CS 491/2 Senior Design Project

Low-Level Design Report

DepthCube

Students:
Alican Büyükçakır
Serkan Demirci
Semih Günel
Burak Savlu

Supervisor:
Prof. Uğur Güdükbay

Jury:
Assoc. Prof. Selim Aksoy
Prof. Özgür Ulusoy

Innovation Expert:
Armağan Yavuz (TaleWorlds)

February 21, 2017

Contents

1 Introduction **2**

1.1 Object Design Trade-offs 3

1.2 Interface Documentation Guidelines 4

1.3 Engineering Standards 4

1.4 Definitions, Acronyms and Abbreviations 5

2 Packages **6**

2.1 Server 6

2.1.1 Network Manager 6

2.1.2 Reconstruction Manager 6

2.1.3 Localization Manager 7

2.2 Client 9

2.2.1 UI Manager 9

2.2.2 Client Network Manager 9

3 Class Interfaces **10**

3.1 Network Manager 10

3.2 Reconstruction Manager 11

3.3 Localization Manager 12

3.4 UI Manager 13

3.5 Client Network Manager 14

3.6 Database 15

4 Third-Party Software **16**

5 Glossary **18**

6 References **20**

1 Introduction

DepthCube is a mobile application that aims to extend mobile devices' environmental understanding for augmented reality (AR) purposes. DepthCube takes your everyday environment and turns it into a virtual playground which you see through mobile device's camera. We want to create an environment where virtual objects interact with the real formation of your environment. In this sense, players will interact with the applications using their mobile devices as gamepads.

In the core application, DepthCube will provide applications with the precise location of the device relative to the room with the fine details about formation of the environment. The problem with the current state of the AR applications is the lack of computational power on the mobile devices, combined with the difficulty of providing real-time environmental formation. We want to tackle these obstacles using state-of-the-art computer vision algorithms, minimizing the need of an additional hardware.

Our purpose is to provide the developers with a reliable reconstruction and localization pipeline to be used in mobile devices. There stands many great ideas to be built on the knowledge of precise device location in indoor scenes. Although present advancements in the literature enables to build a such a system, lack of reliable software creates a big gap which is waiting to be fulfilled.

The lack of competitiveness in the field is for two-fold: Many competitors are more interested in using third party devices like depth cameras and expensive Google Tango devices, which enables system to make construction in the same device. easily . Yet many companies are still interested in building relatively large-scenes which creates storage and memory problems which seems impenetrable.

Our approach is to exclusively dealing with small scenes which makes the problem feasible. We are also willing to pass the computational burden into the server, so that we can be also battery efficient and we won't worry about many constraints posed by mobile devices. A finely optimized sync between server and the mobile device will be our primary concern.

1.1 Object Design Trade-offs

Availability vs. Cost:
With Depth Cube, we attempt to create a low-cost alternative to AR systems with similar robustness. The problem of common implementations stand as their high-cost, which uses expensive hardwares like depth cameras and high-end Nvidia GPUs which are placed in small hand-held devices. Our main approach is cutting these costs by preparing a finely-tuned pipeline, working on powerful servers. Thus, we expect a dramatic decrease in cost and not much loss in robustness compared to current systems. Unlike most advanced AR systems, since the API can function just by using what is present in a regular mobile device, its very low cost becomes one of its greatest strengths against its competitors. To ensure low-cost for different OS users.

Precision vs. Frame Rate:
We intend to produce the backbone of our program as bug-free as possible through multiple testing/debugging sessions. With DepthCube we need to do a lot of real-time operations both during game-play and environment recording. Like all real-time operations we expect some inaccuracies in the readings in these stages. To counter this, we will also execute these operations with better precision in a server and update periodically.

One problem with this approach is finding a good balance between precision and frame rate provided during localization process.Although, we already provide an offline reconstruction phase, whose duration is more-or-less fixed, the localization phase needs proper adjustment of several parameters to serve a healthy localization. We plan to rely on localization accuracy rather than high frame rate. This design decision also stem from the inevitable communication overhead between server and client. Although this communication -rather than relying of expensive third-party sensors- is the unique part of our approach, this also forces us to compromise between frame rate and precision.

Space and Time Complexity vs. Accuracy:
The more details that are desired in the 3d construction of the scene, the more feature points that needs to be extracted from the recorded frames. Therefore, if we want the 3d model to be more accurate and realistic, we need to store more information about the scene and do more computation as well. On the other hand, if we want the feature extraction to be fast, we shall not have as many feature points as we desired due to the limited time; and this causes our point clouds to be less accurate than they can be.

1.2 Interface Documentation Guidelines

The interface documentation guideline will be of the form of a table that is shown as following:

Package	Name of the package the class resides in.
Class Name	Name of the class.
Description	Broad description of the class.
Attributes	Class attributes.
Methods	Definition of methods present in the class. This will consist of describing the signature of the method (method name, arguments and their type, return type etc.), and verbose explanation of what functionality it has.

Table 1: Sample Table for Interface Documentation

1.3 Engineering Standards

UML: Unified Modeling Language was used to reflect the design of the system.[1]

IEEE: IEEE has a software life cycle development standard with code 1074.1-1995 named Guide for Developing Software Life Cycle Processes; which we are planning to follow. [2].

