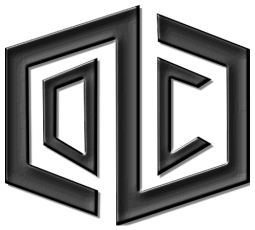
CS 491 Senior Design Project

Analysis 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)

November 6, 2016

Contents

1 Introduction **3**

2 Current Systems **4**

3 Proposed System **5**

3.1 Overview 5

3.2 Requirements 6

3.2.1 Functional Requirements 6

3.2.2 Non-Functional Requirements 7

3.2.3 Pseudo-Requirements 7

3.3 System Models 8

3.3.1 Use Case Model 8

3.3.2 Object and Class Model 12

3.3.3 Dynamic Models 13

3.3.4 User Interface and Screen Mockups 17

4 References **21**

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.

Current Systems

There are some products that use similar technologies to the DepthCube. There are game consoles such as Kinect, or additional sensors like Structure Sensor use depth information to build 3D structure of the environment. Although, they can perform camera tracking and 3D model construction faster, these sensors have limited range for their depth measurements. They cannot construct 3D model of the large environments. Additionally, there are some AR frameworks that run for mobile devices. With these frameworks, developers can build AR applications that can interact with the environment without need of any external hardware. However, they cannot perform well due to low computational power of the target mobile devices. In addition, they use a lot of battery for their calculation.

The most similar technology with DepthCube is Project Tango, currently developed by Google. Project Tango wants to solve the problem of area awareness in real time, and their ultimate goal is the simultaneous localization and mapping. Tango devices include a powerful NVIDIA GPU, a depth sensor, and RGB camera with large field of view. All these to support real-time computations. However these which makes the Tango device relatively expensive, compared to main-stream mobile devices, and it is almost stands like specifically built for AR and VR purposes, rather than serve as a general mobile device. Tango device is heavy and thick compared to the main-stream phones and tablets.

Proposed System

3.1 Overview

DepthCube is an API that utilizes mobile devices' camera for information on real life environment in order to create a virtual playground for users. Our ultimate goal is to create a cheap and effective API that can enable many AR applications including but not limited to games. When we look at similar systems, we can see many innovations originated from the Kinect console, which provided developers with the depth sensor and skeletal tracking. With DepthCube, it is possible to do the same thing, but this time without need of any additional hardware. We believe this makes the product very appealing to customers not only due to its low price but also because it is very accessible for anyone with a mobile device.

To clarify how DepthCube functions, assume that we have a game that is already developed and ready to play for the users. Our exemplary new game will be very similar to Pong, but this time balls bounce on the real walls and objects, rather than on rectangular borders. Users will interact with the virtual ball using their mobile device as a virtual surface where virtual ball bounce on. Here is how the system will work:

- First user will scan her environment using the mobile device's camera. Simultaneously DepthCube will create a sparse representation of the environment, and show the 3D model on the mobile device.

- When scanning phase has been completed, DepthCube will create a dense representation of the room in our dedicated cloud servers. This construction might take several hours. At the end of this construction, DepthCube will create the environment in Unity model, which correctly represents the room.

- Then, Unity model will be transferred to the mobile device which runs the DepthCube Android application, which is built upon Unity Engine.

- Once Unity model is received and wireless connection between the mobile device and server is established, users can run games or applications that are built with DepthCube Unity Framework on their phones.

- During the game(or application) phase, using the constructed scene model from the previous step, location information will be available for the user.

Figure 3.1: DepthCube's understanding the room after scan phase. Each vertex corresponds to a 3D point.

- Since Unity engine has the fine details of the room with the precise location of the mobile device, Unity engine will draw the virtual objects in the correct location and size without disrupting the perspective.

3.2 Requirements

3.2.1 Functional Requirements

- Players should have their own unique username/password.
- DepthCube should not require anything other than an Android device to run.
- Games should be controlled using a mobile device(smartphone/tablet).
- There should be two types of users for the product: developers and players.
- Players should be able to experience the Augmented Reality, given an already developed and ready-to-play game.
- Players should be able to upload videos for 3D construction.
- DepthCube should process user videos to create 3D structure
- Players should be able to download games from the online library.
- Developers should be able to design and develop their own games using our API that provides the location and structural information about the environment.

