pcl;
typedef PointCloud<PointXYZ> CloudXYZ;
typedef PointCloud<Normal> CloudNormal;
typedef PointCloud<VFHSignature308> VFHSig;
typedef CloudXYZ::Ptr CloudXYZPtr;
typedef CloudNormal::Ptr CloudNormalPtr;

CloudXYZPtr cloud(new CloudXYZ());
CloudNormalPtr normals(new CloudNormal());
// Compute normals
...

// Compute VFH
VFHEstimation<PointXYZ, Normal, VFHSignature308> vfh;
vfh.setInputCloud (cloud);
vfh.setInputNormals (normals);
KdTreeFLANN<PointXYZ>::Ptr tree (new KdTreeFLANN<PointXYZ> ()
);
vfh.setSearchMethod (tree);
VFHSig::Ptr vfhs (new VFHSig ());
// Compute the features
vfh.compute (*vfhs);
```

Training data

What else do we need? Training data

- Given partial views from the Kinect
  - we also want views for training



Training data

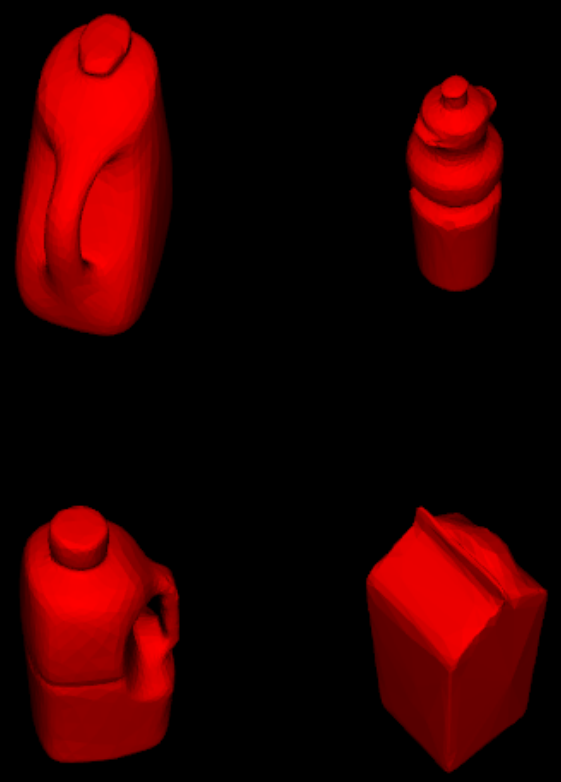
Sources:

- Capturing views with real sensors

Training data

Sources:

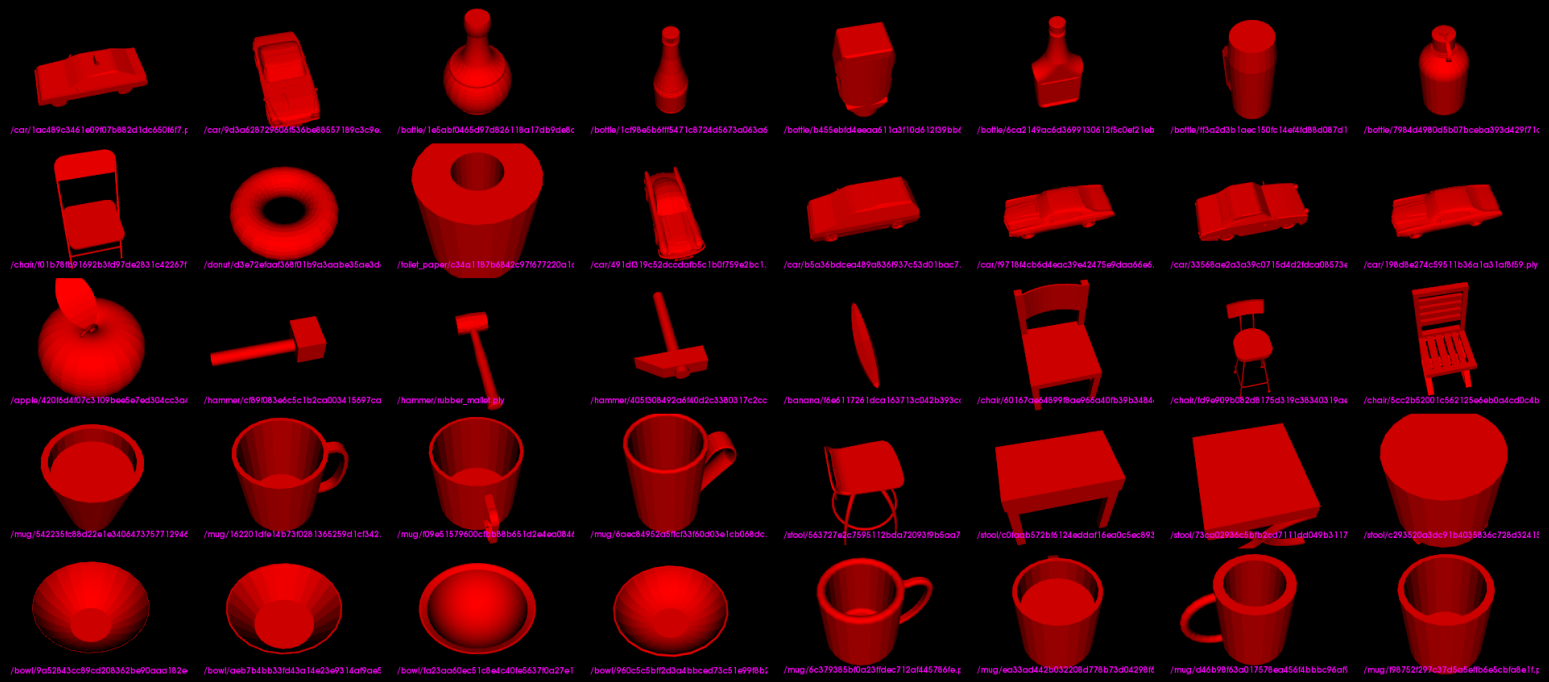
- Scanned object models



Training data

Sources:

- CAD models



Using CAD models

Why and How

- Training on 3D models is ...

Using CAD models

Why and How

- Training on 3D models is ...
  - easy: all you need is a CAD model and PCL

Using CAD models

Why and How

- Training on 3D models is ...
