RTC:PCL User Guide

Geoffrey Biggs

July 15, 2011

Contents

1 Introduction

RTC:PCL is a library of components for working with point clouds. It uses the PCL library[1] for data capture, processing and visualisation.

This software is developed at the National Institute of Advanced Industrial Science and Technology. Approval number H23PRO-????. This software is licensed under the Lesser General Public License. See COPYING and COPYING.LESSER in the source.

2 Requirements

RTC:PCL requires the C++ version of OpenRTM-aist-1.0.0.

RTC:PCL uses the CMake build system[2]. You will need at least version 2.6 to be able to build the component.

RTC:PCL requires PCL version 1.1.

RTC:PCL can optionally take advantage of the RTM:DDS transport. If it is installed, you can compile the components with DDS support by enabling the `DDS_SUPPORT` option in CMake.

3 Installation

3.1 Binary

Users of Windows can install the components using the binary installer. This will install the components and all necessary dependencies. It is the recommended method of installation in Windows.

1. Download the installer from the website.
2. Double-click the executable file to begin installation.
3. Follow the instructions to install the component.
4. You may need to restart your computer for environment variable changes to take effect before using the component.

The components can be launched by double-clicking the relevant executables. For example, `rtc_pclviewer_standalone` will launch the cloud viewer component. Each component is also available as a shared library that can be loaded into a manager. The initialisation function for each component is `rtc_init`.

3.2 From source

Follow these steps to install :

1. Download the source, either from the repository or a source archive, and extract it somewhere.

   ```
   tar -xvzf rtcpcl.tar.gz
   ```

2. Change to the directory containing the extracted source.

   ```
   cd rtcpcl-1.0.0
   ```

3. Create a directory called “build”:

   ```
   mkdir build
   ```

[1] http://www.pointclouds.org
[2] http://www.cmake.org/

Parameter	Effect
corba	Enable input/output of data on the DataPort ports (typically using the CORBA transport).
dds	Enable input/output of data on the DDSPort ports. DDS support must have been compiled in and the RTM:DDS transport must be available.
pointer	Enable input/output of data on the PointerPort ports. PointerPort support must have been compiled in and the RTM:Pointer transport must be available. [Not implemented.]

Table 1: Common configuration parameters.

4. Change to that directory.

   ```
   cd build
   ```

5. Run cmake or cmake-gui.

   ```
   cmake ../
   ```

6. If no errors occurred, run make.

   ```
   make
   ```

7. Finally, install the components. Ensure the necessary permissions to install into the chosen prefix are available.

   ```
   make install
   ```

8. The install destination can be changed by executing ccmake and changing the variable `CMAKE_INSTALL_PREFIX`.

   ```
   ccmake ../
   ```

The components are now ready for use. See the next sections for instructions on configuring each components.

RTC:PCL components can be launched in stand-alone mode by executing the appropriate `*_standalone` executable (installed into `${prefix}/bin`). Alternatively, `librtc*.so` can be loaded into a manager, using the initialisation function `rtc_init`. This shared objects can be found in `${prefix}/lib` or `${prefix}/lib64`.

4 Common Configuration

Some configuration parameters are common across all components in RTC:PCL. These are described in Table 1.

Each port in each component is duplicated for each transport that was enabled at compile time. This means that, for example, if you compiled with both CORBA and DDS support, you will get two of every port; one for each transport type. These ports are prefixed with the name of the transport they correspond to.

5 RTCPCLViewer

RTCPCLViewer is a component for visualising point clouds. It can visualise virtually any point cloud that PCL's visualiser can handle. It is capable of visualising multiple point clouds, either in a single viewport or in separate viewports. The input ports are configured at activation-time based on the value of the `ports` configuration parameter.