3.2.2 Non-Functional Requirements

- Each player should have access to the last 5 3D structures they created.
- DepthCube should support Android 4.4 and above.
- A local and fast connection between DepthCube' server and mobile device is required.
- Mobile interface of the DepthCube application should be user-friendly, clean and simplistic. Users must not have any problems with navigating themselves inside the application.
- Games developed for DepthCube should have at least 30 FPS.
- The provided API for developers should be well-documented, precise and succinct.
- DepthCube should scan user video and create a 3D structure in less than 1 hour.
- DepthCube should support games developed using Unity3D version 5.0.0 and above.
- A system crash should not result in data loss

3.2.3 Pseudo-Requirements

- The server code should be written in C++.
- User informations (username, password and email) should be kept in a relational database (e.g. MySQL).
- The provided API should be developed on Unity3D.

3.3 System Models

3.3.1 Use Case Model

The following is the Use Case diagram that shows the functionalities of our application. Below the figure, use cases are further explained.

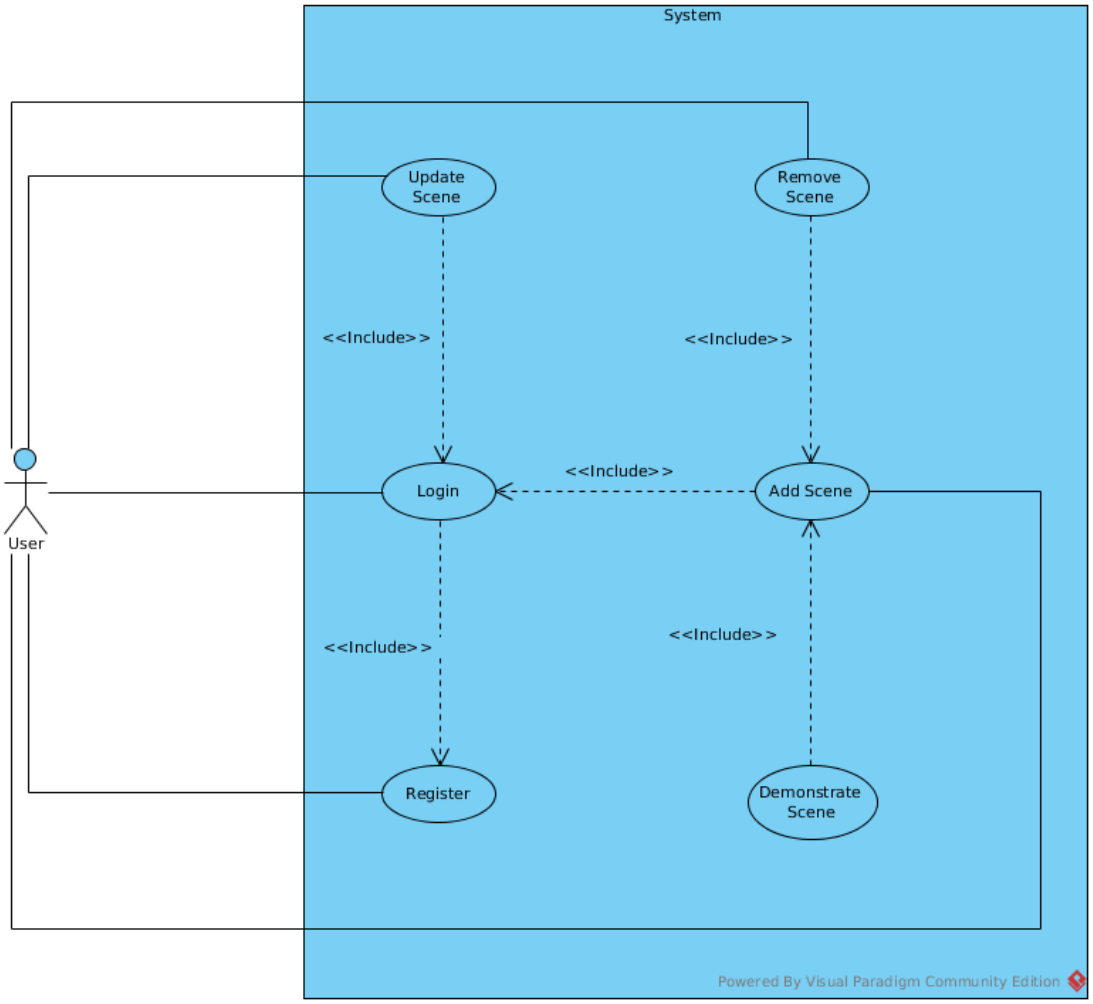


Figure 3.2: Use Case Diagram.

Firstly, user has to be registered to be able to log in. A logged in user can add a scene. This scene can be further modified (updated or deleted). Finally, user demonstrates a scene which has to be previously added to the user's scene list.

Use Case Name: Register
Participating Actors: User
Entry Condition: None.
Flow of Events:

1. User opens the main menu of the application.
2. User clicks on 'Register' button.
3. User fills the blanks with his credentials.
4. User clicks on 'Sign Up' button.

Exit Conditions:

1. User clicks on "Back to Main Menu" button.

Use Case Name: Login
Participating Actors: User
Entry Condition: Register
Flow of Events:

1. User opens the main menu of the application.
