pointcloudlibrary

KARLSRUHE ICRA 2013

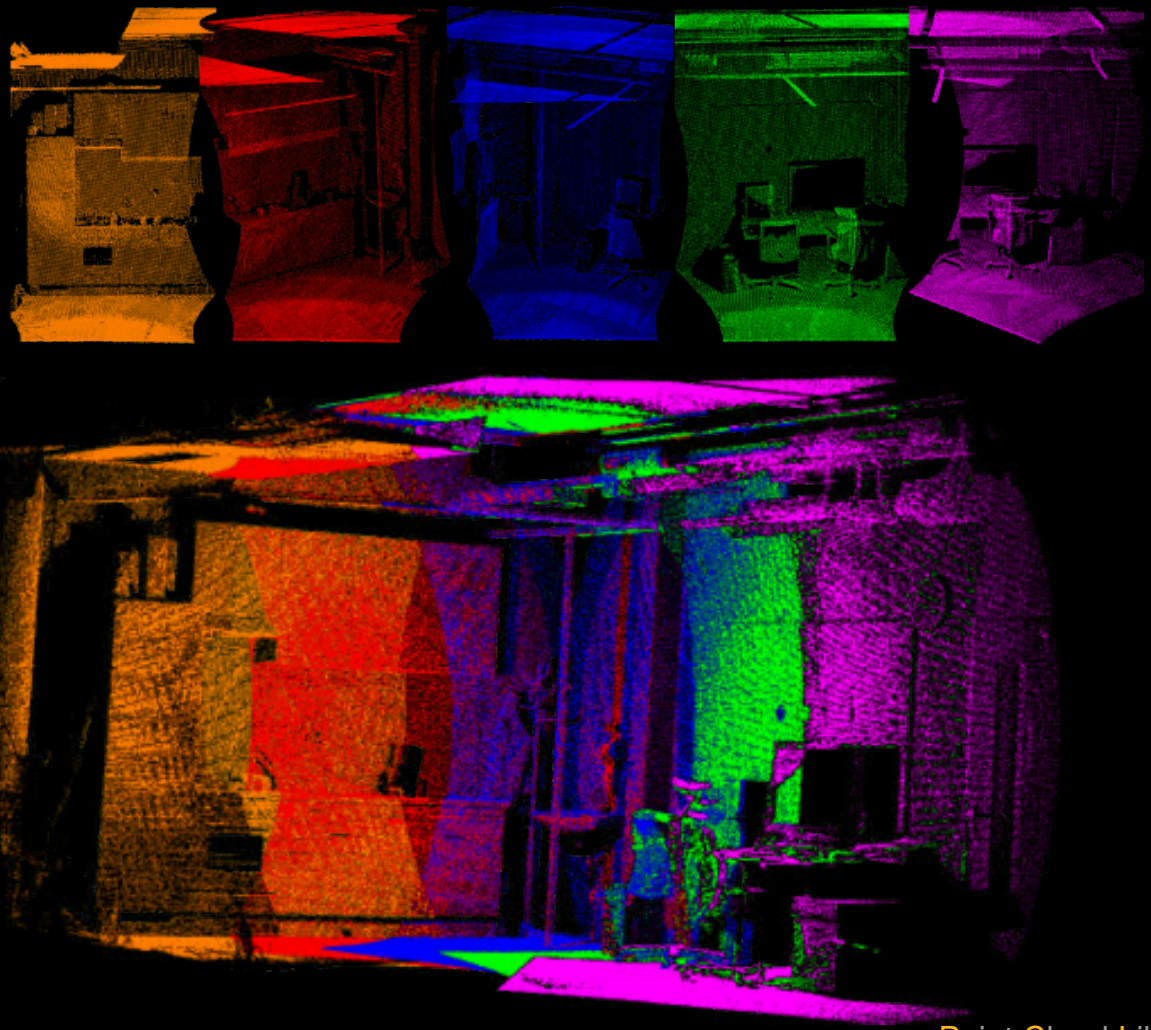
PCL :: Registration

Jochen Sprickerhof

May 18, 2012

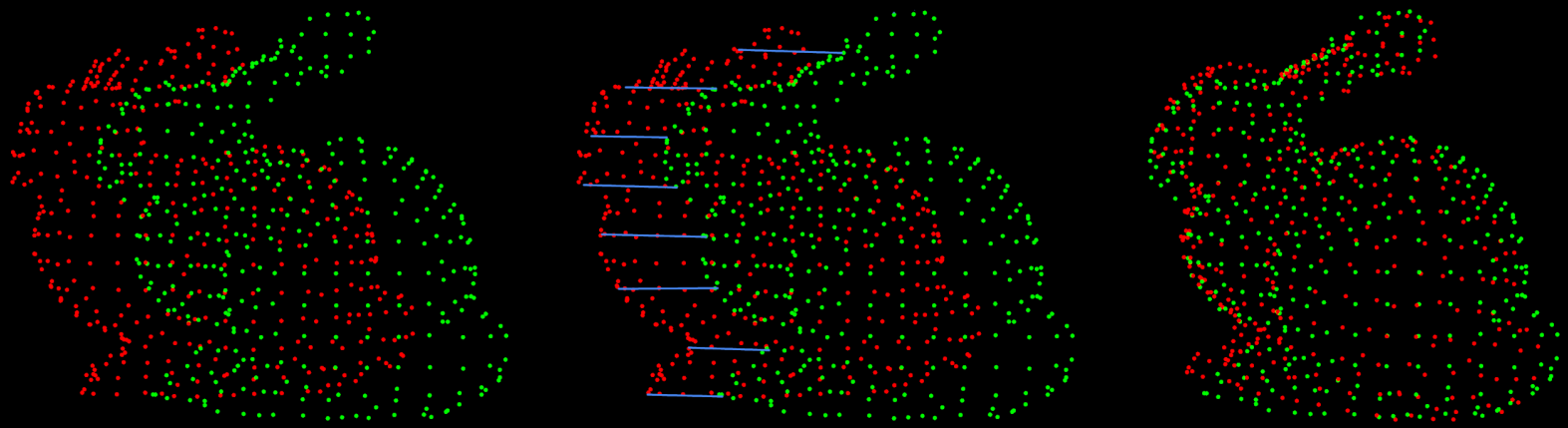
Registration

Registering 3D Point Clouds for building 3D models of objects, table scenes, and whole rooms/buildings/outdoor environments.



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.