ANSI: The ISO 9001:2015 standard, which is applicable to any organization regardless of its size, was adopted to provide quality management [3]

[1]Unified Modeling Language User Guide (Second Edition) - Grady Booch

[2]IEEE Guide for Developing Software Life Cycle Processes - `https://standards.ieee.org/findstds/standard/1074.1-1995.html`

[3]ANSI - `http://webstore.ansi.org/RecordDetail.aspx?sku=ISO+9001%3a2015`

1.4 Definitions, Acronyms and Abbreviations

API: Application Programming Interface, is a set of subroutine definitions, protocols, and tools for building application software [4]

AR: Augmented Reality, is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data [5]

GUI: Graphical User Interface is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. [6]

OS: Operating System is system software that manages computer hardware and software resources and provides common services for computer programs.[7]

DB: Database is a collection of information that is organized so that it can easily be accessed, managed, and updated.[8]

[4]API - `https://en.wikipedia.org/wiki/Application_programming_interface`
[5]AR - `https://en.wikipedia.org/wiki/Augmented_reality`
[6]GUI - `http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html`
[7]OS - `https://en.wikipedia.org/wiki/Operating_system`
[8]DB - `http://searchsqlserver.techtarget.com/definition/database`

2 Packages

2.1 Server

2.1.1 Network Manager

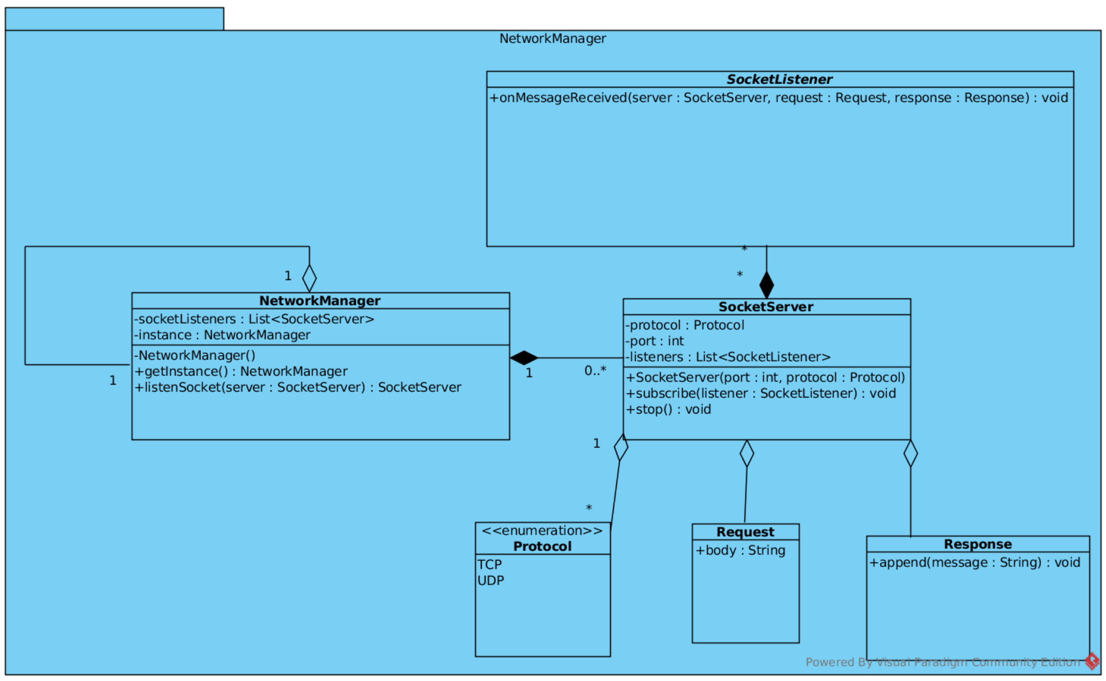

NetworkManager
SocketListener
+onMessageReceived(server : SocketServer, request : Request, response : Response) : void
NetworkManager
-socketListeners : List<SocketServer>
-instance : NetworkManager
-NetworkManager()
+getInstance() : NetworkManager
+listenSocket(server : SocketServer) : SocketServer
SocketServer
-protocol : Protocol
-port : int
-listeners : List<SocketListener>
+SocketServer(port : int, protocol : Protocol)
+subscribe(listener : SocketListener) : void
+stop() : void
<<enumeration>>
Protocol
TCP
UDP
Request
+body : String
Response
+append(message : String) : void
1
1
1
0..*
1
*
*
*
Powered By Visual Paradigm Community Edition

Figure 1: Network Manager Package.

Network Manager package is responsible for server-side network management. Opening up a new socket that will transfer data, subscription of the clients to the sockets and providing appropriate responses to clients are among the tasks of this package.

2.1.2 Reconstruction Manager

Reconstruction package is responsible for reconstruction of the scene that will be created from the video frames that are recorded. Computing and creating point cloud data by matching every next frame by the current, pose estimation and creation of the actual 3d scene by doing triangulization on the point cloud data are among the tasks of this package.