  - easy: all you need is a CAD model and PCL
  - fast: some seconds / model

Using CAD models

Why and How

- Training on 3D models is ...
  - easy: all you need is a CAD model and PCL
  - fast: some seconds / model
  - complete: no missed views, viewpoints evenly spaced around the object

Using CAD models

Why and How

- Training on 3D models is ...
  - easy: all you need is a CAD model and PCL
  - fast: some seconds / model
  - complete: no missed views, viewpoints evenly spaced around the object
  - parameterizable: easily re-trainable with different parameters
    - number of views
    - resolution
    - noise level
    - distance to object

Using CAD models

Why and How

- Training on 3D models is ...
  - easy: all you need is a CAD model and PCL
  - fast: some seconds / model
  - complete: no missed views, viewpoints evenly spaced around the object
  - parameterizable: easily re-trainable with different parameters
    - number of views
    - resolution
    - noise level
    - distance to object
  - sensor independent: adapt code to simulate your favorite sensor characteristics (fov, aliasing, noise, errors, holes)

Using CAD models

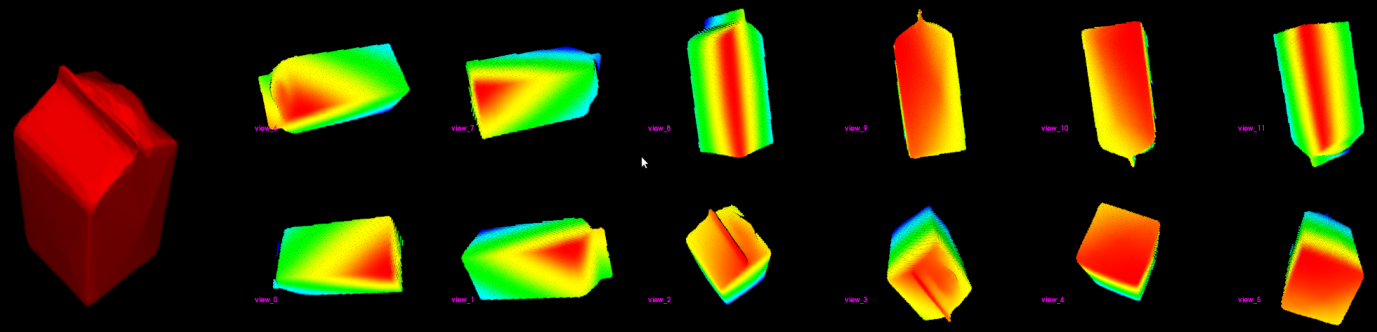
Why and How

- Training on 3D models is ...