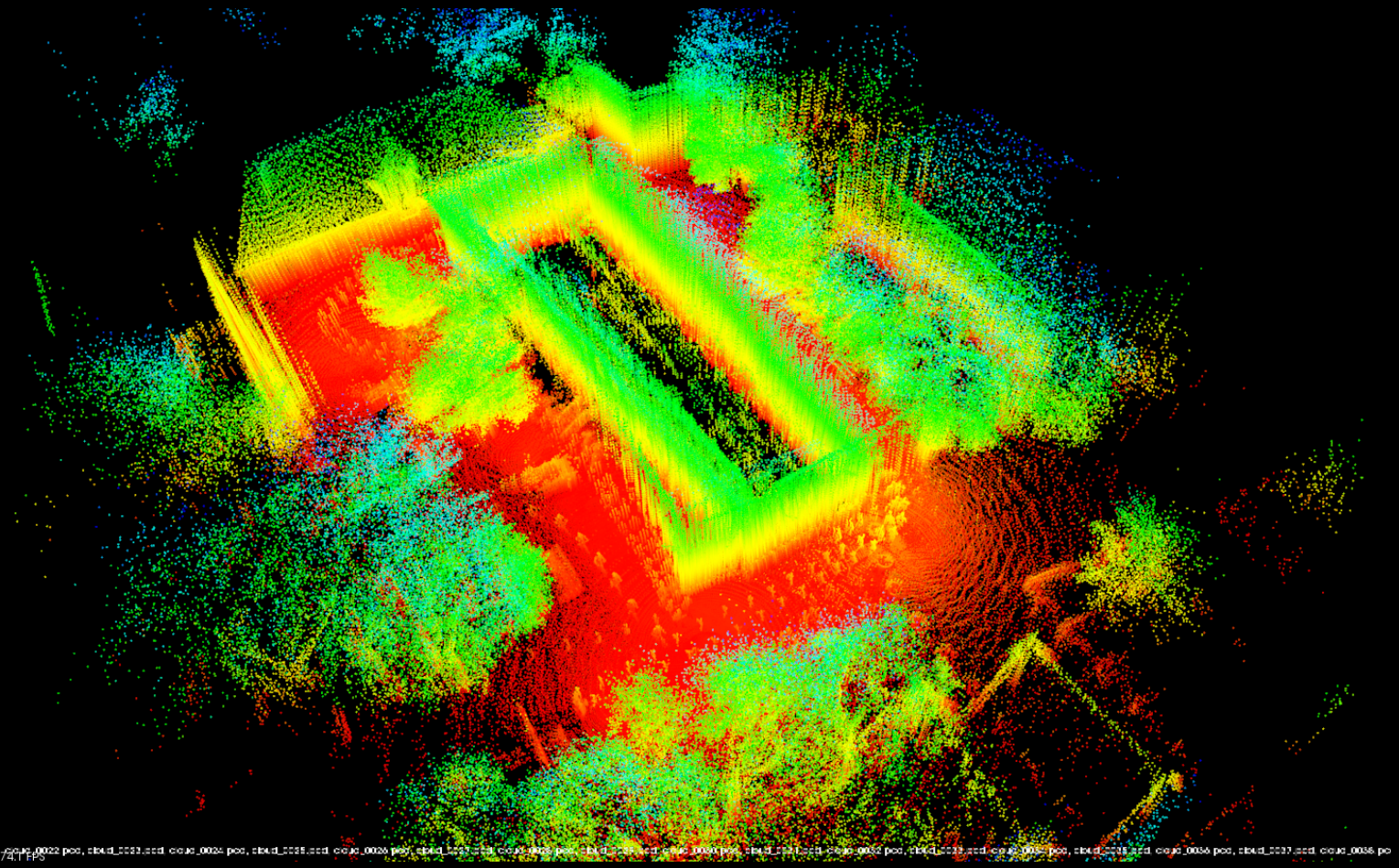
ICP Example

Code: ICP

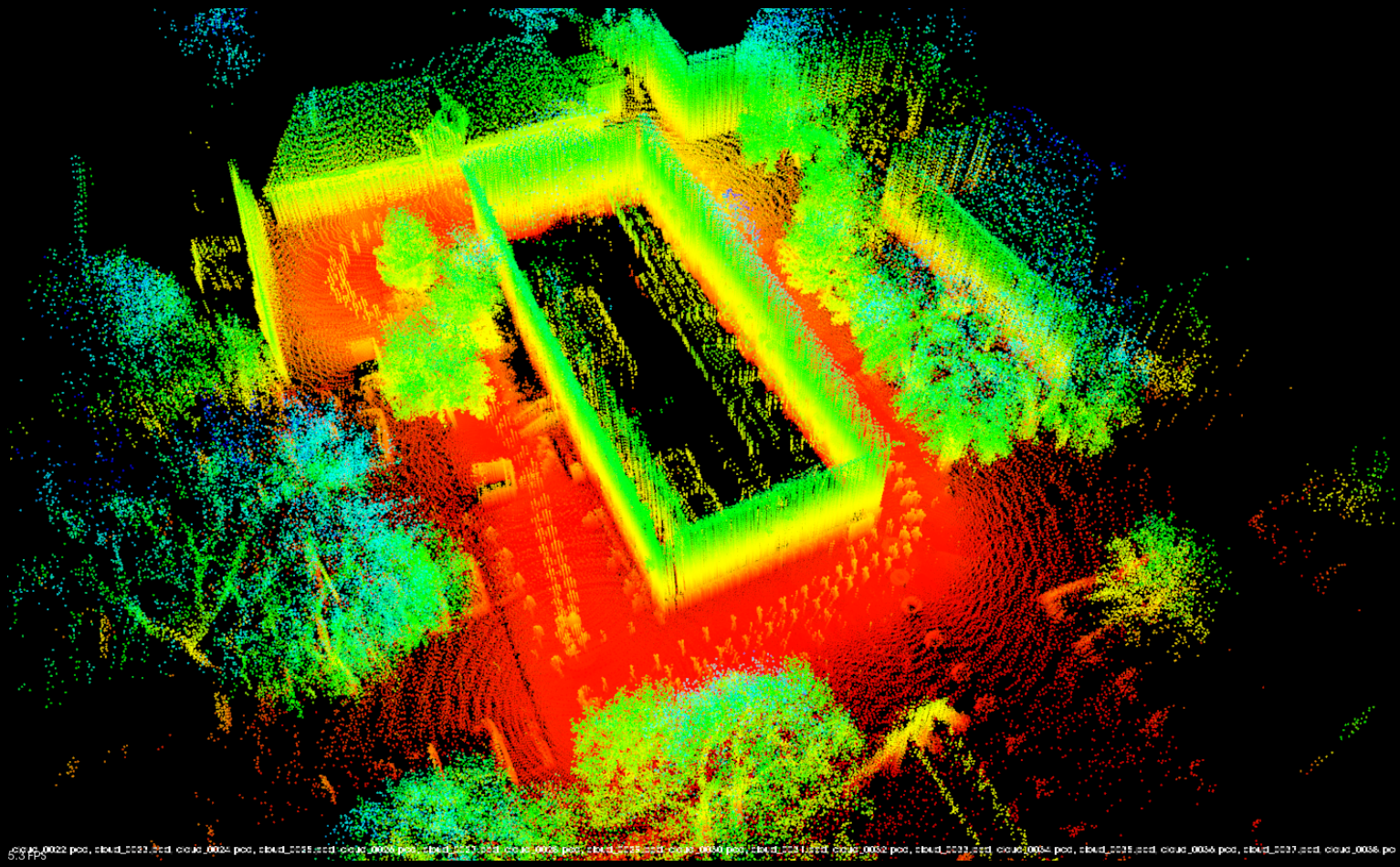
```
pcl::IterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ> icp;
icp.setInputCloud(cloud_in);
icp.setInputTarget(cloud_out);
pcl::PointCloud<pcl::PointXYZ> Final;
icp.align(Final);
```

The ICP API: Termination Criteria

- Max. number of iteration steps
  → set via `setMaximumIterations(nr_iterations)`
- Convergence: Estimated transformation doesn't change (the sum of differences between current and last transformation is smaller than a user-defined threshold)
  → set via `setTransformationEpsilon(epsilon)`
- A solution was found (the sum of squared errors is smaller than a user-defined threshold)
  → set via `setEuclideanFitnessEpsilon(distance)`



Input Data



```
pcl_icp -d 0.75 -r 0.75 -i 50 ../data/cloud_0{022..130}.pcd
```

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):