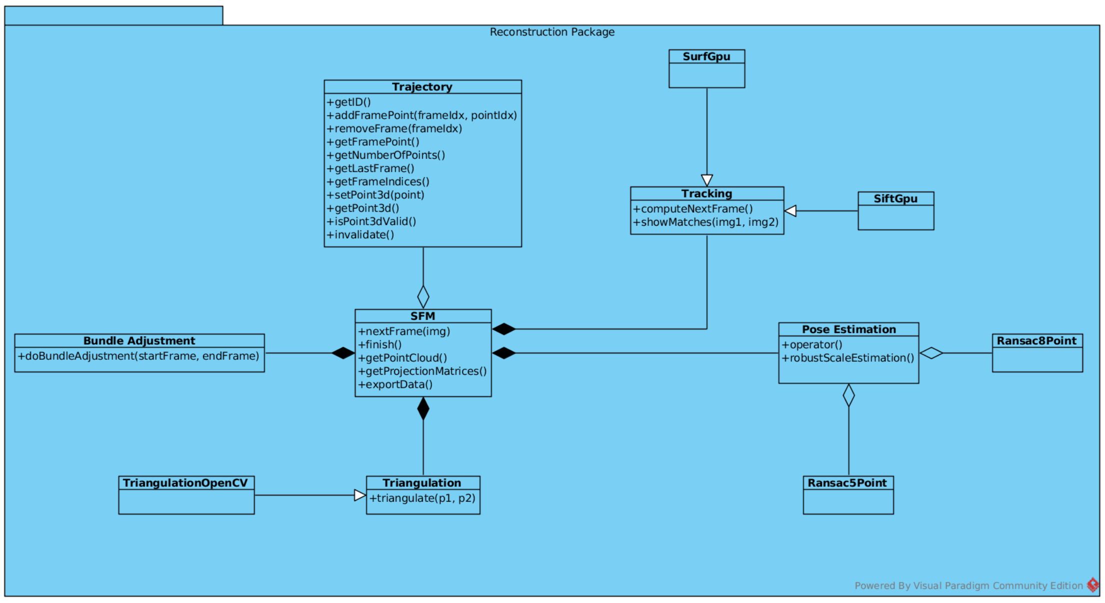


Figure 2: Reconstruction package

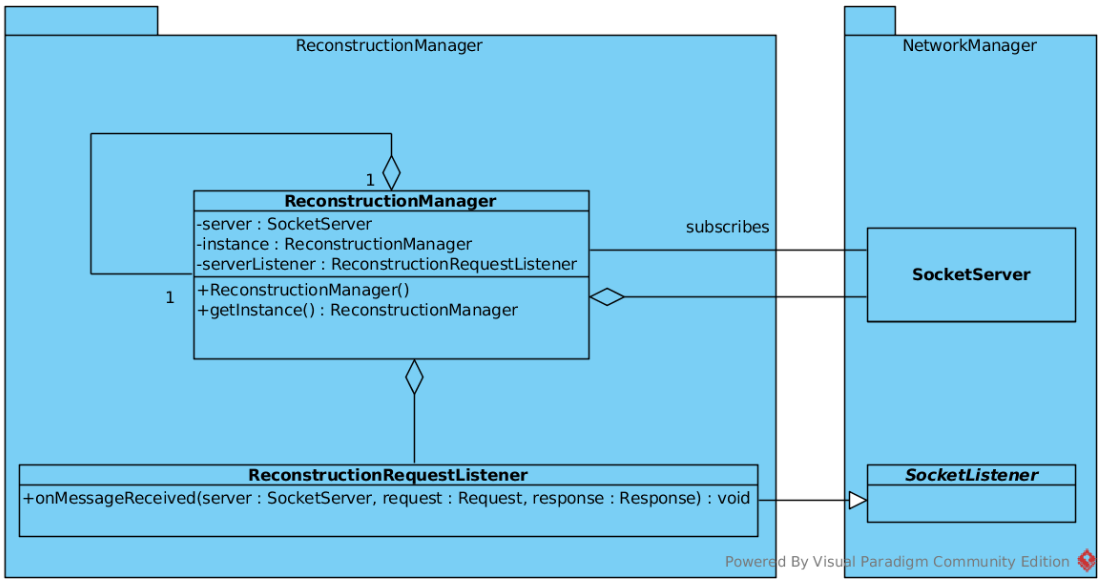

ReconstructionManager
NetworkManager
1
ReconstructionManager
-server : SocketServer
-instance : ReconstructionManager
-serverListener : ReconstructionRequestListener
+ReconstructionManager()
+getInstance() : ReconstructionManager
1
subscribes
SocketServer
ReconstructionRequestListener
+onMessageReceived(server : SocketServer, request : Request, response : Response) : void
SocketListener
Powered By Visual Paradigm Community Edition

Figure 3: Reconstruction Manager with its relation to Network Manager

2.1.3 Localization Manager

Localization package is responsible for localization (i.e. finding the exact location of the device in the scene). Extracting features from the current frame and matching features with the scene's real feature data to localize the device is the task of this package.

