PCL :: Registration
Initial Alignment of 3D Point Clouds
Dirk Holz
November 6, 2011
Registering 3D Point Clouds for building 3D models of objects, table scenes, and whole rooms/buildings/outdoor environments.

1. Initial alignment (registration without previous knowledge)
2. Iterative refinement (given an initial alignment)
3. Multi-view registration
Registering 3D Point Clouds

- Given a source point cloud and a target point cloud
  1. determine correspondence pairs,
  2. estimate a transformation that aligns the correspondences,
  3. apply the transformation to align source and target.

- correspondences can be anything (points/features/...)!
Registration Pipeline in PCL

1. Data acquisition
2. Keypoints estimation
3. Feature descriptors estimation
4. Correspondences estimation (matching)
5. Correspondence rejection method 1
6. Correspondence rejection method 2
7. Transformation estimation
8. Correspondence rejection method N
1. Compute sets of keypoints
2. Compute (local) feature descriptors (e.g. FPFH)
3. Match features to find correspondences
4. Estimate transformation from correspondences
**Code:** Matching Features

```cpp
CorrespondenceEstimation<FeatureT, FeatureT> est;
est.setInputCloud (source_features);
est.setInputTarget (target_features);
est.determineCorrespondences (correspondences);
```

**Example:** Found correspondences / matches

```
$ correspondence_viewer robot0 robot1 -n 50
```
Transformation Estimation

▶ Several methods for computing a transformation $T = (R, t)$ given correspondence pairs $(d_i, m_i)$:
  ▶ Point-to-point
  ▶ Point-to-plane
  ▶ Plane-to-plane
  ▶ ... and many others

▶ Simple solution (based on SVD) for minimizing point-to-point distance (least squares error $E$):

$$E(T) = \sum_i (m_i - (Rd_i + t))^2$$

`pcl::registration::TransformationEstimationSVD`
**Code**: Transformation estimation

```
Eigen::Matrix4f transformation;
TransformationEstimationSVD<PointT, PointT> svd;
svd.estimateRigidTransformation(source, target, correspondences, transformation);
```

**Example**: Simple initial alignment

**Problem**: False correspondences!
Rejecting false correspondences (outliers) using SAC

- Draw three correspondences pairs \( d_i, m_i \)
- Estimate transformation \((R, t)\) for these samples
- Determine inlier pairs with \( ((Rd_i + t) - m_i)^2 < \epsilon \)
- Repeat \( N \) times, and use \((R, t)\) having most inliers

**Code:** SAC-based correspondence rejection

```cpp
CorrespondenceRejectorSampleConsensus<PointT> sac;
sac.setInputCloud(source);
sac.setTargetCloud(target);
sac.setInlierThreshold(epsilon);
sac.setMaxIterations(N);
sac.setInputCorrespondences(correspondences);
sac.getCorrespondences(inliers);
Eigen::Matrix4f transformation = sac.
    getBestTransformation();
```
**Example:** SAC-based correspondence rejection

**Initial correspondences:**

**Inliers:**
Problem: In case of less descriptive features, the best match $m_i$ may not be the true correspondence for $d_i$!

**SAC-IA:** Sampled Consensus-Initial Alignment

1. Draw $n$ points $d_i$ from the source cloud (with a minimum distance $d$ in between).
2. For each drawn $d_i$:
   2.1 get $k$ closest matches, and
   2.2 draw one of the $k$ closest matches as $m_i$ (instead of taking closest match)
3. Estimate transformation $(R, t)$ for these samples
4. Determine inlier pairs with $((Rd_i + t) - m_i)^2 < \epsilon$
5. Repeat $N$ times, and use $(R, t)$ having most inliers
Code: Sampled Consensus-Initial Alignment

```cpp
pcl::SampleConsensusInitialAlignment
  <PointT, PointT, FeatureT> sac_ia;
  sac_ia.setNumberOfSamples (n);
  sac_ia.setMinSampleDistance (d);
  sac_ia.setCorrespondenceRandomness (k);
  sac_ia.setMaximumIterations (N);
  sac_ia.setInputCloud (source);
  sac_ia.setInputTarget (target);
  sac_ia.setSourceFeatures (source_features);
  sac_ia.setTargetFeatures (target_features);
  sac_ia.align (aligned_source);
  Eigen::Matrix4f = sac_ia.getFinalTransformation ();
```
Example: Sampled Consensus-Initial Alignment

$ test_registration robot0 robot1 -i 0.025,0.01,500
**Sampled Consesus-Initial Alignment + refinement**

```bash
$ test_registration robot0 robot1 \
-i 0.025,0.01,500 -r 0.05,0.05,0,100
```
Multi-cloud Registration
That’s it!

Thank you for your attention!

Any questions?