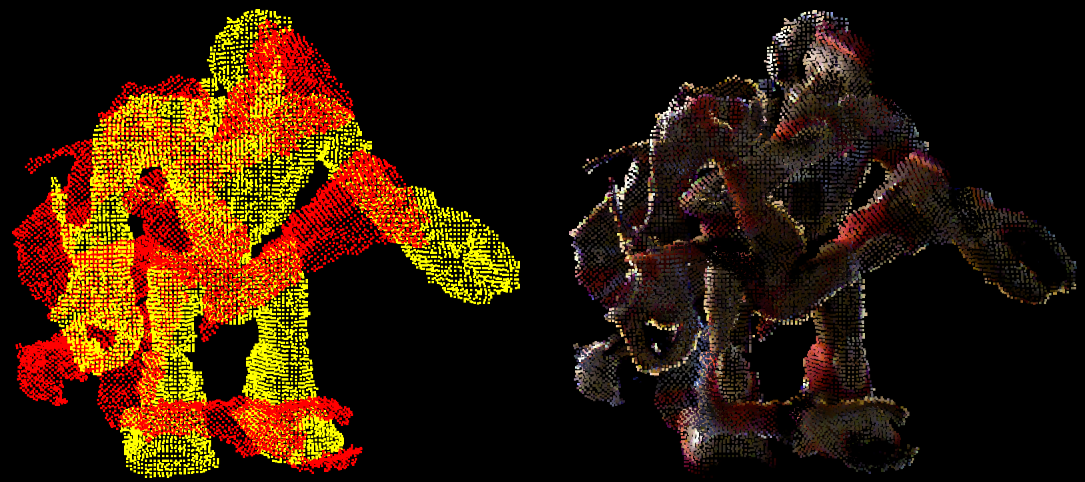
$$E(T) = \sum_i (m_i - (Rd_i + t))^2$$

TransformationEstimationSVD

Code: Transformation estimation

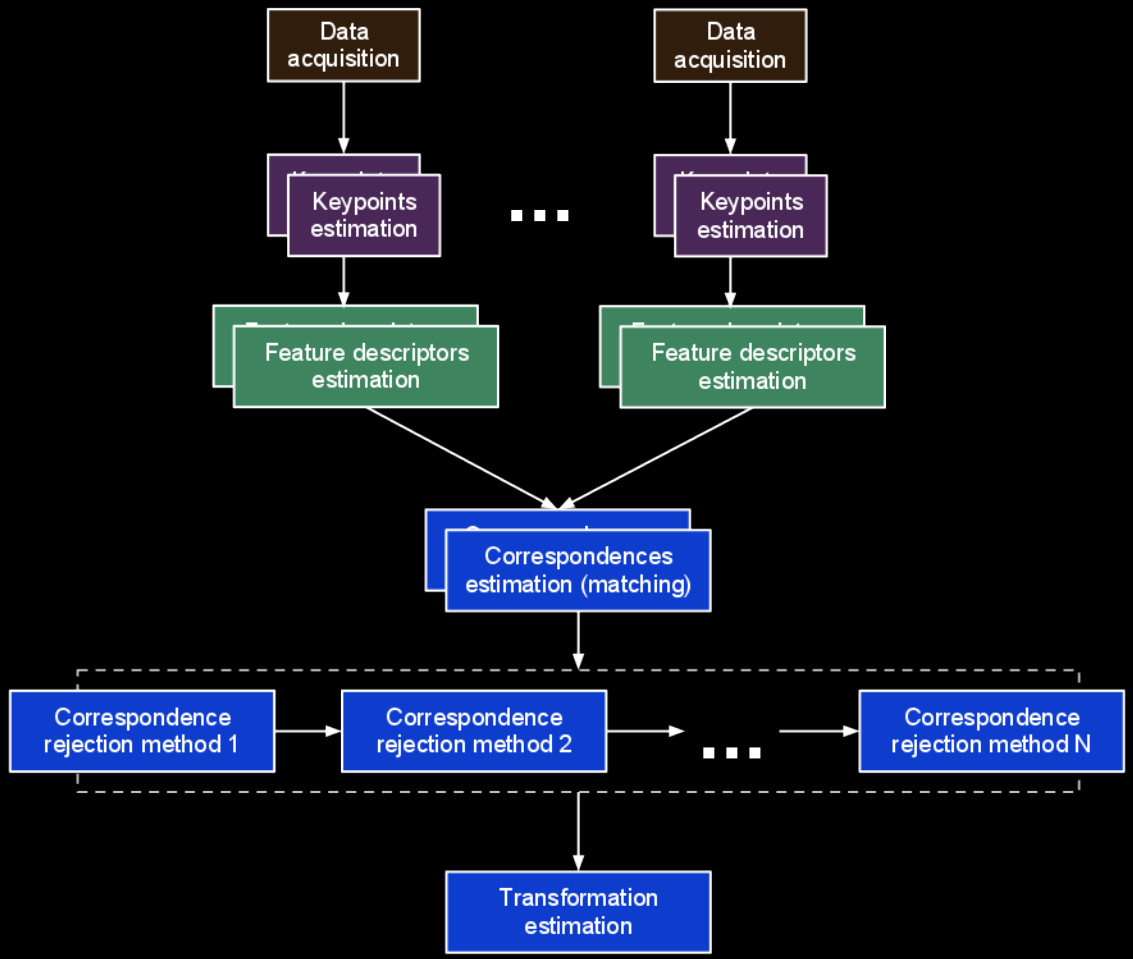
```
Eigen::Matrix4f transformation;
TransformationEstimationSVD<PointT, PointT> svd;
svd.estimateRigidTransformation(src, trgt, corres, trans);
```

Example: Simple initial alignment



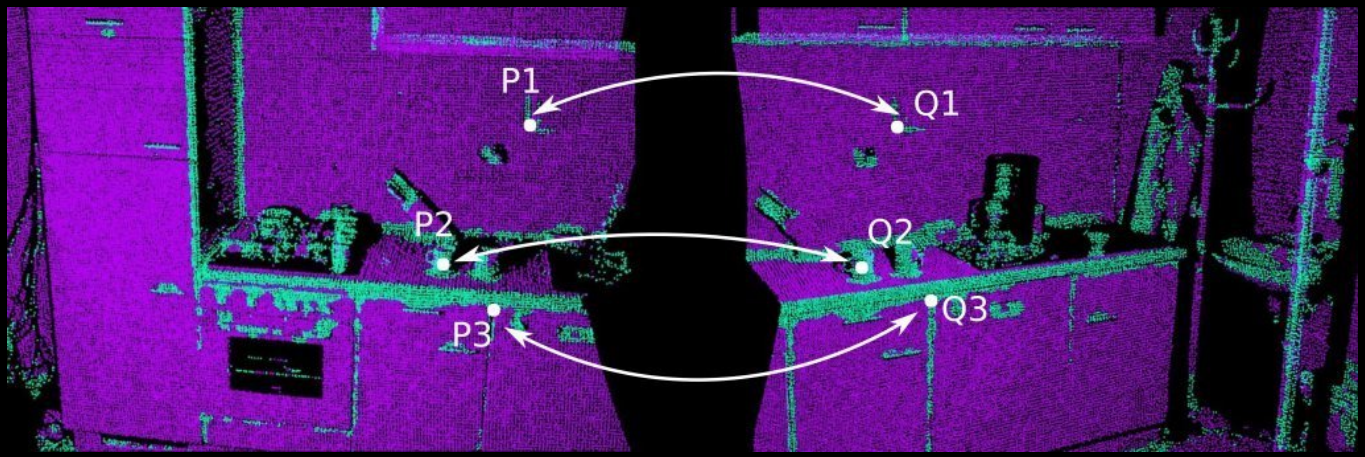
Problem: False correspondences!