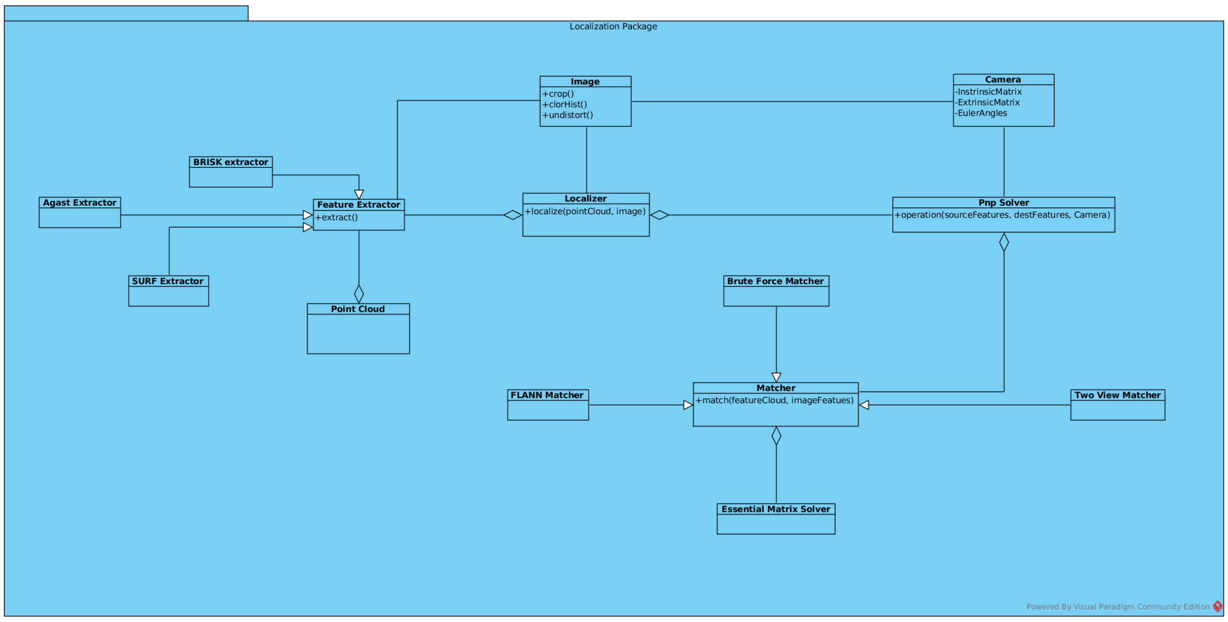


Figure 4: Localization package

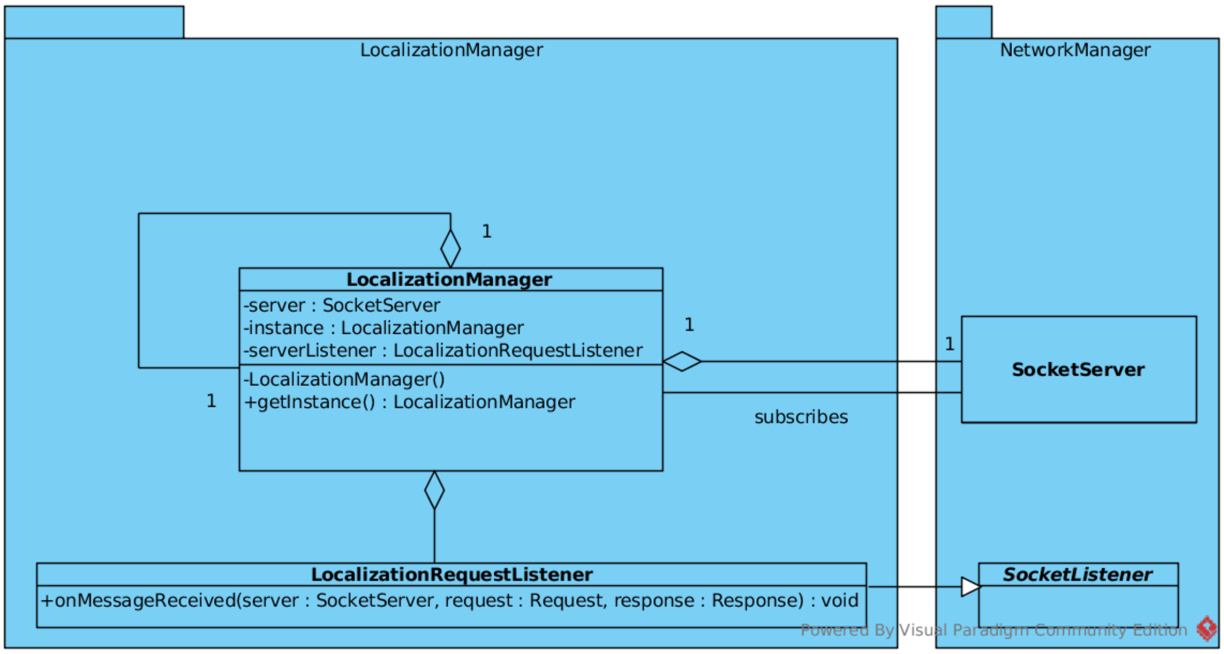

LocalizationManager
NetworkManager
1
LocalizationManager
-server : SocketServer
-instance : LocalizationManager
-serverListener : LocalizationRequestListener
-LocalizationManager()
+getInstance() : LocalizationManager
1
1
1
SocketServer
subscribes
LocalizationRequestListener
+onMessageReceived(server : SocketServer, request : Request, response : Response) : void
SocketListener
Powered By Visual Paradigm Community Edition