2. User clicks on 'Login' button.
3. User fills the blanks with his credentials.
4. User clicks on 'Login' button and logs in.

Exit Conditions:

1. 1. User clicks on "Back to Main Menu" button

Use Case Name: Add Scene
Participating Actors: User
Entry Condition: User clicks to 'My Scenes' button
Flow of Events:

1. User clicks 'Add Scene' button.

2. User starts recording his environment
   (a) User receives feedback about the construction of the scene
3. User stops recording the scene.
4. User waits for the scene construction to be completed.
5. User is shown the constructed scene.
6. User clicks 'Accept', 'Update' or 'Cancel' button to keep, update or discard the scene respectively.

Exit Conditions:

1. User clicks on "Back to Main Menu" button

Use Case Name: Remove Scene
Participating Actors: User
Entry Condition: User clicks to 'My Scenes' button
Flow of Events:

1. User chooses a scene to remove from the list of the scenes.
2. User clicks on 'Remove Scene' button.
3. User is given a confirmation prompt to finalize his answer.
4. User clicks on 'Yes' or 'No' depending on his decision.

Exit Conditions:

1. 1. User clicks on "Back to Main Menu" button

Use Case Name: Update Scene
Participating Actors: User
Entry Condition: User clicks on 'Update' at the end of 'Add Scene'
Flow of Events:

1. User chooses a scene to remove from the list of the scenes.

2. User clicks on 'Remove Scene' button.
3. User is given a confirmation prompt to finalize his answer.
4. User clicks on 'Yes' or 'No' depending on his decision.

Exit Conditions:

1. 1. User clicks on "Back to Main Menu" button

Use Case Name: Demonstrate Scene
Participating Actors: User
Entry Condition: User clicks on 'My Scenes' button
Flow of Events:

1. User chooses a scene.
2. User clicks on 'Demonstrate' button.
3. User is shown his location in the reconstructed scene.
   (a) The shown location of user in the demonstration is updated real-time.