  - easy: all you need is a CAD model and PCL
  - fast: some seconds / model
  - complete: no missed views, viewpoints evenly spaced around the object
  - parameterizable: easily re-trainable with different parameters
    - number of views
    - resolution
    - noise level
    - distance to object
  - sensor independent: adapt code to simulate your favorite sensor characteristics (fov, aliasing, noise, errors, holes)
  - providing additional information: entropy, aspect graphs, common reference frame

Using CAD models

Why and How

Obtain synthetic partial views of CAD models with PCL



```
std::string PLYModel = std::string(argv[1]);
float resx = atof(argv[2]);
float resy = resx;
typedef pcl::PointCloud<pcl::PointXYZ> Cloud;
std::vector <Cloud, ... > views_xyz;
std::vector < Eigen::Matrix4f, ... > poses;
std::vector<float> entropies;

pcl::visualization::PCLVisualizer vis;
vis.addModelFromPLYFile (PLYModel, "mesh1", 0);
vis.setRepresentationToSurfaceForAllActors ();
vis.renderViewTesselatedSphere (resx, resy, views_xyz,
                                poses, entropies);
```

Object Recognition

- What we have so far
  - Input from Kinect
  - Segmentation
  - Descriptors
  - Training data

Object Recognition

- What we have so far
  - Input from Kinect
  - Segmentation
  - Descriptors
  - Training data
  - Matching against DB: L1,L2, Chi-Square,... with FLANN
  - best matching objects+views

Object Recognition

Time for a live-demo...

- Recognition/Classification Demo

Object Recognition with PCL

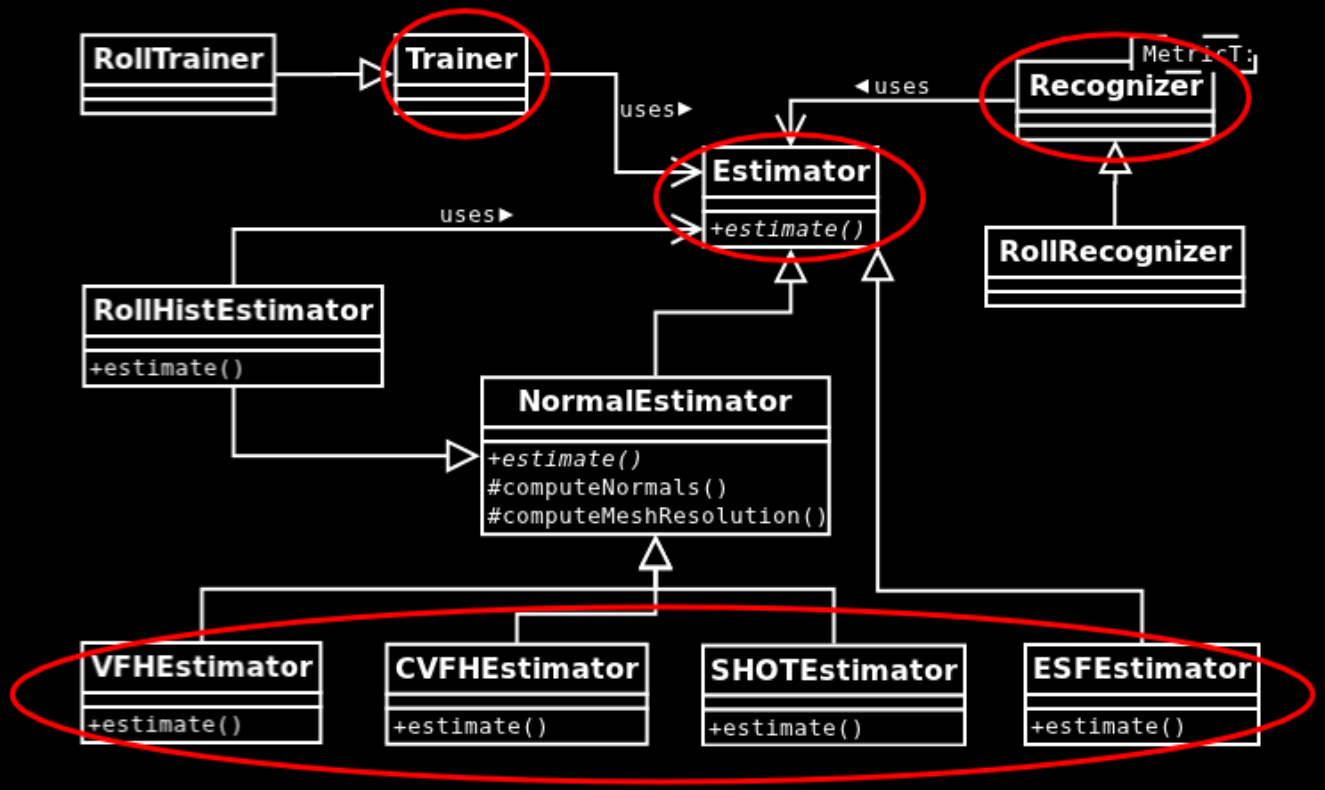
How much code is to write?

```
pcl::visualization::PCLVisualizer vis("kinect");
pcl::apps::DominantPlaneSegmentation<OpenNIFrameSource::PointT> dps;