Figure 5: Localization Manager with its relation to Network Manager

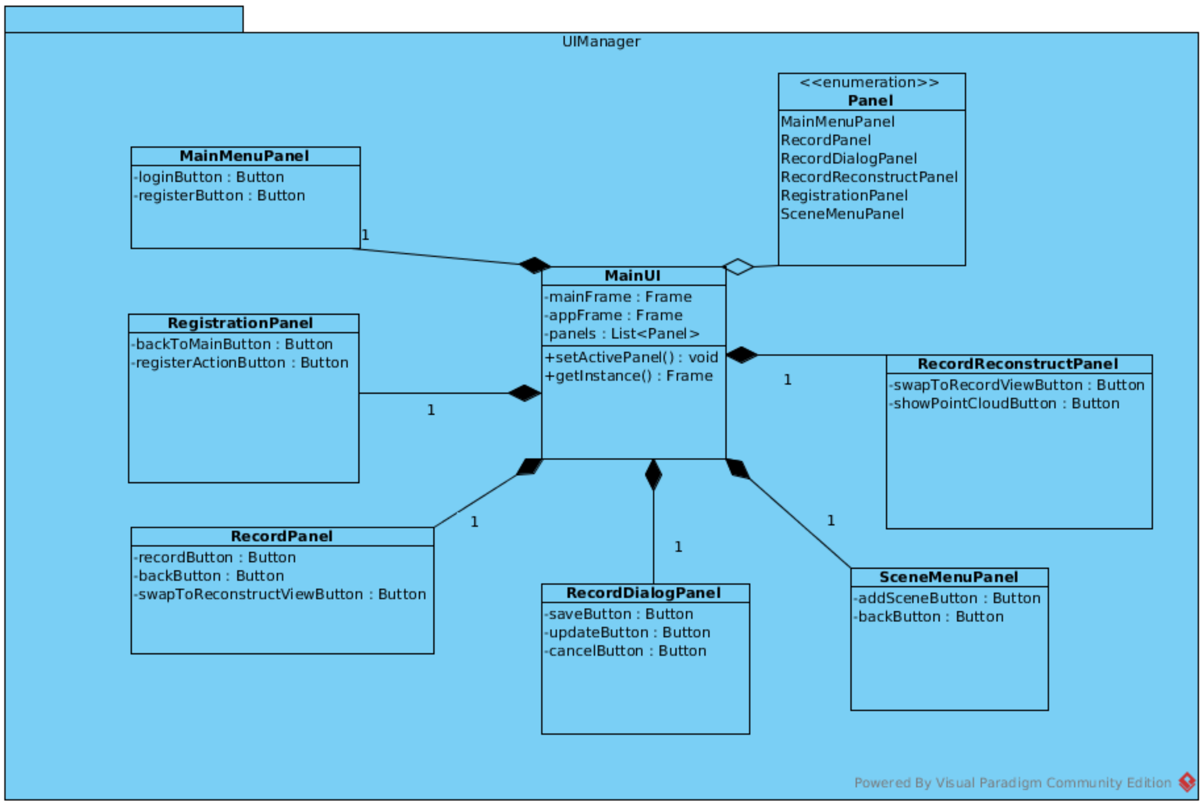


Figure 6: UIManager package

2.2 Client

2.2.1 UI Manager

UI Manager package is responsible for the graphical component of the application. Managing the user interface, switching between different types of panels according to the user input are among the tasks of this package.

2.2.2 Client Network Manager

Client Network Manager package is the package Network Manager's (which is for server-side network operations) counterpart, which is for the client side network operations. Sending request for reconstruction and localization to server and receiving the scene-related data as well as operations like logging in/out are among the duties of this package.