Exit Conditions:

1. User clicks on "Back to Main Menu" button

3.3.2 Object and Class Model

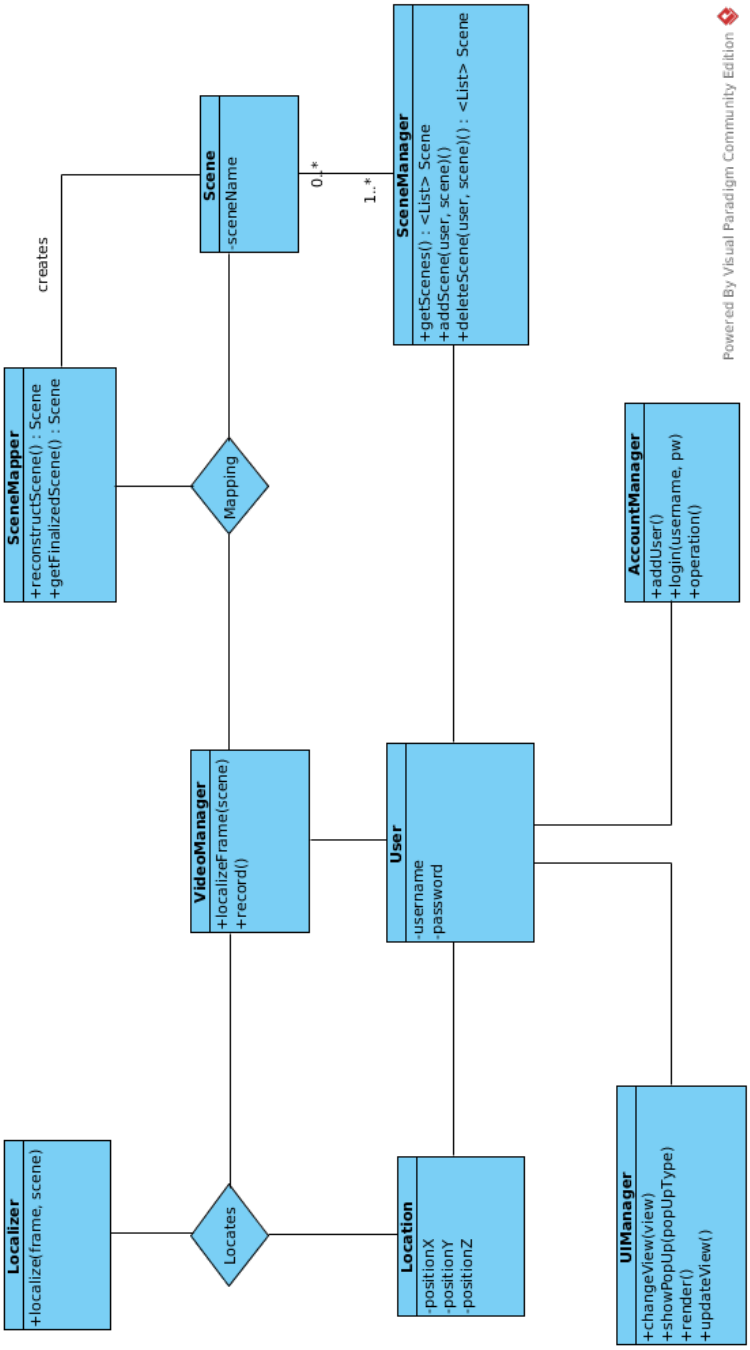


Figure 3.3: Class Diagram.