while(camera.isActive()) {
  frame = camera.snap();
  dps.setInputCloud (frame);
  dps.setMaxZBounds (Z_DIST_);
  dps.setObjectMinHeight (0.005);
  dps.setMinClusterSize (1000);
  dps.setWSize(3);
  dps.setDistanceBetweenClusters(0.1);
  dps.setDownsamplingSize (0.02);
  dps.compute_fast (clusters);
  vis.addPointCloud<OpenNIFrameSource::PointT>(filtered,"frame");
  for(size_t i=0; i < clusters.size(); i++) {
    vis.addPointCloud<OpenNIFrameSource::PointT> (clusters[i], rando

    recognizer.recognize (*cluster[i], model_ids, poses, confidences

    std::string category = getCategory(model_ids[0]);
    Eigen::Vector4f centroid;
    pcl::compute3DCentroid(*cluster[i], centroid);
    vis.addText3D(category, centroid, text_scale, 1, 0, 1, cluster_n
  }
  vis.spinOnce();
```

Object Recognition

its even easier

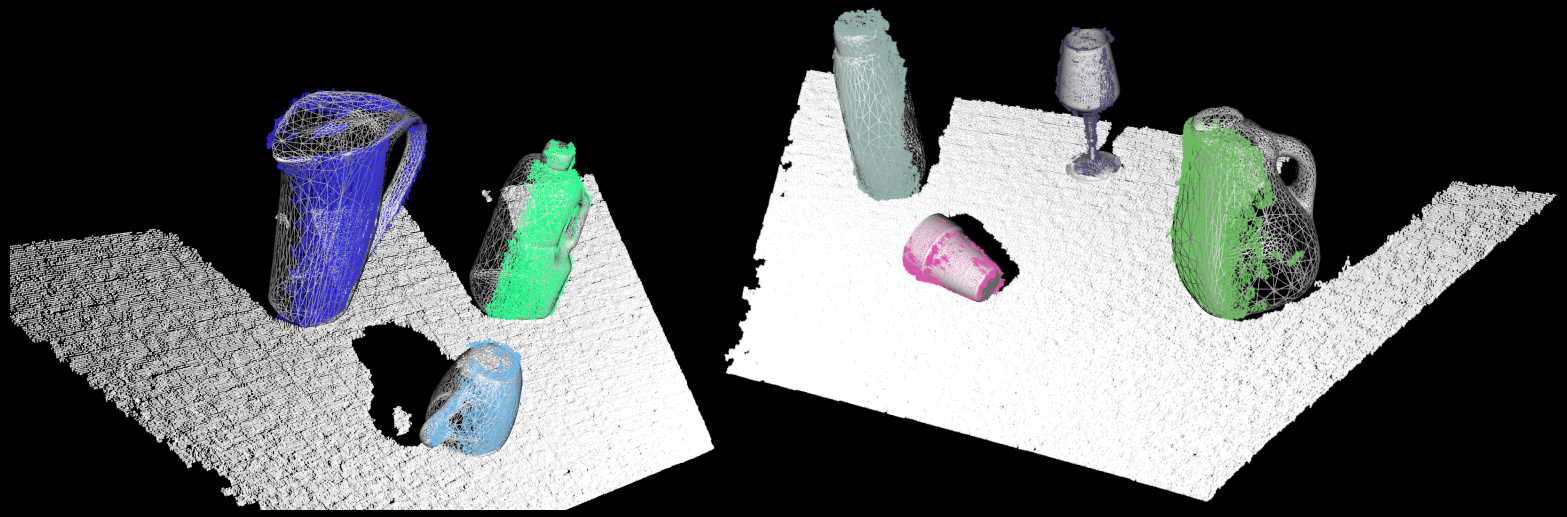
