3D Object Recognition and 6DOF Pose Estimation

Aitor Aldoma

May 10, 2013
This morning, 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.
This morning, 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?
- **pcl::keypoints, pcl::features** covered previously in the morning.
- 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`
Before we start...

From 3D models to 2.5D data

▶ Simulate input from depth/3D sensors.

```cpp
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);
```
1. Introduction

2. Correspondence Grouping

3. Hypothesis Verification

4. A recognition example
Incrementally build clusters of correspondences that are geometrically consistent:

\[ \left| \left| p_{i,m} - p_{j,m} \right|_2 - \left| p_{i,s} - p_{j,s} \right|_2 \right| < \varepsilon \]  

All elements in cluster are geometrically consistent to each other.

Parameters:
- \( \varepsilon \): keypoint inaccuracy, noise
- \( gc\_min\_size \): minimum cluster size (at least 3)
GC constraint is weak, specially for small consensus sizes!

6D space projected to 1D!

Should be specially 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`.

```cpp
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);
```
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)
How to use it within PCL?

- m_s_corrs are correspondences with indices to m_keypoints and s_keypoints.

```cpp
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);
```

Aitor Aldoma, Federico Tombari / PCL :: Recognition
1. Introduction
2. Correspondence Grouping
3. Hypothesis Verification
4. A recognition example
Keep along the recognition pipeline as many hypotheses as possible and use HV to select those best "explaining the scene".

A hypothesis $M_i$ is a model aligned to the scene $S$.

Main goal: Remove FPs without rejecting TPs.
Keep along the recognition pipeline as many hypotheses as possible and use HV to select those best "explaining the scene".

A hypothesis $M_i$ is a model aligned to the scene $S$.

Main goal: Remove FPs without rejecting TPs.

3 options in PCL:
- Greedy [AldomaDAGM12]
- Conflict Graph [PapazovACCV11]
- Global HV [AldomaECCV12]
Reasoning about occlusions to handle occluded objects (in common with all 3 methods).

For each hypothesis $M_i$, count $\#\text{inliers}$ and $\#\text{outliers}$.

Greedily select the best hypothesis 
$(\#\text{inliers} - \lambda \cdot \#\text{outliers})$ ...

... and update the inliers count for successive hypotheses, resort and repeat.

$M_i$ selected if $\#\text{inliers} - \lambda \cdot \#\text{outliers} > 0$.

```cpp
def 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);
```
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 supression stage on a conflict graph.

Two hypothesis are in conflict if they share the same space.

```cpp
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);
```
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:

\[
\tilde{\mathcal{X}} = \arg\min_{\mathcal{X} \in \mathcal{B}^n} \{ \mathcal{F}(\mathcal{X}) = f_S(\mathcal{X}) + \lambda \cdot f_M(\mathcal{X}) \}
\]  

\( \mathcal{F}(\mathcal{X}) \) considers the whole set of hypotheses \( \mathcal{M} \) as a global scene model instead of considering each model hypothesis separately.
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, \ldots, 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 (METSlab library).
HV: Global HV (III)

```cpp
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);
```
Outline

1. Introduction
2. Correspondence Grouping
3. Hypothesis Verification
4. A recognition example
5 object models.