Development Plan for OSS World Challenge 2011

Registration No.	2011- ※Registration No. need not be written.
Program title	Point Cloud Library (PCL)

※ Follow the content below without fail.

1. Program Overview

1) Development goals (background, aims etc.)

The Point Cloud Library (or PCL) is a **large scale, open project for processing 3D point cloud data**, and is designed to be used as the processing backbone of large-scale 3D systems.

Many higher level applications that require solutions involving 3D perception will benefit from using PCL, especially in the field of mobile and personal robotics, but also in computer graphics, photogrammetry and sensing, GIS, etc.

With the advent of inexpensive 3D imaging sensors such as the Microsoft Kinect, there has been an explosion of interest in 3D point clouds across the globe. With all of the new users becoming interested in the field, there has been a surge in the demand for efficient and easy-to-use algorithms for working with 3D data. PCL has been designed to address these needs. Our primary goal is to make PCL a standard which provides a ***solid, well-tested, and widely-used*** infrastructure for **implementing 3D perception functionality** for any academic or business application.

PCL is released under the terms of the **BSD license** and is open source software. It is free for commercial and research use.

PCL is officially a 6 months old project, launched in March 2011. In less than 6 months we managed to gather around 30 institutions (universities, research labs, and companies) around the world to contribute to the project (see http://www.pointclouds.org/about.html). We are proud to be part of such a young, energic, and dynamic community.

2) System configuration

The PCL libraries, demos, and applications are all compatible with multiple operating systems including Microsoft Windows, Linux, and Mac OSX. Support for Android is on the way.

PCL depends on:

- the Boost C++ libraries for cross-platform threading and IO support,
- Eigen for matrix-vector operations and optimized math operations,
- FLANN for fast kd-tree/nearest neighbors searches,
- CMinpack for solving systems of equations and Levenberg-Marquardt solutions,
- and finally, VTK for visualizations.

All the above dependencies allow us to re-use a lot of good open source software, without reinventing the wheel. We thank our collaborators from the above mentioned projects, and we

are in a continuous dialogue with them for improving their own infrastructures/projects. For more information, please check http://www.pointclouds.org/downloads/.

3) Menus

PCL aims to provide a complete solution for processing 3D data like point clouds, meshes, volumetric data, etc.

Applications can use PCL to: estimate geometric properties of a scene, fit 3D shape models and complex mesh surfaces, recognize objects, and align scans from different points of view into a panorama, to name a few. PCL also contains a flexible grabber interface for 3D OpenNI-compliant devices, and a lightweight and very fast visualization infrastructure.

PCL is organized as a core set of base classes along with a collection of modular libraries that provide functionality in a wide range of areas. The current algorithmic capabilities include: feature estimation, filtering, efficient nearest neighbor search, keypoint detection, registration, robust geometric fitting, clustering and segmentation, and surface reconstruction. The current input/output library supports: file storage, 3D image capture, and point cloud data compression and transmission. The current visualization library provides tools for rendering point clouds, surface normals, and common geometric shapes in an interactive display. There is also an “applications” library that contains technology demonstrations and examples of common higher level applications.

For more information, please visit http://www.pointclouds.org/documentation.

4) Language used for development

C++ (compatible with GCC, Intel C++ compiler, Microsoft Visual Studio, PathCC, etc)

CMake (for the build system)

5) Systems used

We develop our code on regular i7 PCs, but we take great care to test it on a variety of platforms, from AMD, to Intel and ARM. We are also porting our core to GPUs using NVidia hardware at the moment. As previously mentioned, Android ports are in the works, which means we test our software on tablets and mobile phones as well.

6) Plan for each development stage

- **Before Nov 2010:** Radu B. Rusu has a set of software tools that he used for his PhD studies.
- **Nov 2010:** Microsoft launched the Kinect, Radu and a few people release the first version of PCL that works as a standalone library, as part of ROS (the Robot Operating System).
- **March 2011:** PCL is officially launched and the website www.pointclouds.org goes live. Between November 2010 and March 2011, the system is completely redesigned, a lot of documentation and tutorials are written, plus many code examples. All algorithms are unit tested and verified, and the PCL project contains about 12 modules.

• **May 2011:** A new testing infrastructure is established to ensure that PCL can operate on Linux, Mac OS X, and Windows. Binary installers are created to simplify installation on all operating systems. All known issues are resolved. PCL 1.0.0 is officially launched. PCL starts receiving support from **NVidia**, and is accepted as part of the **Google Summer Of Code** (GSOC) program.
• **June 2011:** A demonstration of PCL's GPU optimizations is presented at **CVPR** 2011 in Colorado Springs. PCL 1.0.1 is released**.**
• **July 2011:** A day-long tutorial session on PCL is held at **RSS** 2011 in Los Angeles with over 50 participants in attendance. Following the GSOC model, **Toyota** commits to funding new developers. PCL 1.1.0 is released.

7) Number of personnel input and work assignment

Please see list_of_team_members_2011.ods.

