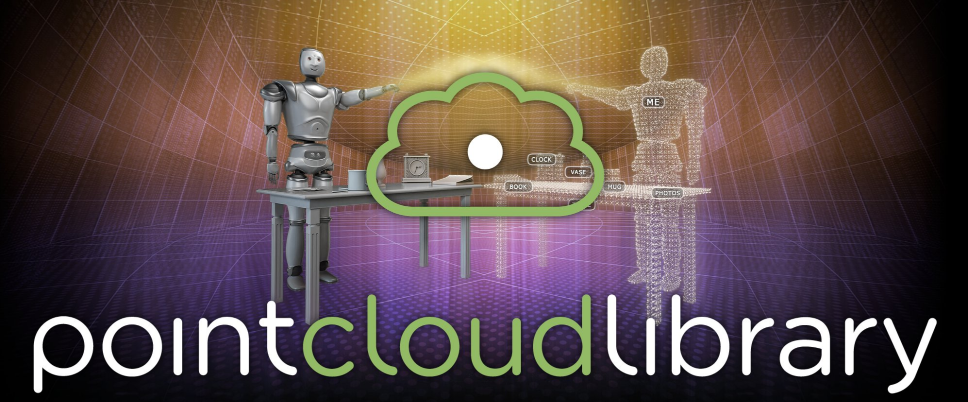


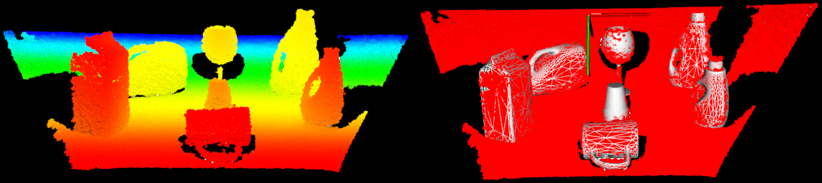
3D Object Recognition and 6DOF Pose Estimation

Aitor Aldoma, Federico Tombari

June 4, 2013

- Before coffee we saw "How does a good feature look like?"
  - Representing shapes (global, local) in a compact manner.
  - How to match features to define correspondences between source and target.

- Before coffee we saw "How does a good feature look like?"
  - Representing shapes (global, local) in a compact manner.
  - How to match features to define correspondences between source and target.
- In this talk:
  - Given a set of models (training data), how do we recognize them and estimate their pose in a particular scene?



pointcloudlibrary

Recognition pipeline

- pcl::keypoints, pcl::features covered previously
- In this talk, we focus on CorrespondenceGrouping and Hypothesis Verification.
- In contrast to registration, we simultaneously deal with several models.
- Other options in PCL:
  - LINEMOD [HinterstoisserPAMI2012]
  - ORR [PapazovACCV2010]
  - Segmentation + global features

1. Correspondence grouping
    - Given a set of correspondences (models - scene), group them together in geometrically consistent clusters from which the pose of the models can be extracted.
    - In contrast to RANSAC based methods, allows simultaneous recognition of multiple objects.
    - Usually applied on recognition pipelines based on local features.

1. Correspondence grouping
   - Given a set of correspondences (models - scene), group them together in geometrically consistent clusters from which the pose of the models can be extracted.
   - In contrast to RANSAC based methods, allows simultaneous recognition of multiple objects.
   - Usually applied on recognition pipelines based on local features.
2. Hypothesis Verification; given a set of object hypotheses with a 6DoF pose.
   - Aims at removing false positives while keeping true positives.
   - Allows merging hypotheses coming from different pipelines in a principled way.

1. Correspondence grouping
   - Given a set of correspondences (models - scene), group them together in geometrically consistent clusters from which the pose of the models can be extracted.
   - In contrast to RANSAC based methods, allows simultaneous recognition of multiple objects.
   - Usually applied on recognition pipelines based on local features.
2. Hypothesis Verification; given a set of object hypotheses with a 6DoF pose.
   - Aims at removing false positives while keeping true positives.
   - Allows merging hypotheses coming from different pipelines in a principled way.

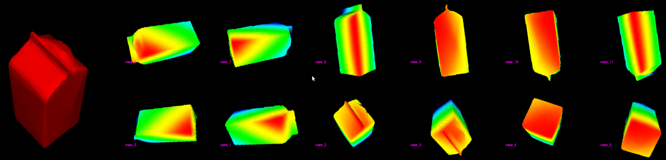
PCL module: pcl::recognition

pointcloudlibrary

Before we start...