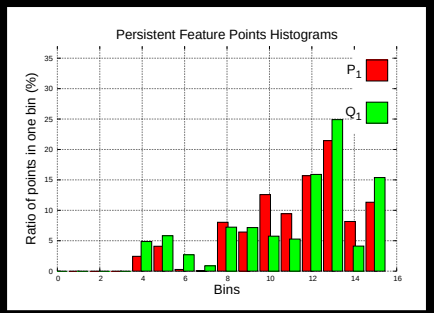
Registration

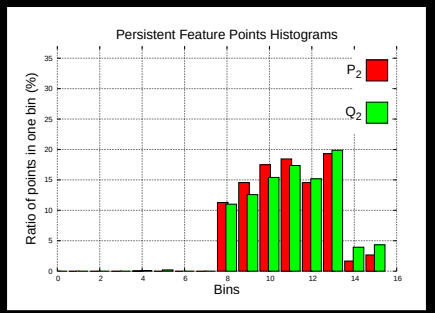


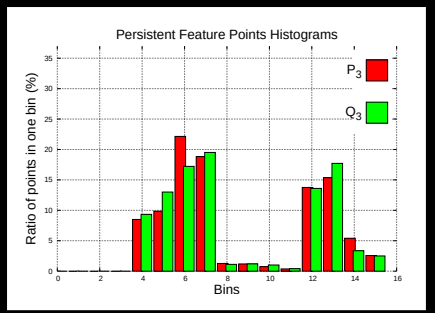
Features

1. Compute sets of keypoints
2. Compute (local) feature descriptors
3. Match features to find correspondences
4. Estimate transformation from correspondences







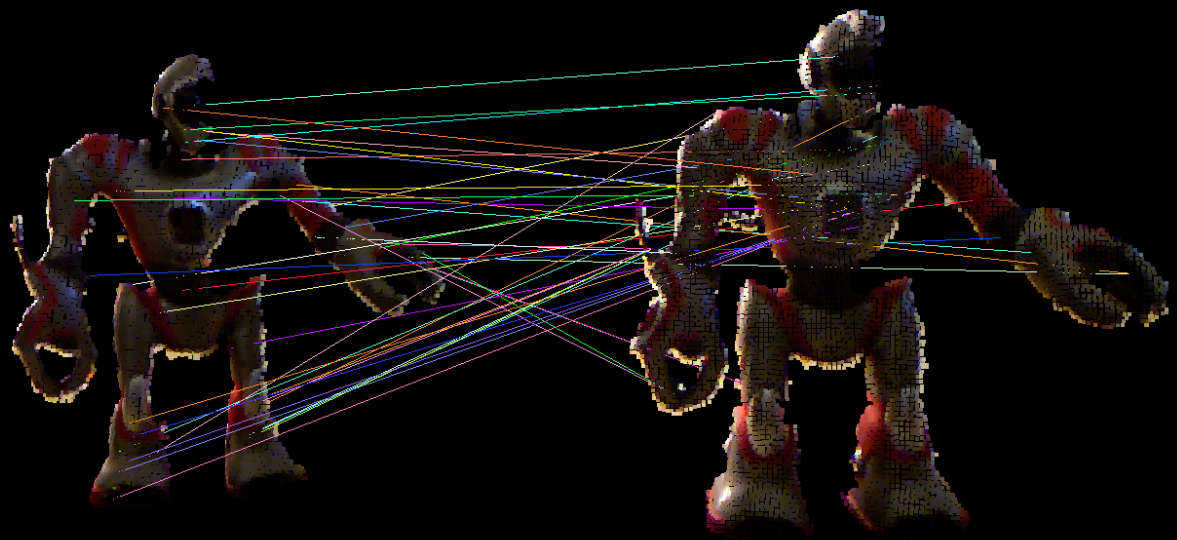


Feature Matching

Code: Matching Features

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

Example: Found correspondences / matches



Outlier Rejection

Rejecting false correspondences (outliers) uing 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

```
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();
```

Outlier Example

Example: SAC-based correspondence rejection

Initial correspondences:

Inliers:

Initial Alignment

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

SAC-IA: Sampled Consesus-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

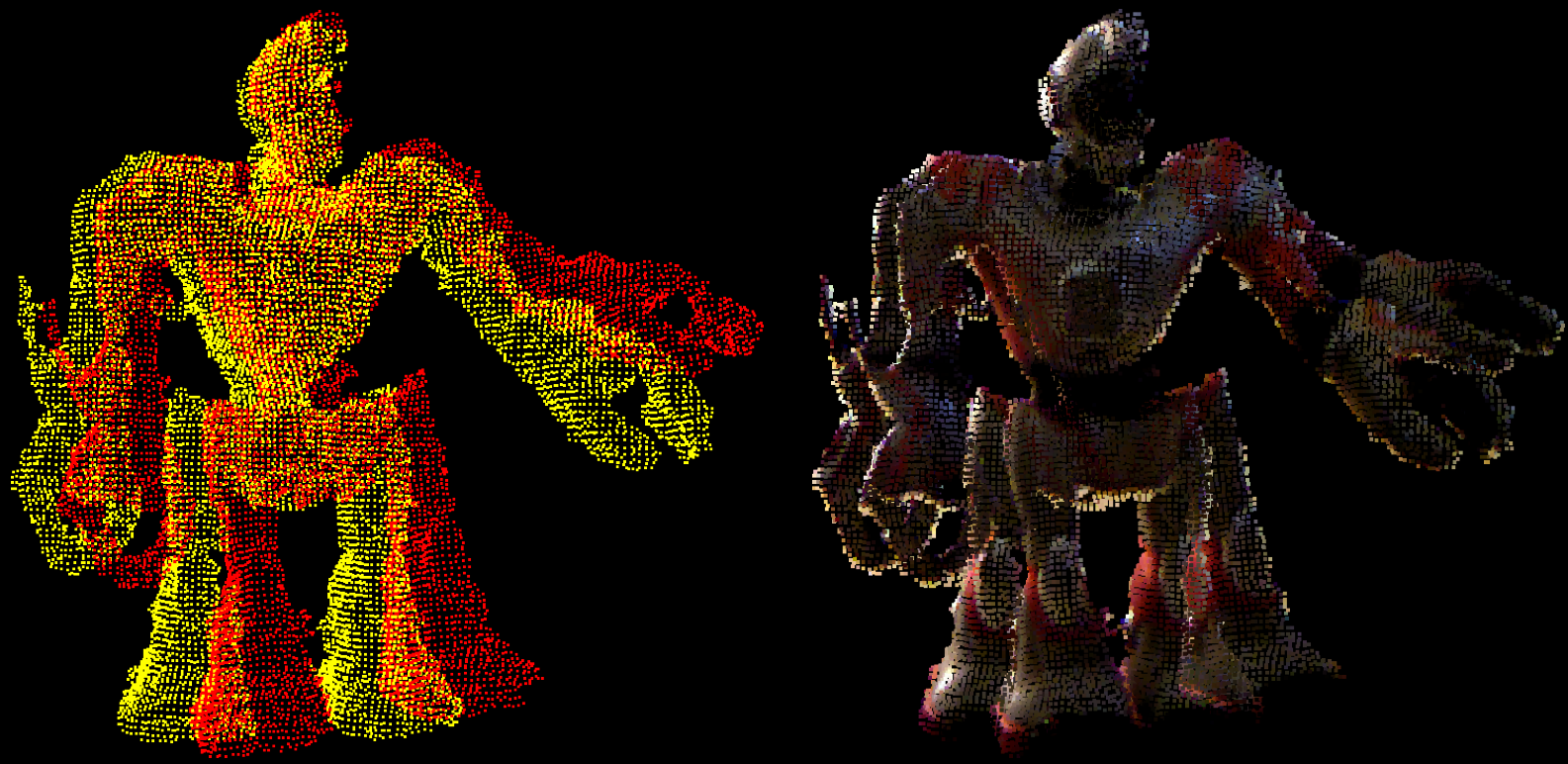
Initial Alignment Code

Code: Sampled Consesus-Initial Alignment

```
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 ();
```