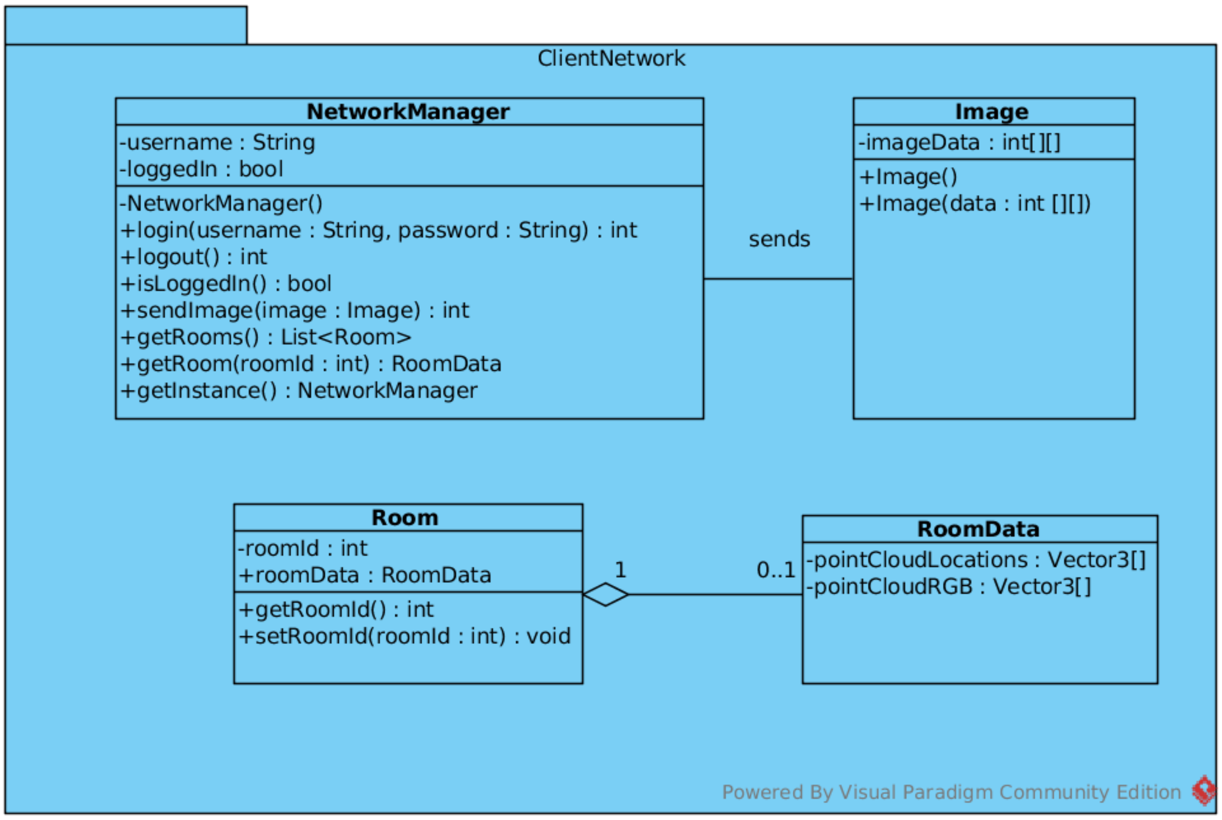

ClientNetwork
NetworkManager
-username : String
-loggedIn : bool
-NetworkManager()
+login(username : String, password : String) : int
+logout() : int
+isLoggedIn() : bool
+sendImage(image : Image) : int
+getRooms() : List<Room>
+getRoom(roomId : int) : RoomData
+getInstance() : NetworkManager
sends
Image
-imageData : int[][]
+Image()
+Image(data : int [][])
Room
-roomId : int
+roomData : RoomData
+getRoomId() : int
+setRoomId(roomid : int) : void
1
0..1
RoomData
-pointCloudLocations : Vector3[]
-pointCloudRGB : Vector3[]
Powered By Visual Paradigm Community Edition

Figure 7: Client Network Manager package

3 Class Interfaces

3.1 Network Manager

The following are the class interfaces for the most significant classes in the Network Manager package:

Package	Network Manager
Class Name	**NetworkManager**
Description	Main class that is singleton and listens to sockets that are given to it as SocketServers.
Attributes	**List<SocketServer>** socketListeners **NetworkManager** instance.
Methods	**getInstance()** : NetworkManager **listenSocket(SocketServer s)** : SocketServer

Table 2: NetworkManager Class

Package	Network Manager
Class Name	**SocketServer**
Description	Encapsulation of a socket that can be subscribed by the clients and can be listened by multiple clients.
Attributes	**Protocol** pr **int** port **List<SocketListener>** listeners
Methods	**subscribe(SocketListener sl)** : void **stop()** : void

Table 3: SocketServer Class

3.2 Reconstruction Manager

The following are the class interfaces for the most significant classes in the Reconstruction Manager package:

Package	Reconstruction Manager.
Class Name	**SFM**
Description	Main class that is responsible for creating the 3d structure of the scene by tracking frames, estimating pose, drawing trajectory and doing triangulation on a given point cloud.
Attributes	**PointCloud** pc **Frame** current **int[][]** projectionMatrix
Methods	**nextFrame(Image img)** : Frame **finish()** : bool **getPointCloud()** : PointCloud **getProjectionMatrices()** : int[][] **exportData()** : bool

Table 4: SFM Class

Package	Reconstruction Manager.
Class Name	**Tracking**
Description	Class that tracks the current and the next frame; matching their SIFT or SURF features.
Attributes	**Frame** current **Frame** next
Methods	**computeNextFrame(Image img)** : Frame **showMatches(Image i1, Image i2)** : List<Point>

Table 5: Tracking Class

3.3 Localization Manager

Package	Reconstruction Manager.
Class Name	**Localizer**
Description	Main class that localizes the recording device's camera.
Attributes	**FeatureExtractor** featureExtractor **PnPSolver** pnpSolver
Methods	**localize(PointCloud pc, Image img)** : int[][]

Table 6: Localizer Class

Package	Reconstruction Manager.
Class Name	**PnPSolver**
Description	Class that solves Perspective-n-Point problem that estimates the pose of the camera given the intrinsic and extrinsic values of the camera and feature points of a frame.
Attributes	**Matcher** matcher
Methods	**operation(FeatureList sourceFeatures, FeatureList destinationFeatures, Camera c)** : int[][]

Table 7: PnPSolver Class

Package	Reconstruction Manager.
Class Name	**FeatureExtractor**
Description	Class that extracts features from given images. It can use different descriptors such as SURF, Agast and Brisk.
Attributes	**SurfExtractor** surf **AgastExtractor** agast **BriskExtractor** brisk
Methods	**extractFeatures(Image img)** : List <Feature>

Table 8: FeatureExtractor Class

3.4 UI Manager

Package	UI Manager.
Class Name	**MainUI**
Description	The Main UI window that is singleton. This window's content (i.e. panels inside it) will change to switch between views.
Attributes	**Frame** mainFrame **Frame** appFrame **List<Panel>** panels **«Enumeration»** Panel
Methods	**setActivePanel(Panel p)** : void **getInstance()** : Frame

Table 9: MainUI Class

As the rest of the classes are consisting of panels which are structurally similar to MainMenuPanel; tables of class interfaces for the rest of the panels are omitted.