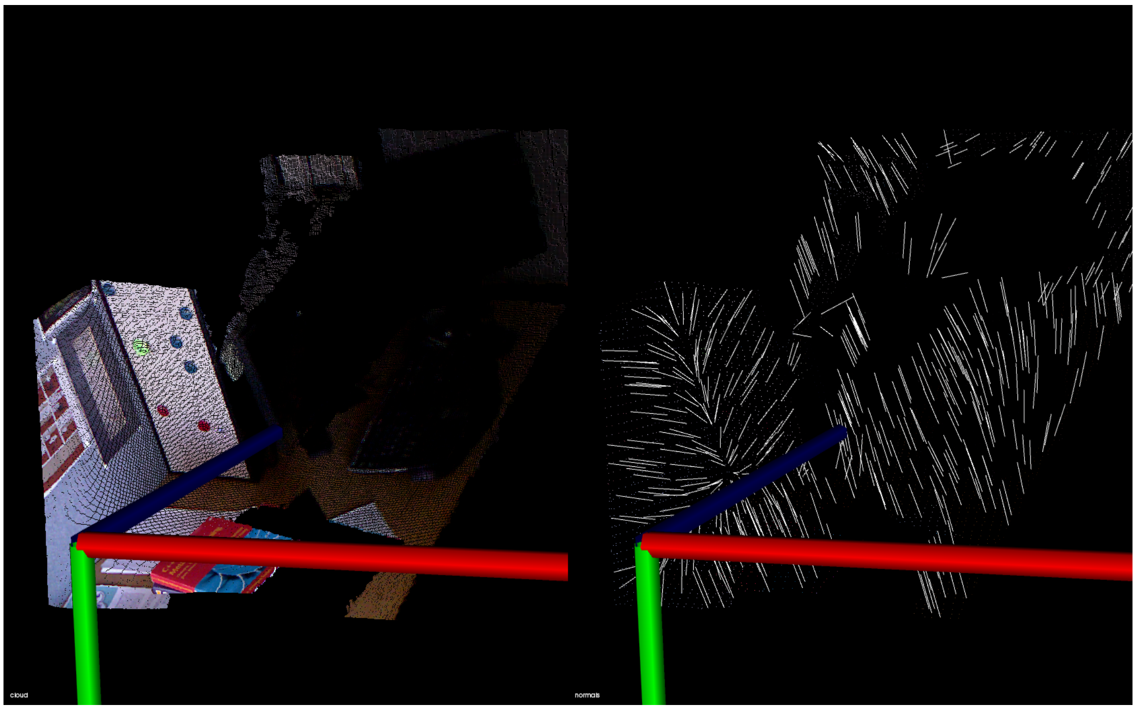

Figure 1: RTCPCLViewer showing a point cloud on the left and its normals on the right.

Parameter	Effect
axes_scale	Set the scale of the axes displayed in the viewports. Set to 0 to disable.
ports	Set the ports that the component will provide, including how the received data is displayed.
show_timestamps	Print timestamps for the received data in the terminal. Useful for checking the lag of the visualisation.

Table 2: Available configuration parameters for RTCPCLViewer.

5.1 Configuration

The available configuration parameters are described in Table 2.

5.2 Ports

RTCPCLViewer does not provide any ports by default. You must set the `ports` configuration parameter to get the the ports you require.

The format of the `ports` parameter is:

`name:viewport:type:colour,...`

where:

name Name of the port.

viewport Viewport number. Set to 0 to display in all viewports.

type The display type for the cloud. One of "p" (plain point cloud), "n;x;y" (normals, where "x" must be an integer giving the normals level and y must be a floating point number giving the normals scale), or "c;x;y" (principal curvatures, where x and y have the same meaning as for normals).

Parameter	Effect
unseen_to_max	When enabled, unseen pixels will be displayed in the colour for the maximum distance.

Table 3: Available configuration parameters for RTCRIViewer.

Name	Type	Data type	Purpose
ri	Input	PointCloud	Receives the range-image-organised point cloud to display.

Table 4: Available ports for the RTCRIViewer component.

colour The colourisation method for the cloud. One of “p” for none, “a” for random, “r” for using the cloud’s RGB data, “x.y.z”, where x, y and z are integers between 0 and 255 depicting red, green and blue colour levels, for a custom colour, and “”x””, where “x” is a field name, for using a custom field for colourisation.

Apart from the name, any parameter may be left black to get the default. Multiple port specifications should be separated by commas. For example, the following string creates three ports. The first, named “pc1”, is a plain point cloud with RGB data, which it is using for colourisation. It is displayed in viewport 1. The second, “pc2”, contains normals information. It is displayed in all viewports with a random colour. The third, named “pc2”, is another plain point cloud. It is displayed in viewport 3 using a custom colour (green).

```
pc1:1:p:r,pc2::n;10;0.1:a,pc3:3:p:0.255.0
```

6 RTCRIViewer

RTCRIViewer is the range image equivalent of RTCPCLViewer. It is used to visualise range images. These are point clouds where each three-dimensional point corresponds to a pixel in a two-dimensional image.