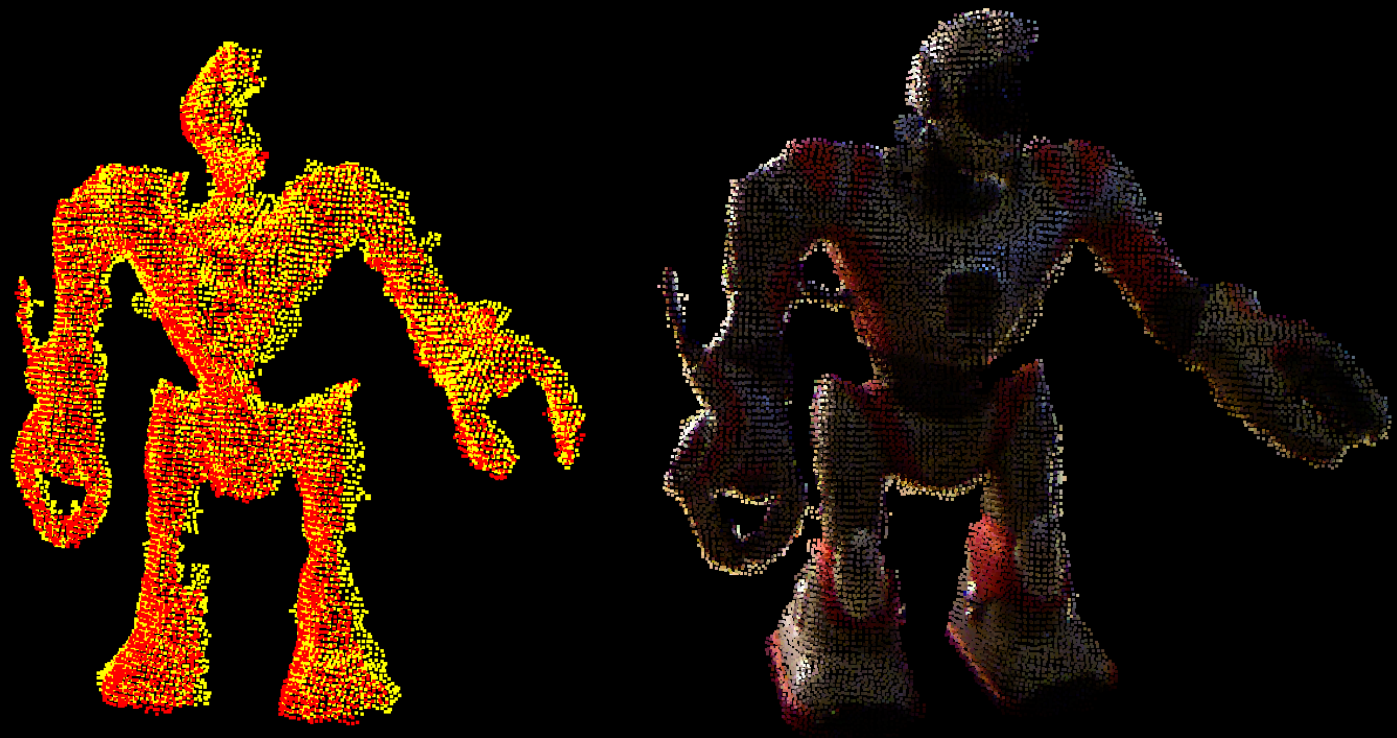
Initial Alignment Example

Example: Sampled Consesus-Initial Alignment



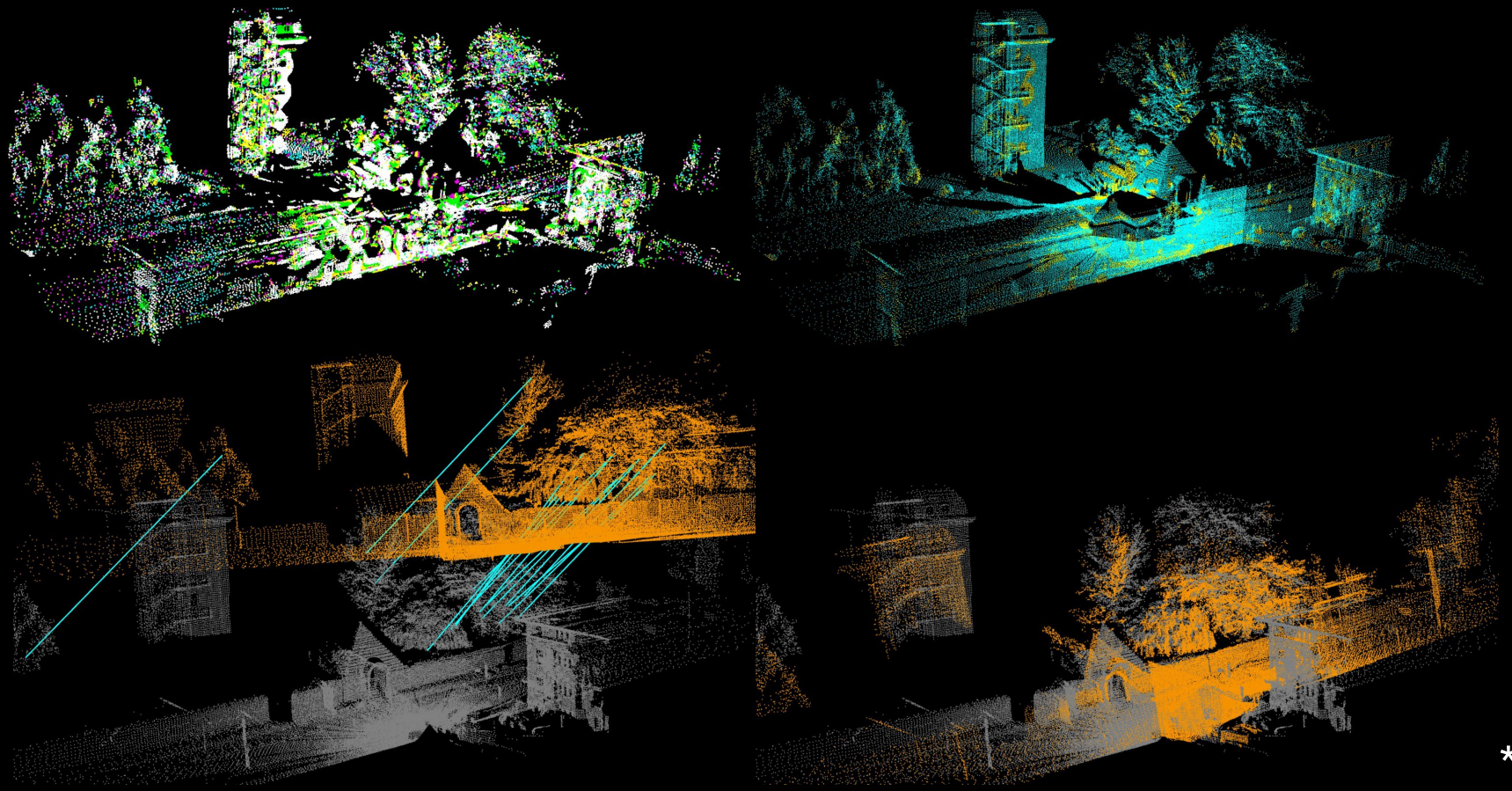
Complete Pipeline

Sampled Consesus-Initial Alignment + refinement



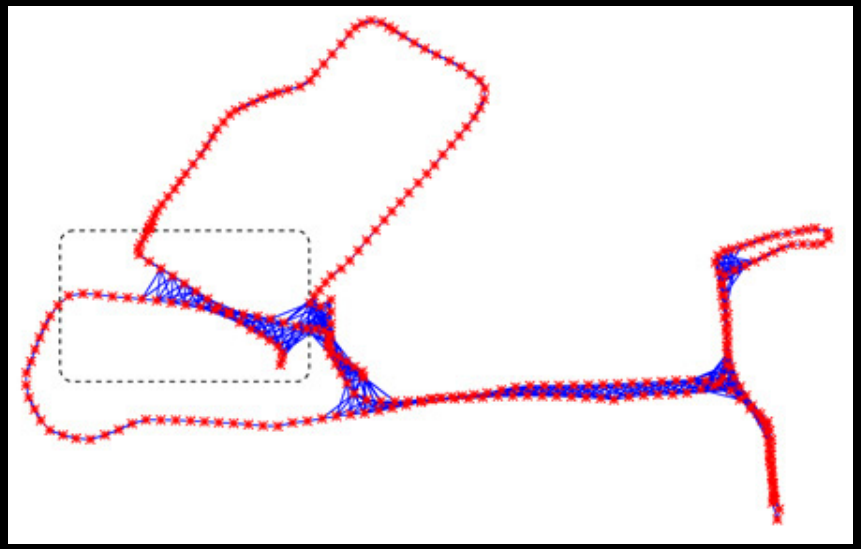
pointcloudlibrary

Examples