2. Long-term prospects of the program developed (No specific form is required.)

2.1. Community Building

We are reaching out to the robotics and computer vision communities. We believe that PCL will be an attractive tool for many different researchers and developers, and we want to show our potential users what PCL is and what it can do for them. We also want to ensure that we have a constant stream of feedback from the community, so that we can maintain a clear idea of what our users need from PCL and how it can improve. In light of these goals, we are very excited to have been accepted to present tutorials at four of the most prestigious conferences in Robotics, Computer Vision, and 3D processing: RSS, IROS, ICCV, GTC.

But we can't focus only on users. We believe that to achieve the kind of adoption rate that will make PCL an industry standard, we must focus on growing our developer community in addition to our user-base. To succeed, we need new developers to join the project and contribute; this can only happen if contributors have respect for the overall framework, find utility in the library's existing tools, and can feel a sense of ownership over the library as a whole. We are working on tools to facilitate a very open and transparent development process, and we are striving to create an inviting developer culture.

2.2. Development Roadmap

Library improvements: In the months ahead, we plan to introduce a number of major improvements to PCL. Our goal for this new release, PCL 2.0, is a state-of-the-art object-oriented design that is flexible, extensible and efficient enough to make PCL ***the*** choice for beginners as well as experts. To achieve this, we are working together with PCL developers all over the world to find a solution that fits the requirements of the largest possible number of users. It's essential that this next version of PCL is able to retain the power and efficiency of the current library while also introducing a simpler API that is flexible, extensible, and easier to use and maintain.

We will then begin work on an extensive list of new features. While PCL has already been widely adopted in many different applications, we acknowledge that the current suite of algorithmic work

that it contains can and should easily be extended. We will be harnessing the skills and passion of our growing community to make significant new advancements to PCL's capabilities. Because this will be a period of very rapid development, we are planning to make major software releases every three months.

Developer resources: In addition to the implementation of new algorithms, we will be working to create a collection of resources that will make it easier for new developers to contribute to PCL. Because there are many more desired features than any of the core developers/maintainers have time to implement and test, we are creating a public document describing all of the desired new functionality, priorities, and links to relevant papers. This will give potential contributors an easy way to see how they can help expand PCL. We are also creating detailed developer guides that will cover topics such as: terminology, coding style, best practices, etc. This ensures that new developers have a consistent set of guidelines to follow and will help create a more cohesive library. Finally, we are hosting a collection of developer blogs, and we encourage all developers to post regular updates. We believe this will facilitate the sharing of our work and ideas and keep the broader community involved in the development process.

User resources: It is also vital to provide resources to help our users to learn about PCL's capabilities and how to employ them effectively. We will continue to maintain a fully documented and browsable online API, with an emphasis on making it easy to discover and use new features. We will also publish a series of progressive tutorials (based on our full-day tutorial curriculum) designed to chain together and form a comprehensive introduction to the library. To complement this, we will be also be continually expanding our large collection of smaller self-contained tutorials. We are making all of this information available on the pointclouds.org website.

Extensions to other languages: If we want to harness the talents of a broader swath of the Computer Vision community, then we need to speak their language - and to a large extent, that language is Matlab. We plan to develop a Matlab interface for PCL to enable Matlab-only developers to use our core tools when developing new higher-level algorithms. We believe this will help accomplish two things. First, if researchers are able to develop powerful new algorithms by using our basic infrastructure, we are helping to accelerate progress in perception. And second, we expect PCL-based Matlab routines to serve as useful prototypes for new PCL algorithms. Although a researcher who only works in Matlab is unlikely to contribute directly to PCL, a Matlab interface will allow us to indirectly benefit from his or her work: after all, a Matlab script written using extensive calls to our existing API will likely be much easier to port to C++ and incorporate into PCL than an algorithm written using pure Matlab. We have already found developers in the community that will be willing to create and maintain such Matlab interfaces to PCL, and we will be collaborating with them on guiding the process.

Of course, the same arguments can be made for other scripting languages, like Python or even D. Although we can't pursue every possibility, we will be listening closely to the needs of the community, and we are open to developing scripting language interfaces for any language that would be likely to increase our user and contributor base enough to offset its development and maintenance costs.

2.3. OpenPerception

PCL and OpenCV (its 2D computer vision library sister) are two sides of the same coin. While the distinction between 2D (OpenCV) and 3D (PCL) may seem like a clear one, in practice, most per-

ception problems have both 2D and 3D aspects. We believe that the divide between OpenCV and PCL must be overcome and that uniting the two libraries would eliminate redundancies and simplify the development experience for our users in the perception and robotics communities. While we feel it would be premature to launch this initiative before the algorithms in PCL have matured, the requirements for this eventual unification will be one of our foremost considerations during the design and development of PCL 2.0. We will be working closely with the OpenCV developers towards this goal.

※ Name the file ‘development plan-team name (team leader’s name)’ before saving it, please.