3.3.3 Dynamic Models

Sequence Diagrams

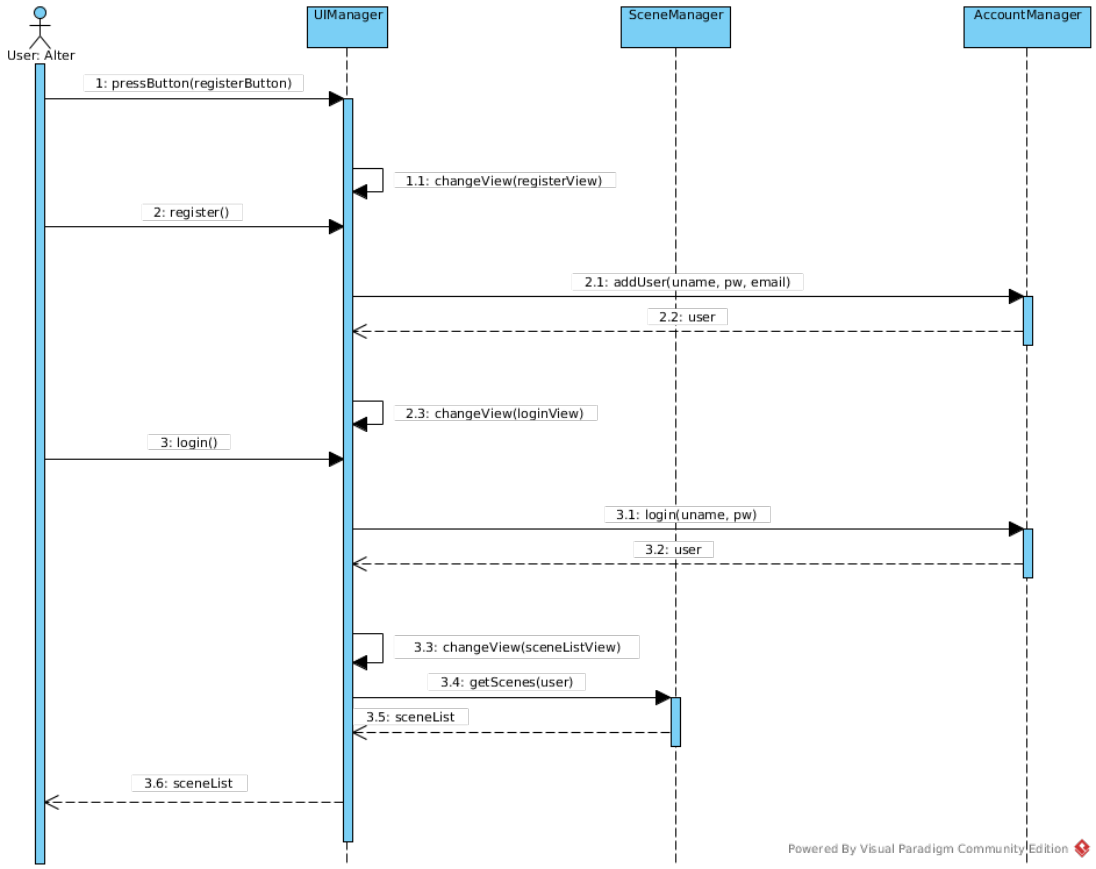


Figure 3.4: Sequence Diagram for Register and Login processes.

Scenario Name: Register and Login
Actors: User
Flow of Events:
This is Alter's first time opening the application. He first clicks on 'Register' button and fills in the required fields for the registration to be completed. Then, he presses 'Register' button again to submit his registration. After registration is completed, application returns back to main menu. Alter, happily, fills in his username and password and logs in to the system. He then encounters with the Scene List screen.