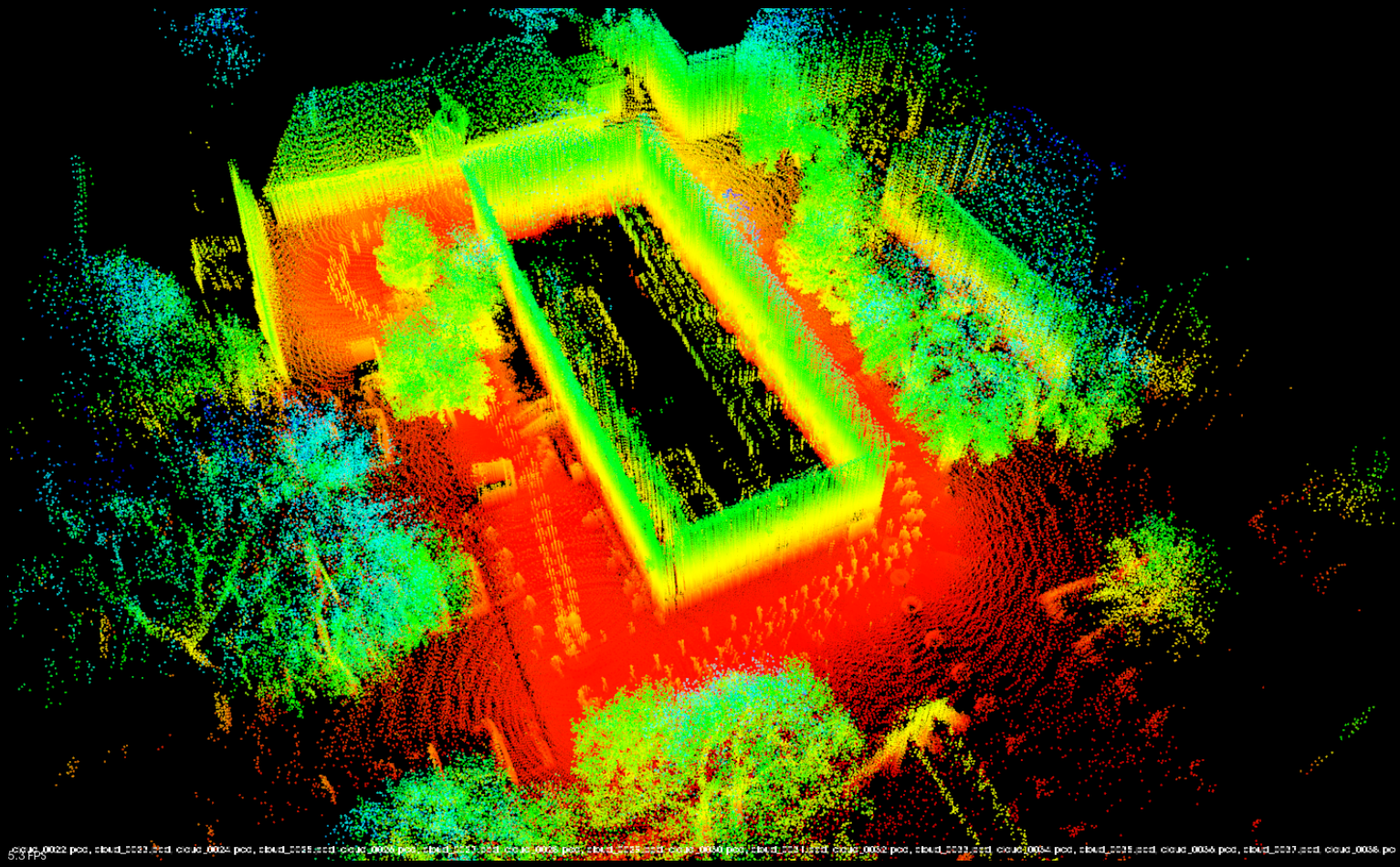
*

Global Relaxation

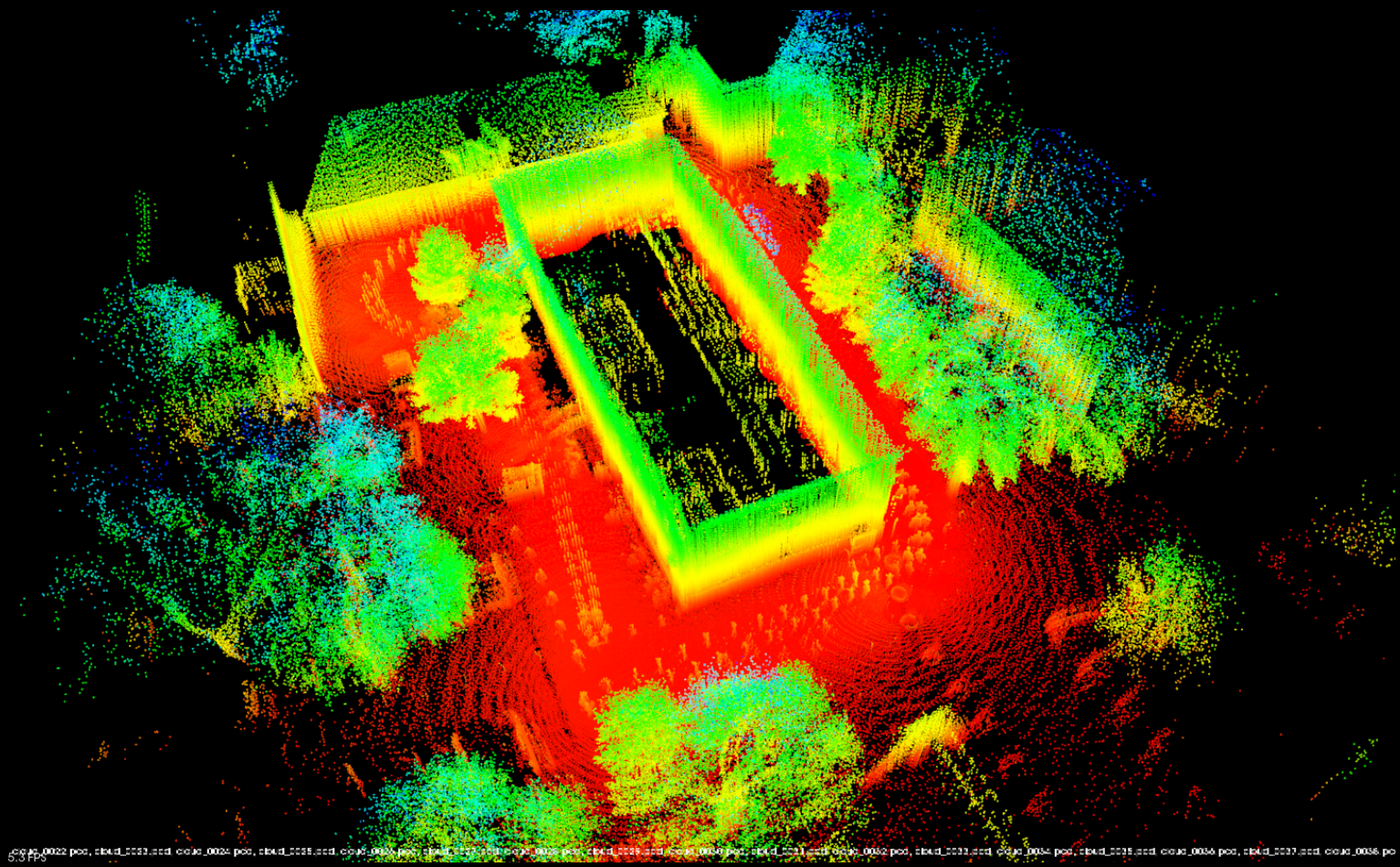


[Borrmann 2007]

```
pcl::registration::LUM<PointType> lum;
lum.setMaxIterations (lumIter);
lum.setConvergenceThreshold (0.001f);
for (int i = 1; i < n; i++)
{
  lum.addPointCloud (cloud[i]);
}