From 3D models to 2.5D data

- Simulate input from depth/3D sensors.



```
typedef pcl::PointCloud<pcl::PointXYZ>::Ptr CloudPtr;
pcl::apps::RenderViewsTesselatedSphere render_views;
render_views.setResolution (resolution_);
render_views.setTesselationLevel (1); //80 views
render_views.addModelFromPolyData (mapper); //vtk model
render_views.setGenOrganized(false);
render_views.generateViews ();
std::vector< CloudPtr > views;
std::vector < Eigen::Matrix4f > poses;
render_views.getViews (views);
render_views.getPoses (poses);
```

pointcloudlibrary

Outline

1. Introduction
2. Correspondence Grouping
3. Hypothesis Verification

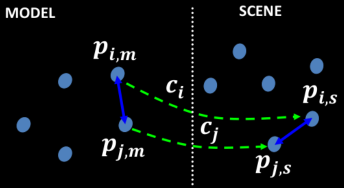
pointcloudlibrary

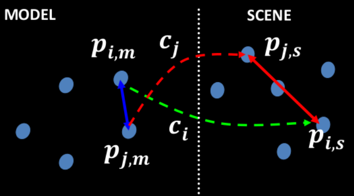
Geometric consistency

- Incrementally build clusters of correspondences that are geometrically consistent:

$$\Big| ||p_{i,m} - p_{j,m}||_2 - ||p_{i,s} - p_{j,s}||_2 \Big| < \varepsilon \qquad (1)$$

- All elements in cluster are geom. consistent to each other.
- Parameters:
  - ε : keypoint inaccuracy, noise
  - *gc_min_size* : minimum cluster size (at least 3)

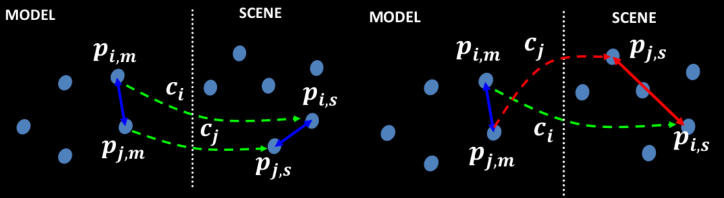




pointcloudlibrary

Geometric consistency (II)

- GC constraint is weak, especially for small consensus sizes!
- 6D space projected to 1D!
- Should be especially used when data is noisy or presents artifacts that do not permit to compute a repeatable RF (see Hough3D).



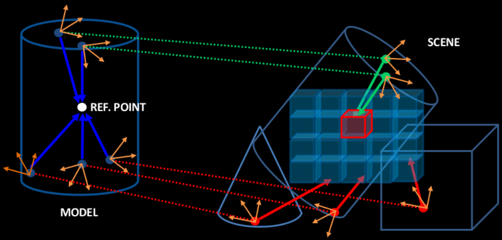
Geometric consistency (III)

- How to use it within PCL?
- m_s_corrs are correspondences with indices to m_keypoints and s_keypoints.

```
pcl::CorrespondencesPtr m_s_corrs; //fill it
std::vector<pcl::Correspondences> clusters;
pcl::GeometricConsistencyGrouping<PT, PT> gc_clusterer;
gc_clusterer.setGCSize (cg_size);
gc_clusterer.setGCThreshold (cg_thres);
gc_clusterer.setInputCloud (m_keypoints);
gc_clusterer.setSceneCloud (s_keypoints);
gc_clusterer.setModelSceneCorrespondences (m_s_corrs);
gc_clusterer.cluster (clusters);
```

Hough 3D voting

- Correspondence votes are accumulated in a 3D Hough space. [TombariIPSJ2012]
- Each point associated with a repeatable RF, RFs used to:
  - reduce voting space from 6 to 3D...
  - ... by reorienting the voting location
- Local maxima in the Hough space identify object instances (handles the presence of multiple instances of the same model in the scene)



pointcloudlibrary

Hough 3D voting (II)