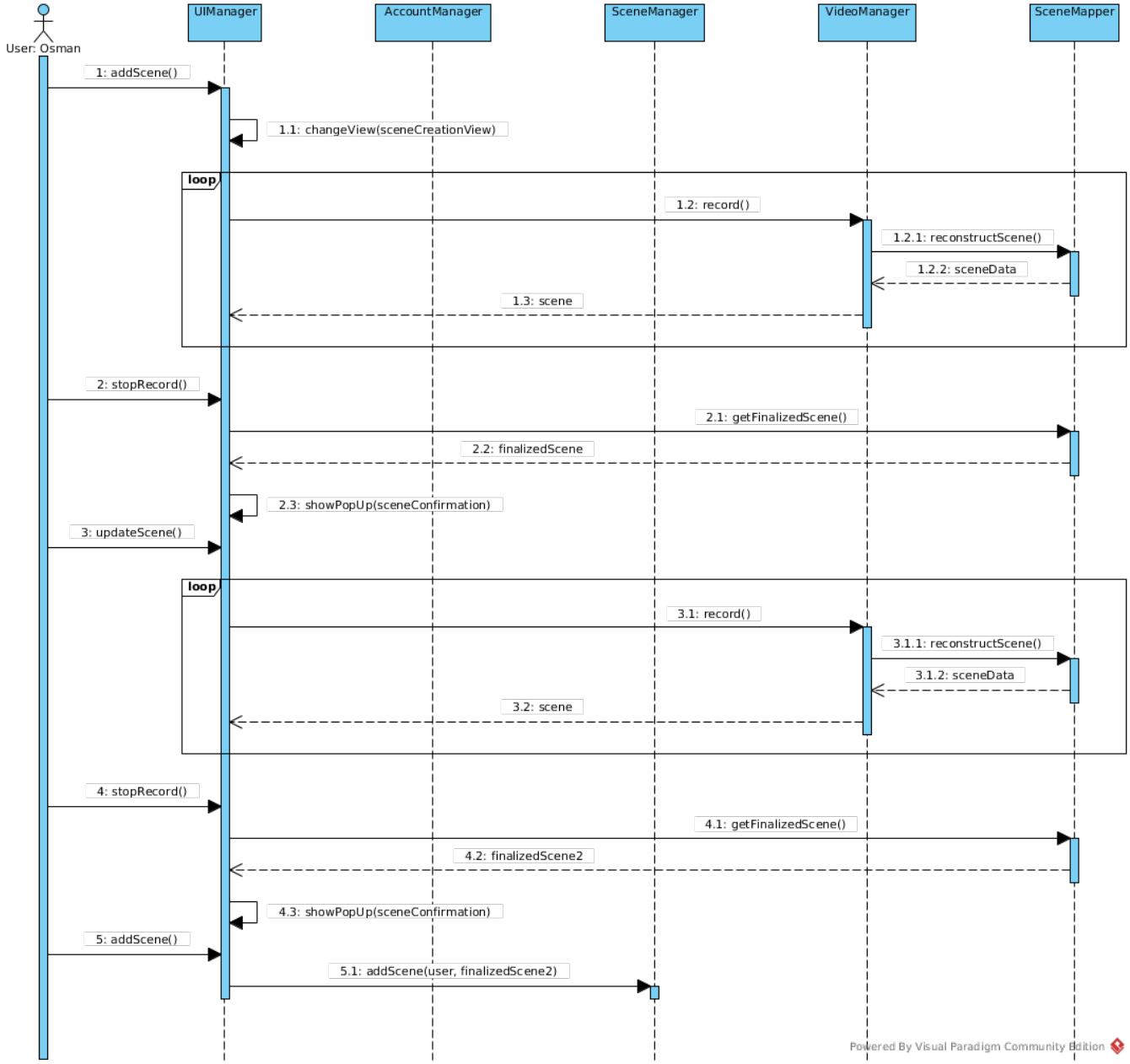


Figure 3.5: Sequence Diagram for Scene Creation.

Scenario Name: Scene Creation
Actors: User
Flow of Events:
Osman is already registered to the system and logged in to the application. He presses 'Add Scene' button and his device's camera starts recording his environment. Meanwhile, he can see the current state of the reconstructed scene in his device's screen. He walks around his environment and he sees that the reconstructed scene that is displayed gets better and

better as he shows more and more frames of the environment. Osman acknowledges that the reconstructed scene looks good enough and stops recording. He waits for some time so that the finalized version of the scene is constructed. He does not like the final version and wants to improve some parts of the reconstructed scene. He presses 'Update Scene' and stars recording with his device's camera again. At the end of this recording session, he clicks on 'Accept' to confirm the scene's reconstruction as satisfactory. The scene is saved in the scene list.