//for all point pairs (i, j)
lum.setCorrespondences (i, j, correspondences);
lum.compute ();
```

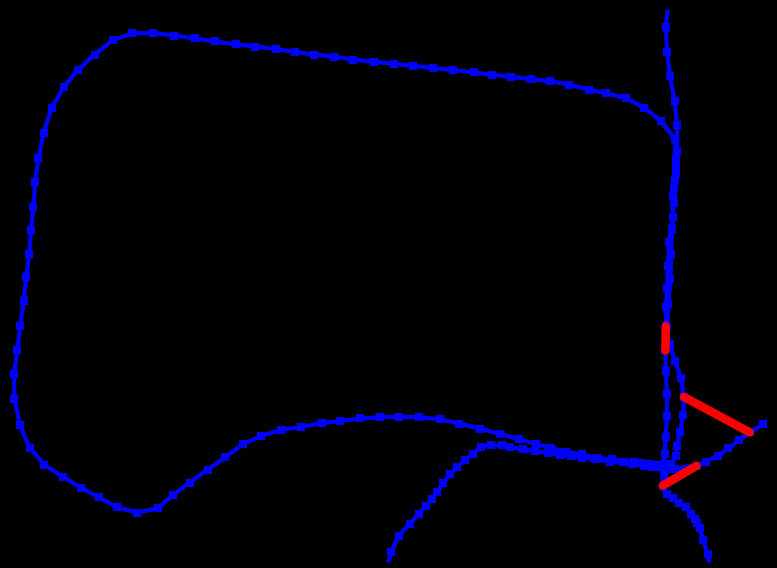


```
pcl_icp -d 0.75 -r 0.75 -i 50 ../data/cloud_0{022..130}.pcd
```

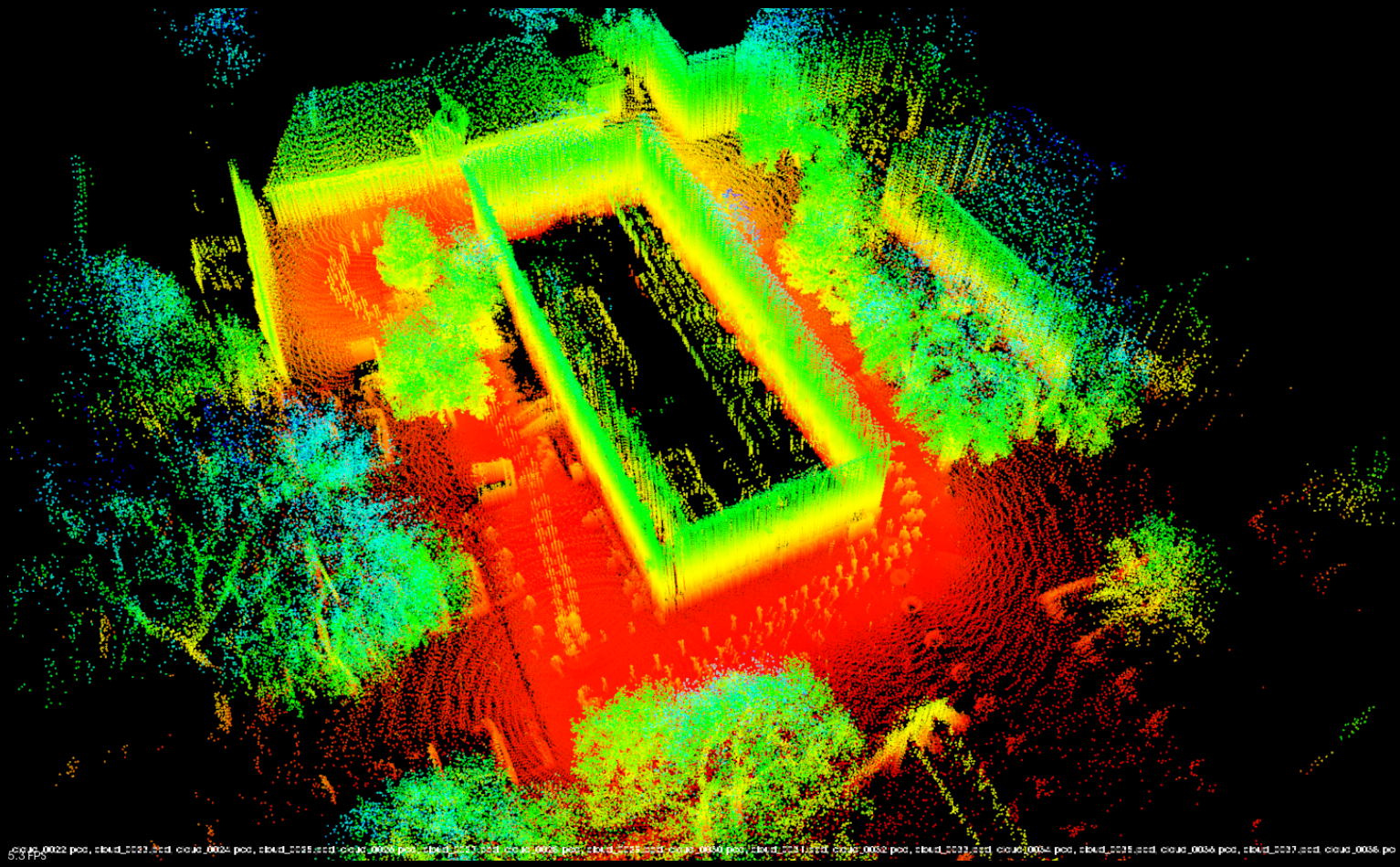


```
pcl_lum ../icp/cloud_0{022..130}.pcd
```

