Module Interface Specification for PCD: Partially Covered Detection of Obscured People using Point Cloud Data

Team #14, PCD
Tarnveer Takhtar
Matthew Bradbury
Harman Bassi
Kyen So

January 17, 2025

1 Revision History

Date	Version	Notes
Jan 17, 2025	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

SRS Documentation can be found on GitHub.

symbol	description
SRS	Software Requirements Specification
PCD	Partially Covered Detection Program
PCL	Point Cloud Library

Contents

3 Introduction

The following document details the Module Interface Specifications for Partially Covered Detection (PCD) software. The sections in this document describes each module in our software and how each module interacts with each other. Additional information and documentation can be found in System Requirement Specifications (SRS).

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/takhtart/PCD.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by PCD: Partially Covered Detection of Obscured People using Point Cloud Data.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
PointXYZRGBA	$\mathbb{P}$	point cloud data in the PCL Library

The specification of PCD uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, PCD uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	Kinect Stream
Behaviour-Hiding Module	Application Control Input Data Read Input Classifier Input Classifier Ranking Bounding Box Display
Software Decision Module	Point Cloud Data Structures Input Processing Command Line Interface Graphical User Interface

Table 1: Module Hierarchy

6 MIS of Kinect Stream

6.1 Module

kinect

6.2 Uses

- Input Data Read 8
- Point Cloud Data Structure 12

6.3 Syntax

6.3.1 Exported Constants

None.

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
kinect	video frame stream from kinect in BGRA format and depth frame stream	- pcl::PointXYZ	-
		- pcl::PointXYZRGBA	
		- pcl::PointXYZI	
		- pcl::PointXYZRGB	

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

- Size of the room (affects # of points in point cloud)
- Lighting (affects # of points and colour of the point cloud)
- Angle of the Kinect (affects layout of the points)

6.4.3 Assumptions

- User selects live stream rather than offline view
- Kinect is connected and running without issue.
- Kinect has a clear and unobstructed view of the environment (lens are not covered)

6.4.4 Access Routine Semantics

kinect():

- output: PointCloudT::ConstPtr input_cloud
- Precondition: user calls live_stream mode
- Postcondition: user terminates program or selects exit

6.4.5 Local Functions

None

7 MIS of Application Control

7.1 Module

main

7.2 Uses

- Input Data Read 8
- Input Classifier Module 9
- Command Line Interface 14
- Graphical User Interface 15
- Bounding Box Display 11

7.3 Syntax

7.3.1 Exported Constants

None.

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	None	None	-

7.4 Semantics

7.4.1 State Variables

PointcloudT::Ptr cloud

7.4.2 Environment Variables

- Processing speed of device

7.4.3 Assumptions

Device has the processing power needed.

7.4.4 Access Routine Semantics

main():

- transition: connects the final filtered cloud (cloud after processing) to the GUI module to display the output.

7.4.5 Local Functions

None.

8 MIS of Input Data Read

8.1 Module

reader

8.2 Uses

8.3 Syntax

8.3.1 Exported Constants

None.

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
reader	std::cin user_input	None.	-

8.4 Semantics

8.4.1 State Variables

- std::cin user_input : Choice that the user made on what mode to run the program in.

8.4.2 Environment Variables

8.4.3 Assumptions

The user provides a valid input ($\mathbb{Z}$) corresponding with the correct option

8.4.4 Access Routine Semantics

main():

- transition: Converts the input data into the data structure used by the Input Processing Module

8.4.5 Local Functions

None.

9 MIS of Input Classifier

9.1 Module

classify

9.2 Uses

- Bounding Box Display 11
- Point Cloud Data Structures 12

9.3 Syntax

9.3.1 Exported Constants

None.

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
classify	(PointCloudT::Ptr)cloud, (PointCloudT::Ptr) cloud-filtered	None.	-

9.4 Semantics

9.4.1 State Variables

- *cloudFiltered = *personCloud : updated cloud value for the data set
- personCloud (PointCloudT::Ptr) : the cluster that is being identified as a human within the frame.

9.4.2 Environment Variables

None.

9.4.3 Assumptions

- The cloud has been properly filtered to allow for a good reading to quickly identify the human on screen.

9.4.4 Access Routine Semantics

classify(cloud, cloudfiltered)(): transition: This module will take the the filtered values and filter down further to just identify the human within the frame and only leave the data points connected to that person. Connects the filtered point cloud to the Application Control module

9.4.5 Local Functions

None.

10 MIS of Input Classifier Ranking

10.1 Module

ranking

10.2 Uses

10.3 Syntax

10.3.1 Exported Constants

None.

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
ranking	dataPoint($\mathbb{P}$)	None.	-

10.4 Semantics

10.4.1 State Variables

- weights ($\mathbf{P}^n$): An array of size n containing the ordered weights of the pcd points

10.4.2 Environment Variables

None.

10.4.3 Assumptions

- The input point cloud data is valid.
- The classification strategy is implemented correctly to be able to order the weights

10.4.4 Access Routine Semantics

ranking(dataPoint)():

- transition: This module will take in the dataPoint and add it to the list and order it into the array. It connects the sorted array of ranking to the Input Classifier module.

10.4.5 Local Functions

None.