6.1 Configuration

The available configuration parameters are described in Table 3.

6.2 Ports

The ports provided by the RTCRIViewer component are described in Table 4.

7 RTCPCLOpenNI

RTCPCLOpenNI provides an input component for reading from OpenNI-compatible depth cameras, such as the Microsoft Kinect. It can provide the data as a point cloud or a range image.

7.1 Configuration

The available configuration parameters are described in Table 5.

7.2 Ports

The ports provided by the RTCPCLOpenNI component are described in Table 6.

Parameter	Effect
device	The device number to open. Must be the path to the device node, or the index of the device, such as "#1" or "#2".
output_xyz	Enable outputting a plain XYZ point cloud, with no colour information.
output_xyzrgb	Enable outputting an XYZRGB point cloud.
output_ri	Enable outputting a range image.

Table 5: Available configuration parameters for RTCPCL.

Name	Type	Data type	Purpose
xyz	Output	PointCloud	Publishes the XYZ point cloud.
xyzrgb	Output	PointCloud	Publishes the XYZRGB point cloud.
ri	Output	PointCloud	Publishes the range image.

Table 6: Available ports on the RTCPCLOpenNI.

8 RTCPCDLoad

RTCPCDLoad loads point clouds stored in PCD files and sends them over its output port. It is useful for replaying a previously-captured and saved (such as with RTCPCDSave) point clouds.

8.1 Configuration

The available configuration parameters are described in Table 7.

8.2 Ports

The ports provided by the RTCPCDLoad component are described in Table 8.

9 RTCPCDSave

RTCPCDSave performs the opposite function of RTCPCDLoad. It saves a received point cloud into a PCD file.

9.1 Configuration

The available configuration parameters are described in Table 9.

9.2 Ports

The ports provided by the RTCPCDSave component are described in Table 10.

10 RTCPCLCuboid

This component generates a random point cloud where all points fit within a cuboid. The dimensions of the cuboid and the number of points can be configured. A new random point cloud is generated each time the component executes.

10.1 Configuration

The available configuration parameters are described in Table 11.

Parameter	Effect
write_once	Only write the point cloud once. When disabled, the point cloud will be written every time the component executes.
pcd_file	File name of the PCD file to load the point cloud from.
point_type	The type of point used by the cloud stored in the PCD file.

Table 7: Available configuration parameters for RTCPCDLoad.

Name	Type	Data type	Purpose
out	Output	PointCloud	The point cloud loaded from the PCD file.

Table 8: Available ports on the RTCPCDLoad component.

10.2 Ports

The ports provided by the RTCPCLCuboid component are described in Table 14.

11 RTCPCLRainbowTube

Like RTCPCLCuboid, this component produces a sample point cloud. In this case, the point cloud is a tube of regularly-spaced points. The length of the major and minor axes, as well as the length of the overall tube, can be configured. The tube is coloured in RGB.

11.1 Configuration

The available configuration parameters are described in Table 13.

11.2 Ports

The ports provided by the RTCPCLCuboid component are described in Table 14.

12 RTCPCLVoxelFilter

RTCPCLVoxel filter is a point cloud processing component. Its purpose is to down-sample a point cloud, reducing the density of the points using a voxel filter. Point clouds are often over-sampled when they come out of the sensor. Passing them through a voxel filter can reduce the cloud complexity while maintaining enough geometric accuracy for subsequent processing. This allows those subsequent algorithms to execute much faster.

12.1 Configuration

The available configuration parameters are described in Table 15.

12.2 Ports

The ports provided by the RTCPCLVoxelFilter component are described in Table 16.

13 RTCPCLNormals

RTCPCLNormals illustrates creating a component to deal with specific types of point clouds. It takes in an XYZRGB point cloud, calculates the normal information from the points, and outputs

Parameter	Effect
binary	Save the point cloud in binary format instead of ASCII format.
write_once	Only save the first point cloud received. When disabled, every point cloud received will be saved, overwriting the previous point cloud.
pcd_file	File name of the PCD file to save the point cloud into.