Package	UI Manager.
Class Name	**MainMenuPanel**
Description	Main menu panel that has login and register buttons. It is the first screen that a user will encounter when the application starts. Additionally, it has the DepthCube logo and text entries for entering a username and its corresponding password.
Attributes	**Button** loginButton **Button** registerButton
Methods	

Table 10: MainMenuPanel Class

3.5 Client Network Manager

Package	Client Network Manager.
Class Name	**NetworkManager**
Description	NetworkManager class of the client-side is responsible for login and logout operations, sending images to server for server's operations, and receiving the room (scene) information from the server.
Attributes	**String** username **boolean** loggedIn
Methods	**login(String username, String password)** : int **logout()** : int **isLoggedIn()** : boolean **sendImage(Image img)** : int **getRooms()** : List<Room> **getRoom(int roomId)** : RoomData **getInstance()** : NetworkManager

Table 11: NetworkManager Class (Client Network Manager)

Package	Client Network Manager.
Class Name	**Room**
Description	Room is the class that is an abstraction of a scenery whose 3D-structure is generated by re-construction algorithms. It uses an instance of a RoomData class to keep record of the point cloud data of the scenery.
Attributes	**int** roomId **RoomData** roomData
Methods	**getRoomId()** : int **setRoomId(int roomId)** : void

Table 12: Room Class

3.6 Database

MySQL 5.6 server will be used to store user related data. Password of the user will be encryped using a hashing function along with randomly generated. Hashing algorithm will be SHA-256 algorithm. Images and additional room data will be stored in the file system inside "user_id/room_id" folder.

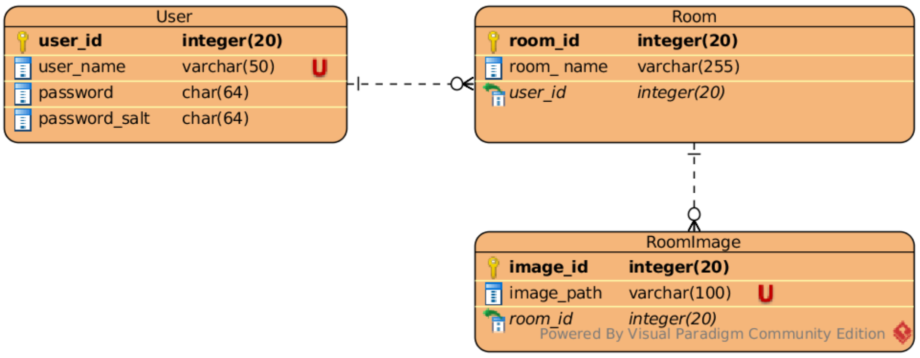


Figure 8: ER Diagram for database constructed in the server.

```
CREATE TABLE Room (
  room_id      int(20) NOT NULL AUTO_INCREMENT,
  room_name    varchar(255) NOT NULL,
  user_id      int(20) NOT NULL,
  PRIMARY KEY (room_id));
CREATE TABLE RoomImage (
  image_id   int(20) NOT NULL AUTO_INCREMENT,
  image_path varchar(100) NOT NULL UNIQUE,
  room_id    int(20) NOT NULL,
  PRIMARY KEY (image_id));
CREATE TABLE `User` (
  user_id       int(20) NOT NULL AUTO_INCREMENT,
  user_name     varchar(50) NOT NULL UNIQUE,
  password      char(64) NOT NULL,
  password_salt char(64) NOT NULL,
  PRIMARY KEY (user_id));
```

Figure 9: SQL database generation code

4 Third-Party Software

We will employ different third-party open source software for critical components of DepthCube. Since the problem of structure from motion is considered as solved, there are many packages and libraries, with highly changing quality and speed, which can provide this important service to us without involving in highly complex and scientific software writing. Especially for structure form motion algorithms, we have several options to employ.

- **Bundler[8]:** Bundler is one of the first open source large scale structure from motion algorithm developed in Cornell University by Noah Snavely. Although still popular and highly used, software is not maintained is quite old. Bundler does not always support new hardware. Although we had experiments with Bundler, we decided to search for other options.

- **VisualSFM[7]:** One other open source structure-from-motion algorithm package is VisualSFM developed by Changchang Wu. Although he stopped maintaining the package, VisualSFM provides and easy-to-use GUI and it supports incremental building of the environment. It also supports command line actions, which is essential for our server setup. One other nice thing about VisualSFM is it has a healthy community, which can provide information when necessary. We are currently using VisualSFM since it is well established and its behaviour -and potential weaknessesare well known and agreed to be still one of the finest SFM implementations.

- **OpenCV[11]:** OpenCV is one of the most popular open-source computer vision libraries. We will use OpenCV both in client and server side. In the client side OpenCV will provide tracking and feature-extraction. In the server side, OpenCV will facilitate feature matching and other PNP and RANSAC algorithms. Also OpenCV's Python support enables fast prototyping for new ideas.
- **Colmap[9]:** Colmap is one of the latest structure from motion algorithms, developed in 2016 by Johannes L. Schonberger from University of North Carolina Chapell Hill. Colmap supports new optimizations in standard structure-from-motion algorithm. However it does not provide necessary features like dense reconstruction naively. Also its user community is relatively small, therefore finding software support is relatively hard. Yet Colmap has relatively easy-to-learn software and good documentation.
- **Theia Vision Library[10]:** Theia is also one of the youngest offline structure from motion algorithms. It is developed by Chris Sweeney in UC Santa Barbara, during 2015-2016. Theia is designed to provide easy-to-use interface for multi-view geometry algorithms, although it is built with structure from motion in mind, it can support many other software development. It is designed to minimize the number of external dependencies.