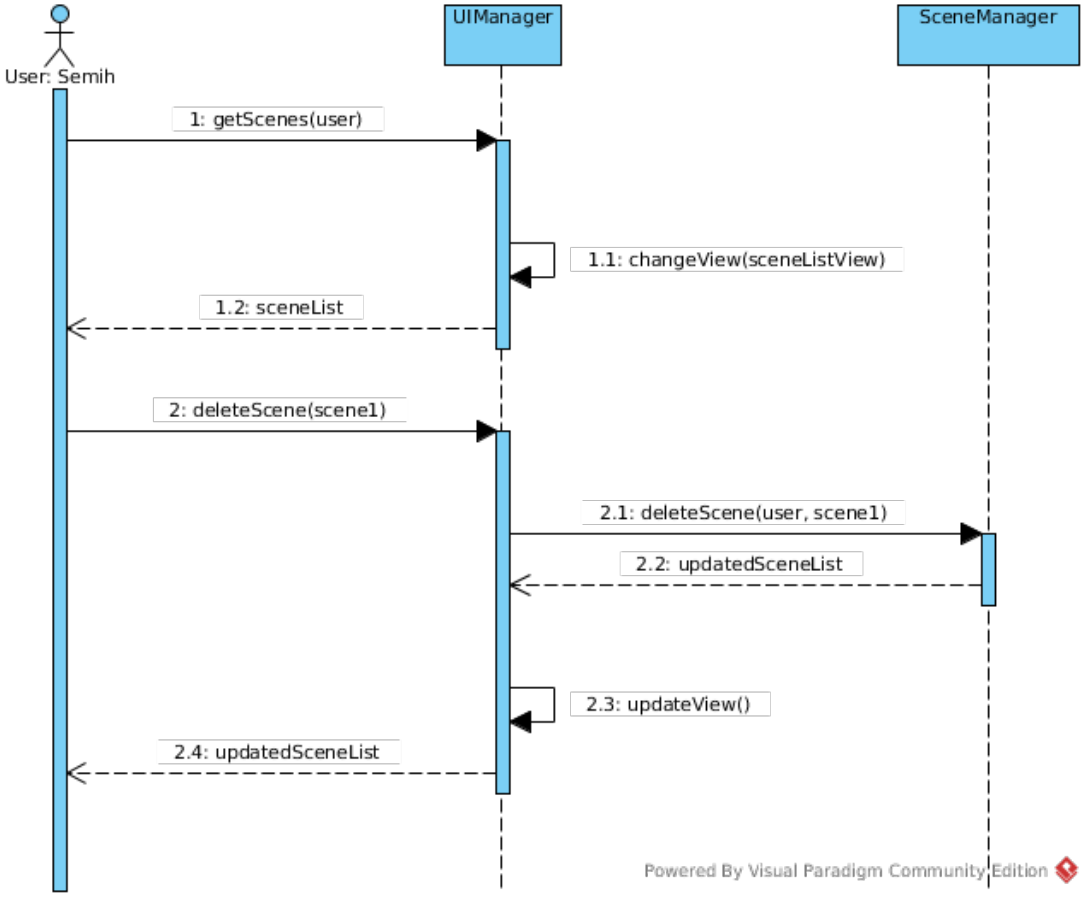


Figure 3.6: Sequence Diagram for Scene Deletion.

Scenario Name: Scene Deletion
Actors: User
Flow of Events:
Semih moved from his old house and wants to delete the scene data of his house from his scene list. He enters the scene list and selects the scene that he wishes to delete. Then he presses 'Remove' button. The scene is deleted from the scene list.

