Point Cloud Library (PCL)

Radu B. Rusu

over 110 developers/contributors worldwide!... and counting

July 1, 2011
What is this all about?
Before we begin...

1. name is not important (!!!). We can call this:
1. name is not important (!!!). We can call this:
   - 3D Library
Before we begin...

1. **name** is not important (!!!). We can call this:
   - 3D Library
   - 3D Toolkit
Before we begin...

1. **name** is not important (!!!). We can call this:
   - 3D Library
   - 3D Toolkit
   - 3D Point Cloud Toolkit
Before we begin...

1. **name** is not important (!!!). We can call this:
   - 3D Library
   - 3D Toolkit
   - 3D Point Cloud Toolkit
   - Point Cloud Library
Before we begin...

1. **name** is not important (!!!). We can call this:
   - 3D Library
   - 3D Toolkit
   - 3D Point Cloud Toolkit
   - Point Cloud Library
   - Point Cloud Toolkit
Before we begin...

1. **name** is not important (!!!). We can call this:
   - 3D Library
   - 3D Toolkit
   - 3D Point Cloud Toolkit
   - Point Cloud Library
   - Point Cloud Toolkit
   - Poop 3D... 😊
Before we begin...

1. **name** is not important (!!!). We can call this:
   - 3D Library
   - 3D Toolkit
   - 3D Point Cloud Toolkit
   - Point Cloud Library
   - Point Cloud Toolkit
   - Poop 3D... 😊

2. **what matters**: we need 3d infrastructure for the masses!
Before we begin...

1. **name** is not important (!!!). We can call this:
   - 3D Library
   - 3D Toolkit
   - 3D Point Cloud Toolkit
   - Point Cloud Library
   - Point Cloud Toolkit
   - Poop 3D...

2. what matters: **we need 3d infrastructure for the masses!**

3. analogy to **Boost**: modular, compact, efficient, fast, open
Goal: Solve n-D perception problems!
What is he talking about? What is n-D perception?

- at the very least $n = 3$: x,y,z.
- + RGB = 4D
- + normal information = 7D
- + surface curvature = 8D
- + ... = n-D
The world is not 2D

there’s no robotics without 3D geometry
2D thinking might be slowing us down

- great progress in 2D computer vision

- still poised by false positives. **not good enough** for robotics.
3D has arrived!

- Kinect was just the beginning
- Asus XtionPRo is next
- other devices coming soon

- 3D pocket cameras!
- high resolution, decent SNR
- potential for good close range 3D
- at some point => back to stereo!
Kinect enables robotics

© UC Berkeley, MIT, Willow Garage
Why

2. How

3. PCL

4. Apps

5. Specifics
Why  How  PCL  Apps  Specifics

http://pointclouds.org/
What is PCL (Point Cloud Library)?

PCL is:

- a large scale, open project\(^1\) for 3D point cloud processing

We are basically changing our attitude from this:

---

\(^1\) source code is BSD licensed, management process is completely open
What is PCL (Point Cloud Library)?

to this:

This means:

- their complexity is equally high
- many things in perception (that we do not think of) could help robotics
What is PCL (Point Cloud Library)?

- leverage the **broader** (non-robot) 3D perception community
- help advance **specific goals** in personal robotics
- many people don’t care about robotics, but they care about perception
- bring them to robotics later
What is PCL (Point Cloud Library)?

- Open source
- BETTER infrastructure
- Free beta testers
- More developers
- Community. More users.
Point Cloud Library

Statistics

www.pointclouds.org

- News
- Videos and examples
- Tutorials and support
- Facebook/Twitter

Statistics (3 months):
- over 500,000 views
- over 90,000 visits
- 6 continents
- 138 countries
- over 4,000,000 SVN repository hits
An incredible community-driven push for 3D!

The project is currently supported by engineers and scientists from a number of organizations, geographically distributed all around the world, including:

Please see the developers website at [http://dev.pointclouds.org/](http://dev.pointclouds.org/) for more details.

over 110 active developers and contributors world-wide (!)
Why How PCL Apps Specifics

PCL (3/3)

dev.pointclouds.org

► Source repository, Downloads
► Issue tracking
► Wiki
► Mailing lists: pcl-users@ & pcl-developers@
175 out of 417 organizations accepted only (!)
PCL accepted (!) 😊
got 11 slots
conferences: RSS, IROS, ICCV, CVPR

► **RSS** - Robotics Science and Systems :: July 1

► **IROS** - Intelligent Robots and Systems :: September 25

► **ICCV** - International Conference on Computer Vision
Demo Applications (1/5)