- some generic classes do most of the work



Object Recognition with PCL

Recognition + Pose

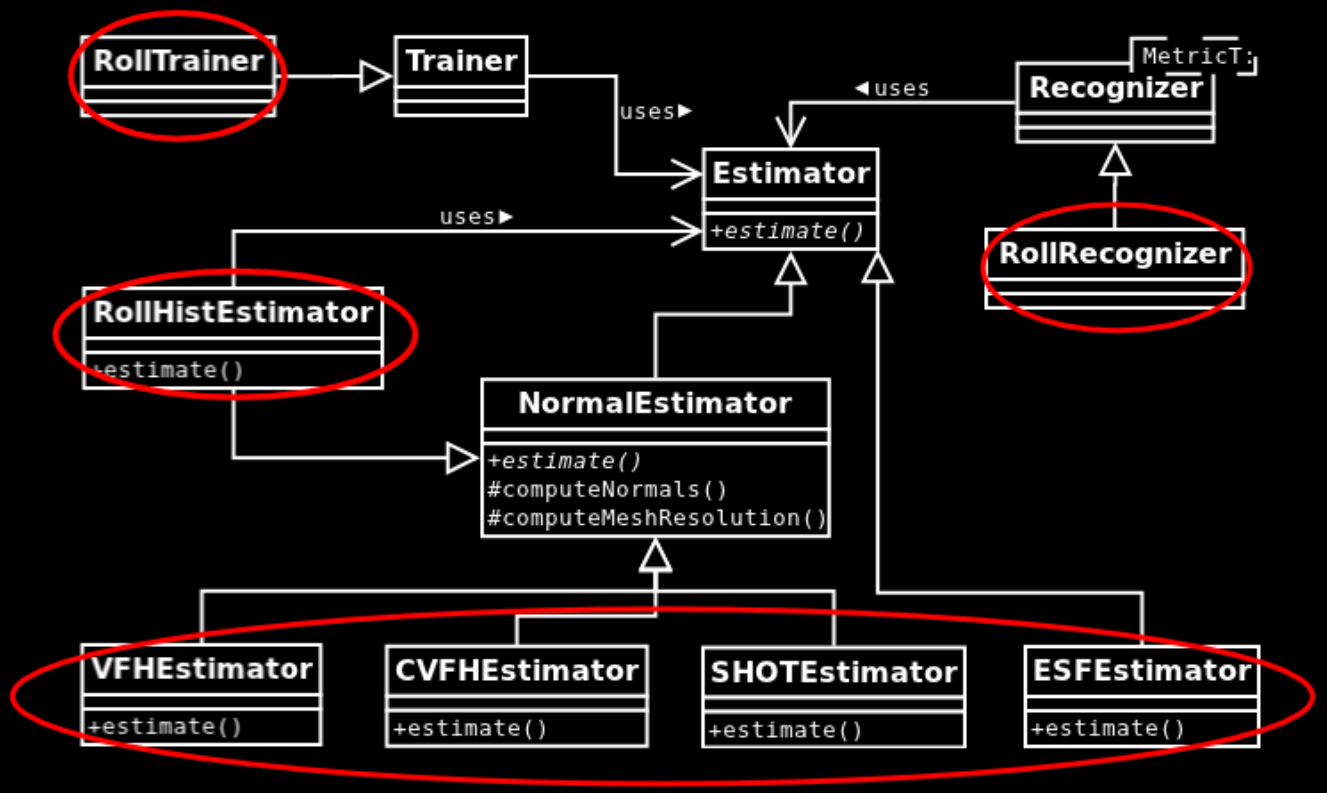
- Aligned CAD models with objects



Object Recognition with PCL

Recognition + Pose

- classes for



Object Recognition

Time for a live-demo...

- Recognition/Classification Demo with Pose

Databases

Where to get training data?

- Are there freely available sources out there?

Databases

Where to get data...

- Training data is freely available
  - 3D-Net (3d-net.org)
  - Google's 3D Warehouse, www.123dapp.com, turbosquid, ...
  - RoboEarth
  - Princeton Shape Benchmark
  - Willow Garage household objects DB
  - KIT household object database, TUM, ...
  - ...

Databases

Having lots of CAD models...

- Nearest neighbor classification video

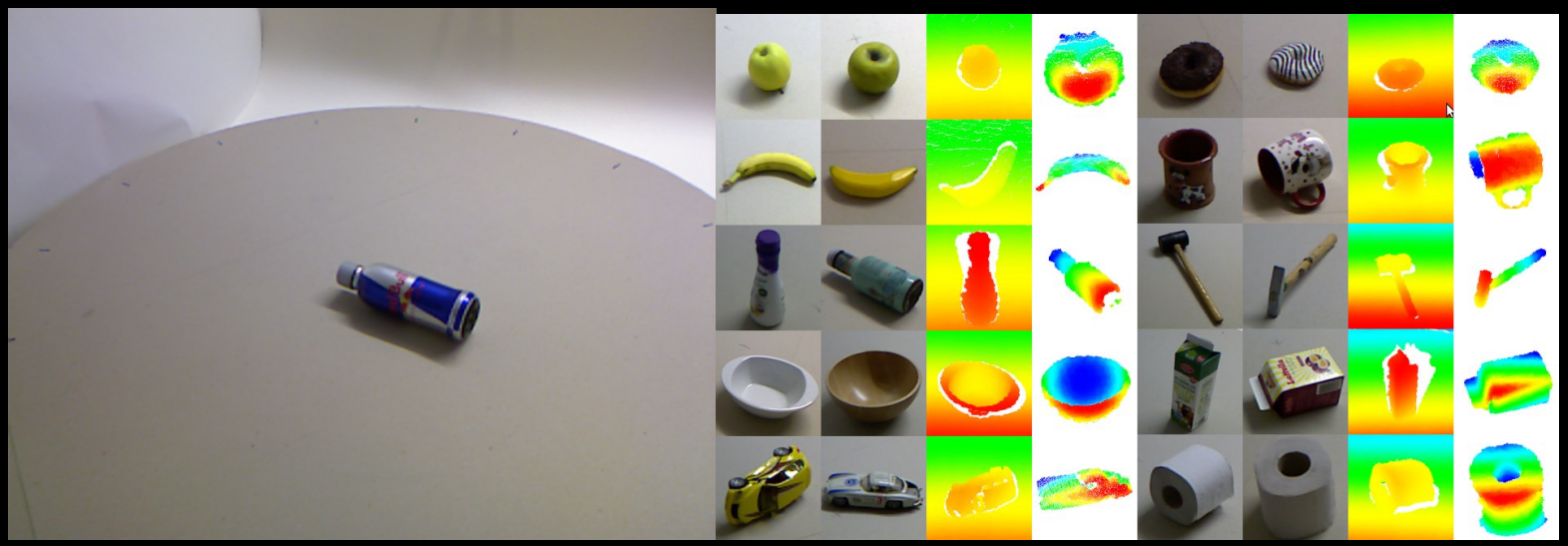
Evaluation

- Are there test databases out there?