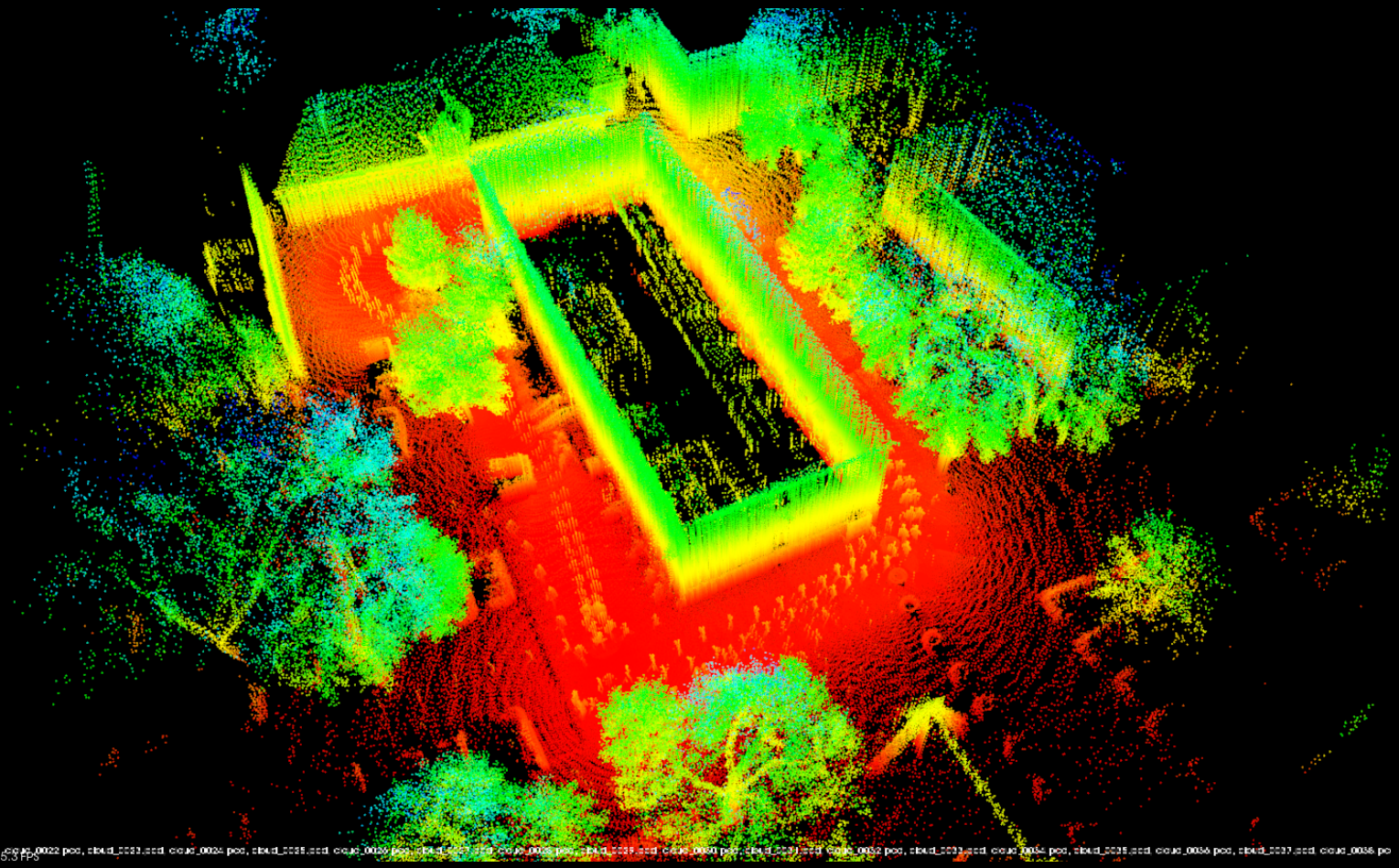
ELCH



```
pcl::registration::ELCH<PointType> elch;
elch.setReg (icp);
for (int i = 1; i < n; i++)
{
  elch.addPointCloud (cloud[i]);
}
elch.setLoopStart (first);
elch.setLoopEnd (last);
elch.compute ();
```

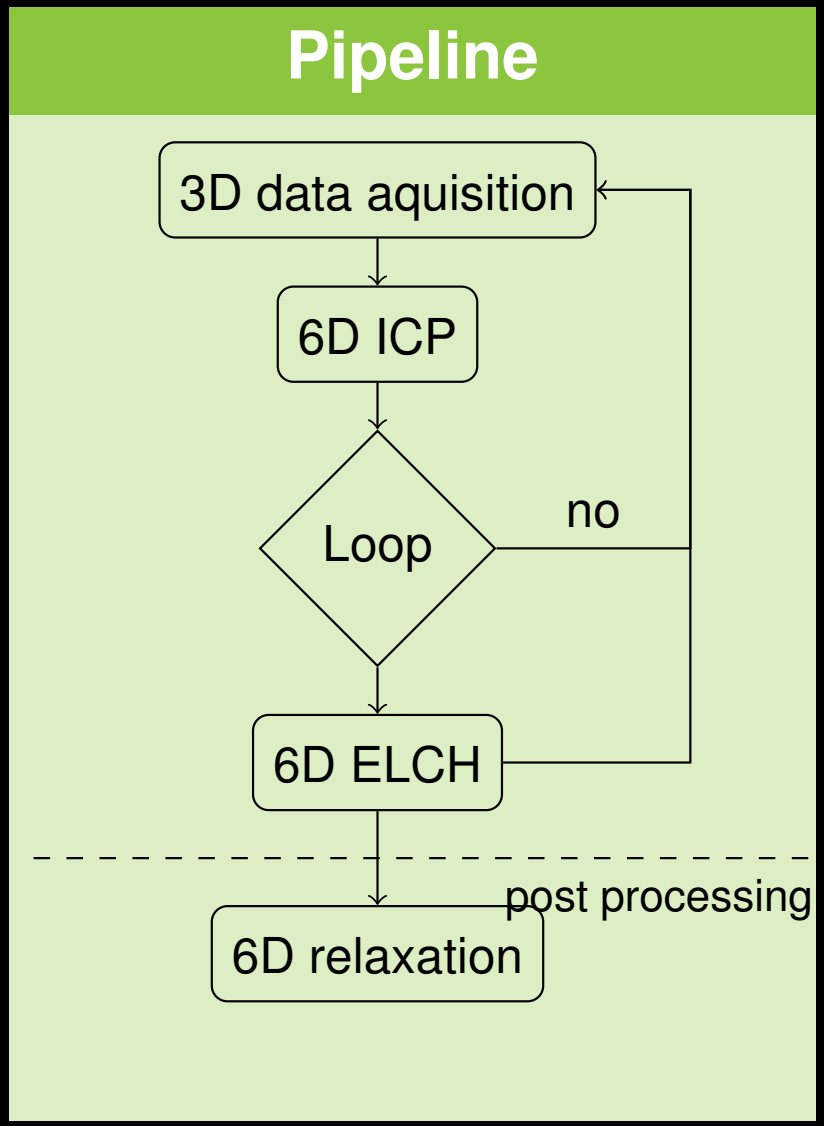


```
pcl_icp -d 0.75 -r 0.75 -i 50 ../data/cloud_0{022..130}.pcd
```



```
pcl_elch -d 1.0 -r 1.0 -i 100 ../icp/cloud_0{022..130}.pcd
```

Global Relaxation



6D relaxiation:

- 6D Lu & Milios (LUM)
- G2O
- TORO

thanks to Frits Florentinus

Results