11 MIS of Bounding Box Display

11.1 Module

boundingBox

11.2 Uses

- Input Classifier Module

11.3 Syntax

11.3.1 Exported Constants

None.

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
boundingBox	humancloud($\mathbb{P}$), minpt($\mathbb{P}$), maxpt($\mathbb{P}$)	None.	-

11.4 Semantics

11.4.1 State Variables

- thickness ($\mathbf{R}$): An float with the thickness of the box
- $scale_factor$ ($\mathbf{R}$): Factor that adjusts the box size.

11.4.2 Environment Variables

None.

11.4.3 Assumptions

- The input cloud points are valid to provide an accurate drawing of the box.
- The filtered cloud properly depicts a human.

11.4.4 Access Routine Semantics

boundingBox(humancloud,minpt,maxpt)():
transition: This module will take in inputed data points and add draw out a box using the max and min points provided. It connects the cloud and calculated bounding box to be presented at the GUI module.

11.4.5 Local Functions

None.

12 MIS of Point Cloud Data Structures

12.1 Module

struct

12.2 Uses

- Input Processing 13
- Graphical User Interface 15

12.3 Syntax

12.3.1 Exported Constants

None.

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
struct	None.	None.	-

12.4 Semantics

12.4.1 State Variables

- typedef pcl::PointXYZRGBA PointT : Data structure designed to store each individual point in a point cloud
- typedef pcl::PointCloud<PointT>PointCloudT : Data Structure designed to store an entire point cloud

12.4.2 Environment Variables

None.

12.4.3 Assumptions

The point cloud data from kinect stream is valid

12.4.4 Access Routine Semantics

struct():

- transition: This module is the point cloud data structure for storing point cloud data such as ones captured from the kinect and ones processed by our cloud processing algorithm. It connects generalized data types that abstract point cloud data to the Input Processing and GUI modules.

12.4.5 Local Functions

None.

13 MIS of Input Processing

13.1 Module

process_cloudOCV

13.2 Uses

- Command Line Interface 14
- Graphical User Interface 15

13.3 Syntax

13.3.1 Exported Constants

None.

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
process_cloudOCV	PointCloudT::Ptr cloud	None.	-

13.4 Semantics

13.4.1 State Variables

None.

13.4.2 Environment Variables

None.

13.4.3 Assumptions

Data captured by the kinect are correctly processed and stored in the point cloud data structure

13.4.4 Access Routine Semantics

process_cloudOCV(cloud,cloud_filtered):

- transition: it removes noise, perform plane removal, and skin point detection and transitions the detected skin points and filtered cloud to the Input Classifer module.

13.4.5 Local Functions

None.

14 MIS of Command Line Interface

14.1 Module

cmd

14.2 Uses

- Input Processing Module 13
- Graphical User Interface 15

14.3 Syntax

14.3.1 Exported Constants

None.

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
cmd	None.	None.	-

14.4 Semantics

14.4.1 State Variables

None.

14.4.2 Environment Variables

- Keyboard (Used to select modes)
- Mouse (Interacts with command prompt)

14.4.3 Assumptions

- Working keyboard and mouse is connected
- Kinect is properly setup and connected
- Proper file types are uploaded

14.4.4 Access Routine Semantics

cmd():

- transition: Provides the Data Read module with the user's option of offline versus live as well as the location of the downloaded .pcd file (for offline mode).

14.4.5 Local Functions

None.

15 MIS of Graphical User Interface

15.1 Module

gui

15.2 Uses

None.

15.3 Syntax

15.3.1 Exported Constants

None.

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
gui	PointcloudT::Ptr filtered_cloud	Visulaized Point Cloud	-

15.4 Semantics

15.4.1 State Variables

- std::thread visualizer_thread (displaying the filtered cloud)
- std::thread visualizer_thread2 (displaying the original cloud)

15.4.2 Environment Variables

- Mouse (To move around within the visualized point cloud)

15.4.3 Assumptions

Point cloud was correctly processed and stored with the point cloud data structure

15.4.4 Access Routine Semantics

gui():

- output: Uses the visualizer to deploy a GUI which displays the 3D point clouds (filtered and original)

15.4.5 Local Functions

None.

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

16 Appendix

N/A

Appendix — Reflection

1. What went well while writing this deliverable?

   Everyone on the team was on track with their sections of the assignment and we were able to thick of better ways to break up some modules to make more sense.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   Getting used to the new year and so it was a slow start trying to get back into the flow, but once we started working it came back.

3. Which of your design decisions stemmed from speaking to your client(s)or a proxy (e.g. your peers, stakeholders, potential users)? For those thatwere not, why, and where did they come from?

   Most of the module break up comes from talking to our client becuase they helped us focus on their vision for the project but making the inputs and specific variables were all done independently.

4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?

   For now no real document had to be changed becuase the structure for this assignment was thought of before through the many client meets. This allowed for a strong structure.

5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

   With unlimited resources the ability to capture better imaging with the kinect would allow for a faster and more precise human detection algorithm. Maybe also being able to better maximize the human detection to better fit a humaniod shape.

6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)

   Other design implications would just involve taking a different approach to creating the algorithm. The issue with for example a solution that does not use hue or skin color is limiting our ability to full captalize on the fact that the sensor picks up RGB as well.