PCL Application Examples :: ROS3D contest
Demo Applications (2/5)

PCL Application Examples :: ROS3D contest

© Garratt Gallagher, MIT
Demo Applications (3/5)

PCL Application Examples :: ROS3D contest
Demo Applications (4/5)

PCL Application Examples :: ROS3D contest

Altitude Control Using Kinect Data
Demo Applications (5/5)

PCL Application Examples :: ROS3D contest

© University of Ulster
1. Why
2. How
3. PCL
4. Apps
5. Specifics
PCL 1.0.1

PCL 1.0.1 is a patch release, API compatible with 1.0. Here's a few of the release highlights:

- Please note that version 1.0.0 had a flaw when creating ASCII pcir files. This version includes the tool pcl_convert_NaN nan to fix this.
- Added VTK file visualization to pcl_view
- Adding the cmath< pcl<FLANN headers, thus reducing compile time for user code.
- Fixed integralimage<estimation
- Tutorial updates and fixes + new tutorials. Changed tutorial structure to split CPP files from RST text.
- Better doxygen documentation for many functions.
- Fixed a bug in ConditionalRemoval<filter where the keep_organized condition was reversed.
- Removed BorderDescription and <Histogram< C> from the list of explicit template instantiations.
- Added PointXY point registration macros.
- Added ExtractIndicesSelf unit test.
- Fixed a list of alignment issues on 32-bit architectures.
- Made ASCII files more readable when data consists of multiple lines.
- Internal changes for PCDReader/PCDWriter, where NaN data is represented as "nan".
- Speed up compilation with MSVC by adding PPM for multiprocessor builds.
- Added voxel grid command line tool filter.
- Issues fixed: #242, #267, #297, #323, #236, #226, #246, #234, #216, #196, #219, #207, #194, #192, #183, #178, #154, #134, #145, #155, #172
- Added support for PathScale path< compiler.
- Added support for Intel icc C+ compiler.
- Added support for GCC 4.6 C+ compiler.
- Added preliminary support for Clang C+ compiler.
- FixedPCL cmake and PCLConfig cmake completed.

New to PCL? Visit our downloads page and give it a try. If you're looking for ideas on how to get started, be sure to check out our growing collection of tutorials.
PCL (Point Cloud Library) structure

PCL

- uses **SSE** optimizations for fast computations
- uses **OpenMP** for parallelization (Boost.MPI to come!)
- data passing between modules using **shared pointers**
- uses **CUDA** for GPGPU programming (trunk only atm)
- unit tests, examples, tutorials (!)
- **apps**: higher level + integration
PCL (Point Cloud Library) structure

...split into a collection of smaller, modular C++ libraries:

- **libpcl_keypoints**: nD interest points
- **libpcl_features**: nD feature descriptors
- **libpcl_surface**: surface meshing/reconstruction techniques
- **libpcl_filters**: point cloud data filters and smoothing
- **libpcl_io**: I/O operations, 3D camera drivers (e.g., Kinect)
- **libpcl_kdtree**: fast nearest neighbor operations
- **libpcl_octree**: downsampling, compression, change detection
- **libpcl_range_image**: efficient 3D operations
- **libpcl_sample_consensus**: RANSAC, MSAC, MLESAC, planes, spheres, etc
- **libpcl_segmentation**: model segmentation operations
- **libpcl_registration**: point cloud registration methods
- **libpcl_visualization**: 2D/3D visualization library
1.0.1 released

- full support for OpenNI (Kinect, PrimeSense, Asus XTion)

- Linux, Windows, MacOS support 🌍💨💻 (next: Android)
PPG: Perception Processing Graphs

- Philosophy: *write once, parameterize everywhere*
- PPG: Perception Processing Graphs

```
Clusters (1..N) (PointIndices) -> Normals (PointCloud) -> Model (ModelCoefficients)
Indices (rest) (PointIndices) -> Input (PointCloud) -> Inliers (PointIndices)
Projected inliers (PointCloud)
```
Why PPG?

- Algorithmically:
  door detection = table detection = wall detection = ... 
- the only thing that changes is: parameters (constraints)!
Good API practices simplify development and testing:

```cpp
cpcl::Feature<PointT> feat;
feat = pcl::Normal<PointT> (input);
feat = pcl::FPFH<PointT> (input);
feat = pcl::BoundaryPoint<PointT> (input);
...
feat.compute (&output);
...```
Project activity

Why

How

PCL

Apps

Specifics

PCL 1.0.1 (8/8)

Point Cloud Library (PCL) - http://pointclouds.org
http://pointclouds.org/