- How to use it within PCL?
- m_s_corrs are correspondences with indices to m_keypoints and s_keypoints.

```
typedef pcl::ReferenceFrame RFType;
pcl::PointCloud<RFType>::Ptr model_rf; //fill with RFs
pcl::PointCloud<RFType>::Ptr scene_rf; //fill with RFs
pcl::CorrespondencesPtr m_s_corrs; //fill it
std::vector<pcl::Correspondences> clusters;

pcl::Hough3DGrouping<PT, PT, RFType, RFType> hc;
hc.setHoughBinSize (cg_size);
hc.setHoughThreshold (cg_thres);
hc.setUseInterpolation (true);
hc.setUseDistanceWeight (false);
hc.setInputCloud (m_keypoints);
hc.setInputRf (model_rf);
hc.setSceneCloud (s_keypoints);
hc.setSceneRf (scene_rf);
hc.setModelSceneCorrespondences (m_s_corrs);
hc.cluster (clusters);
```

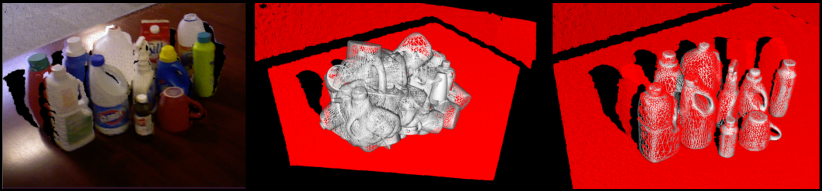
pointcloudlibrary

Outline

1. Introduction
2. Correspondence Grouping
3. Hypothesis Verification

Hypothesis Verification

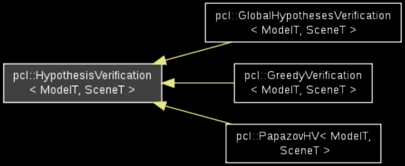
- Keep along the recognition pipeline as many hypotheses as possible and use HV to select those best "explaining the scene".
- A hypothesis $\mathcal{M}_i$ is a model aligned to the scene $\mathcal{S}$.
- Main goal: Remove FPs without rejecting TPs.



pointcloudlibrary

Hypothesis Verification

- Keep along the recognition pipeline as many hypotheses as possible and use HV to select those best "explaining the scene".
- A hypothesis $\mathcal{M}_i$ is a model aligned to the scene $\mathcal{S}$.
- Main goal: Remove FPs without rejecting TPs.
- 3 options in PCL:
  - Greedy [AldomaDAGM12]
  - Conflict Graph [PapazovACCV11]
  - Global HV [AldomaECCV12]



pointcloudlibrary

HV: Greedy

- Reasoning about occlusions to handle occluded objects (in common with all 3 methods).
- For each hypothesis $\mathcal{M}_i$, count $\#$*inliers* and $\#$*outliers*.
- Greedily select the best hypothesis ($\#$*inliers* $- \lambda \cdot \#$*outliers*) ...
- ... and update the inliers count for successive hypotheses, resort and repeat.
- $\mathcal{M}_i$ selected if $\#$*inliers* $- \lambda \cdot \#$*outliers* > 0.

```
pcl::GreedyVerification<pcl::PointXYZ, pcl::PointXYZ> greedy_hv(lambda
greedy_hv.setResolution (0.005f);
greedy_hv.setInlierThreshold (0.005f);
greedy_hv.setSceneCloud (scene);
greedy_hv.addModels (aligned_hypotheses, true);
greedy_hv.verify ();
std::vector<bool> mask_hv;
greedy_hv.getMask (mask_hv);
```

pointcloudlibrary

HV: Conflict Graph

- First, a sequential stage that discards hypotheses based on percentage of inliers and outliers.
- From the remaining hypotheses, some are selected based on a non-maxima suppression stage on a conflict graph.
- Two hypothesis are in conflict if they share the same space.

```
pcl::PapazovHV<pcl::PointXYZ, pcl::PointXYZ> papazov;
papazov.setResolution (0.005f);
papazov.setInlierThreshold (0.005f);
papazov.setSupportThreshold (0.08f); //inliers
papazov.setPenaltyThreshold (0.05f); //outliers
papazov.setConflictThreshold (0.02f);
papazov.setSceneCloud (scene);
papazov.addModels (aligned_hypotheses, true);
papazov.verify ();
std::vector<bool> mask_hv;
papazov.getMask (mask_hv);
```