Evaluation

Test databases

- from 3D-Net
  - 1600 Kinect scenes, 10 Classes, single object, multiple poses
  - +1600 scenes for 40 additional classes



Evaluation

Test databases

- from SHREC
  - http://www.aimatshape.net/event/SHREC/
  - Shape Retrieval Contest of Range Scans
  - Range images captured using a Minolta Laser Scanner
  - segmented objects, as mesh

Evaluation

Test databases

- from Lai's RGBD-DB
  - http://www.cs.washington.edu/rgbd-dataset/
  - 300 common household objects, 51 classes
  - on Turntable with Kinect, 3 diff. viewing angles
  - segmented objects, as colored point cloud

Evaluation

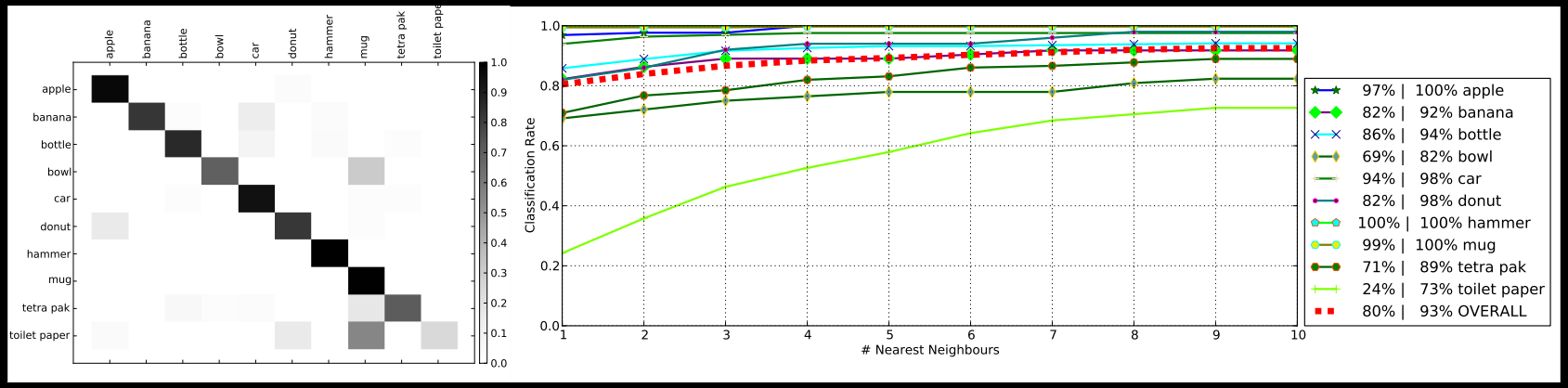
Evaluation code

- e.g. to test your new feature/descriptor
  - implement your feature/descriptor in PCL
  - get training models, put into directory
  - get a test database
  - use provided evaluation code

```
./category_evaluation
-models_dir ../ICCV/ModelDatabase
-training_dir ../ICCV/ModelDatabase_trained
-test_dir ../3DNET/Cat10_TestDatabase
-descriptor_name vfh
-n_nearest_neighbours 4
```

Evaluation

Easy evaluation



The end

- PCL demo-apps and code to simplify evaluation and testing
- helps you concentrate on your research area