Table 9: Available configuration parameters for RTCPCDSave.

Name	Type	Data type	Purpose
in	Input	PointCloud	Receives the point cloud to save.

Table 10: Available ports on the RTCPCDSave component.

an XYZRGBNormal cloud containing both the XYZ information and the Normal information. This cloud can be directly displayed in RTCPCLViewer.

13.1 Configuration

The available configuration parameters are described in Table 17.

13.2 Ports

The ports provided by the RTCPCLNormals component are described in Table 18.

14 RTCPCLBase

RTCPCLBase is not a component in its own right. It is the base class used by one-input, one-output processing components such as RTCPCLVoxelFilter.

The base class is available as a library installed by RTC:PCL. This library is called `rtcpclbase`, and the header file is `rtcpclbase.h`. By including this library, you can rapidly create new point cloud processing components. For an example of using the library, see the source for the RTCP-CLVoxelFilter component.

14.1 Configuration

RTCPCLBase does not provide any configuration parameters of its own.

14.2 Ports

RTCPCLBase provides the ports described in Table 19. These are available for use by components that inherit from RTCPCLBase.

15 PointCloud type

The PointCloud data type used by RTC:PCL to transport point clouds is defined in `pointcloud.idl`. This file is installed into `${prefix}/include/rtcpcl/idl`, so you can use it in your own components. To make the use of this data type easier, RTC:PCL also installs a library containing the data type compiled in each transport RTC:PCL is compiled to support (by default, this is CORBA and DDS). By linking to this library, called `rtcpcl_pointcloud_type`, and including the appropriate headers for the transports you want to use (such as `pointcloud.hh` for the default CORBA transport), you can immediately gain the use of this data type.

Parameter	Effect
count	Sets the number of points that will be generated.
low_x	The lower bound on the x-coordinate of the points.
high_x	The upper bound on the x-coordinate of the points.
low_y	The lower bound on the y-coordinate of the points.
high_y	The upper bound on the y-coordinate of the points.
low_z	The lower bound on the z-coordinate of the points.
high_z	The upper bound on the z-coordinate of the points.

Table 11: Available configuration parameters for RTCPCLCuboid.

Name	Type	Data type	Purpose
out	Output	PointCloud	Publishes the randomly-generated point cloud.

Table 12: Available ports on the RTCPCLCuboid component.

16 Examples

Example configuration files are provided in the `${prefix}/share/rtcpcl/examples/conf/` directory.

Parameter	Effect
step	Sets the distance along the z-axis between each ring of points.
angle_step	Sets the angle around the z-axis between each column of points.
x_radius	Sets the length of the ellipsoid axis on the x-axis.
y_radius	Sets the length of the ellipsoid axis on the y-axis.
length	Sets the length of the tube along the z-axis.

Table 13: Available configuration parameters for RTCPCLRainbowTube.

Name	Type	Data type	Purpose
out	Output	PointCloud	Publishes the randomly-generated point cloud.

Table 14: Available ports on the RTCPCLCuboid component.

Parameter	Effect
res_x	Sets the x-axis resolution of the voxel grid.
res_y	Sets the y-axis resolution of the voxel grid.
res_z	Sets the z-axis resolution of the voxel grid.
point_type	Sets the point type to be filtered. For example, "xyz" or "xyzrgb".

Table 15: Available configuration parameters for RTCPCLVoxelFilter.

Name	Type	Data type	Purpose
in	Input	PointCloud	Receives the point cloud to be filtered.
out	Output	PointCloud	Publishes the filtered point cloud.

Table 16: Available ports on the RTCPCLVoxelFilter component.

Parameter	Effect
radius	Sets the search radius for finding neighbour points to calculate normals with.

Table 17: Available configuration parameters for RTCPCLNormals.

Name	Type	Data type	Purpose
in	Input	PointCloud (XYZRGB)	Receives the point cloud to be filtered.
out	Output	PointCloud (XYZRGBNormal)	Publishes the filtered point cloud.

Table 18: Available ports on the RTCPCLNormals component.

Name	Type	Data type	Purpose
in	Input	PointCloud	Receives the point cloud to be processed.
out	Output	PointCloud	Publishes the processed point cloud.

Table 19: Available ports on the RTCPCLBase component.