5 Glossary

6-DOF: 6 Degrees of Freedom. Refers to the freedom of movement of a rigid body in three-dimensional space. Specifically, the body is free to change position as forward/backward (surge), up/down (heave), left/right (sway) translation in three perpendicular axes, combined with changes in orientation through rotation about three perpendicular axes.[9]

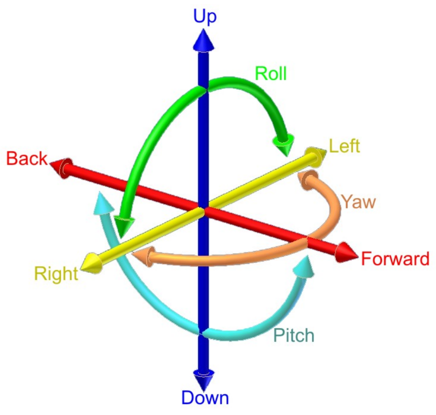
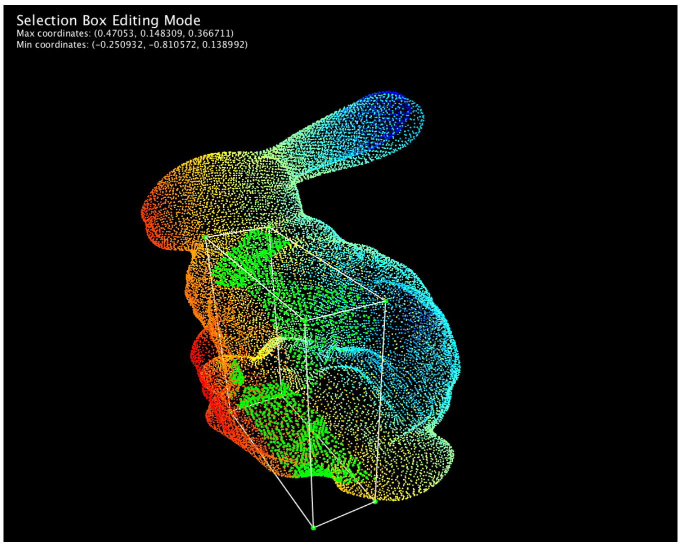


Figure 10: Six Degrees of Freedom.

Point Cloud: A point cloud is a set of data points in some coordinate system.



```
Selection Box Editing Mode
Max coordinates: (0.47053, 0.148309, 0.366711)
Min coordinates: (-0.250932, -0.810572, 0.138992)
```

Figure 11: A sample colorized point cloud [5].

[9]Six Degrees of Freedom - `http://xinreality.com/wiki/Degrees_of_freedom`

Mesh: A mesh is a collection of vertices, edges, and faces that describe the shape of a 3D object.

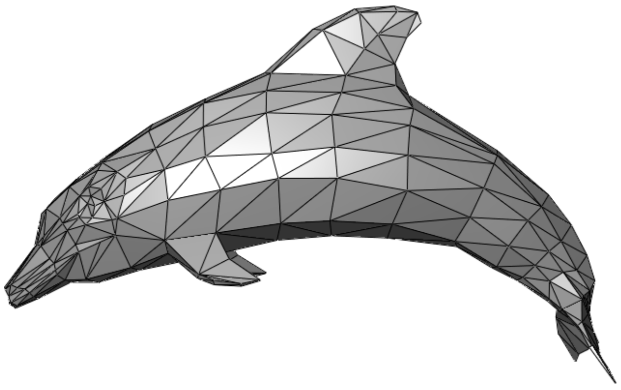


Figure 12: A triangle mesh representing a dolphin.

FAST: Features from Accelerated Segment Test, is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision tasks.[1]

BRIEF: Binary Robust Independent Elementary Features. A feature descriptor that uses binary strings as an efficient feature point descriptor.[2]

SFM: Structure From Motion, is a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences that may be coupled with local motion signals. [10]

SLAM: Simultaneous Localization and Mapping, is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it. [11]

[10]Structure From Motion - `https://en.wikipedia.org/wiki/Structure_from_motion`
[11]SLAM - `https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping`

6 References

References

[1] Rosten, Edward, and Tom Drummond. *Machine learning for high-speed corner detection.* European conference on computer vision. Springer Berlin Heidelberg, 2006. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Calonder, Michael, et al. *Brief: Binary robust independent elementary features.* European conference on computer vision. Springer Berlin Heidelberg, 2010.

[3] CS491 Senior Design Project I - Guidelines: `http://www.cs.bilkent.edu.tr/CS491-2/CS491.html`

[4] Mur-Artal, Raúl, and Juan D. Tardós. *Probabilistic semi-dense mapping from highly accurate feature-based monocular slam.* Proceedings of Robotics: Science and Systems Rome, Italy 1 (2015).

[5] 3D Point Cloud Editor `http://paradise.caltech.edu/~yli/software/pceditor.html`

[6] UJIIndoorLoc Data Set `https://archive.ics.uci.edu/ml/datasets/UJIIndoorLoc`

[7] VisualSFM : A Visual Structure from Motion System `http://ccwu.me/vsfm/`

[8] Bundler: Structure from Motion (SfM) for Unordered Image Collections `https://www.cs.cornell.edu/~snavely/bundler/`

[9] COLMAP - Structure-From-Motion and Multi-View Stereo `http://people.inf.ethz.ch/jschoenb/colmap/`

[10] Theia Vision Library `http://www.theia-sfm.org/`

[11] OpenCV - Open Source Computer Vision Library. `http://opencv.org/`