HV: Global HV

- Consider the two possible states of a single hypothesis $x_i = \{0, 1\}$ (inactive/active).
- By switching the state of an hypothesis, we can evaluate a global cost function that tell us how good the current solution $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ is.
- Formally, we are looking for a solution $\tilde{\mathcal{X}}$ such that:

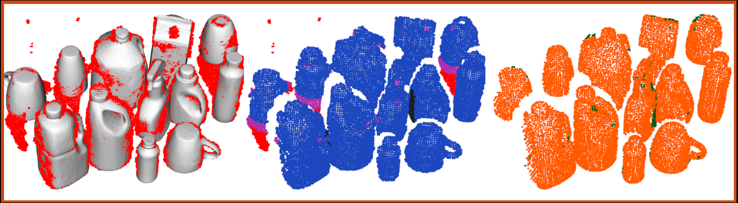
$$\tilde{\mathcal{X}} = \underset{\mathcal{X} \in \mathbb{B}^n}{\text{argmin}} \left\{ \mathfrak{F}(\mathcal{X}) = f_{\mathcal{S}}(\mathcal{X}) + \lambda \cdot f_{\mathcal{M}}(\mathcal{X}) \right\} \quad (2)$$

- $\mathfrak{F}(\mathcal{X})$ considers the whole set of hypotheses ($\mathcal{M}$) as a global scene model instead of considering each model hypothesis separately.

pointcloudlibrary

HV: Global HV (II)

- $\mathfrak{F}(\mathcal{X})$ simultaneously enforces cues defined on the scene $\mathcal{S}$ as well as cues defined on the set of hypothesis, $\mathcal{M}$.



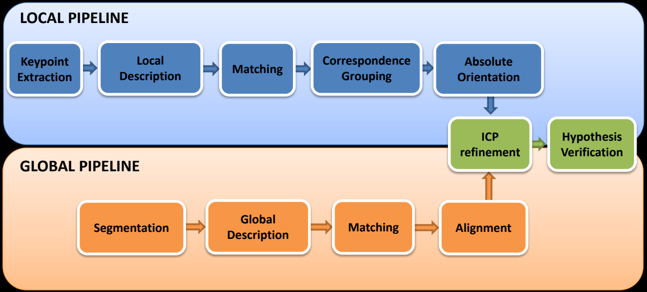
- Given a certain configuration of $\mathcal{X} = \{x_1, ..., x_n\}$:
  - Maximize number of scene points explained (orange).
  - Minimize number of model outliers (green).
  - Minimize number of scene points multiple explained (black).
  - Minimize number of unexplained scene points close to active hypotheses (yellow, purple)
- Optimization solved using Simulated Annealing or other metaheuristics (METSlib library).

pointcloudlibrary

HV: Global HV (III)

```
pcl::GlobalHypothesesVerification<pcl::PointXYZ, pcl::PointXYZ> go;
go.setResolution (0.005f);
go.setInlierThreshold (0.005f);
go.setRadiusClutter (0.04f);
go.setRegularizer (3.f); //outliers' model weight
go.setClutterRegularizer (5.f); //clutter points weight
go.setDetectClutter (true);
go.setSceneCloud (scene);
go.addModels (aligned_hypotheses, true);
go.verify ();
std::vector<bool> mask_hv;
go.getMask (mask_hv);
```

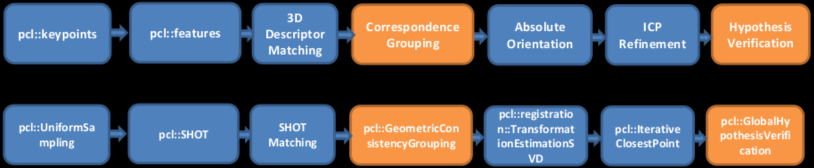
Global and local descriptors



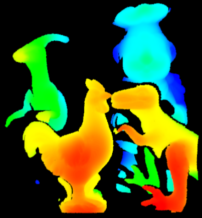
Note: global descriptor matching often does not yield automatically the object pose!

pointcloudlibrary

Example



- 5 object models (Mian dataset), pipeline as in [Aldoma ECCV12]



/home/aitor/data/Mians_dataset/scenes/pcl_scenes//rs7.ply.pcd

Recognition results

Ground truth