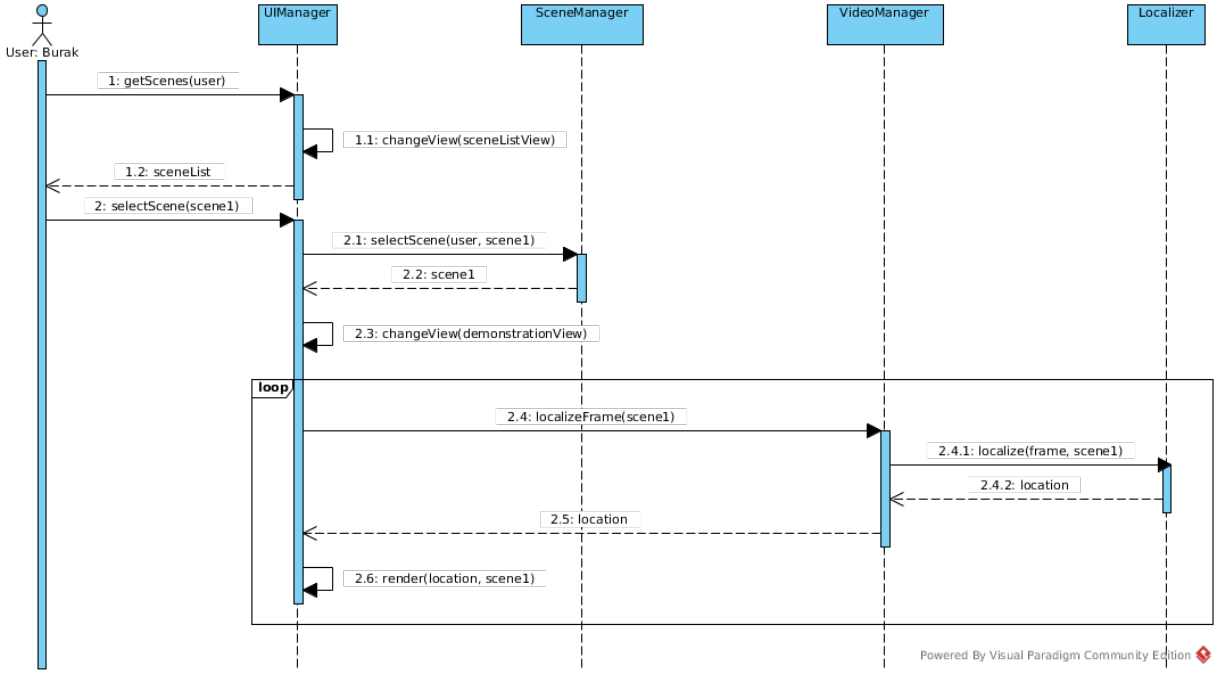


Figure 3.7: Sequence Diagram for Scene Demonstration.

Scenario Name: Scene Demonstration
Actors: User
Flow of Events:
Burak wants to show preciseness and amazingness DepthCube to his friends. He enters a room that he previously built a scene from. Then, he selects a scene from the scene list. He presses 'Demo' button to initiate augmented reality showcase. As he moves around his room, he sees on his phone's screen that his location is updated and precisely located in the 3D-model of his room.

3.3.4 User Interface and Screen Mockups

Figure 3.8: Main Menu Screen.

Figure 3.9: Registration Screen.

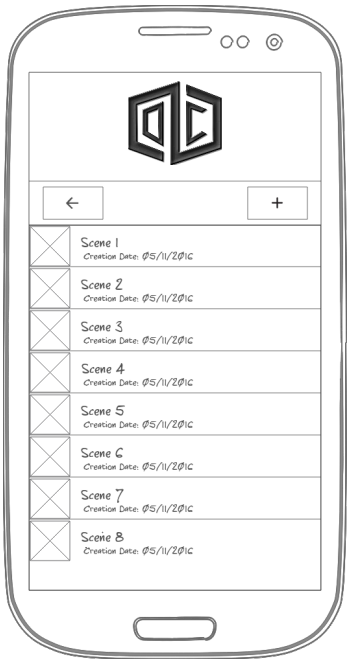


Figure 3.10: Scene List.

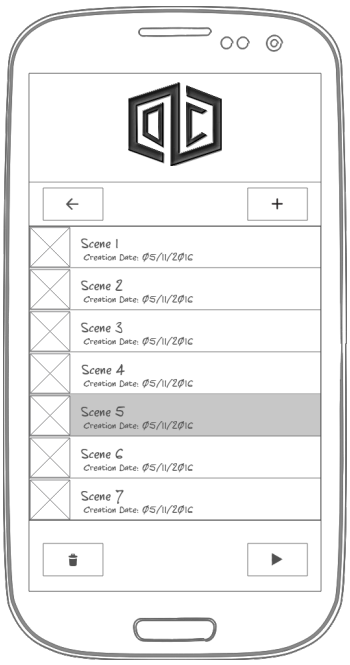


Figure 3.11: Scene List when a scene is selected.

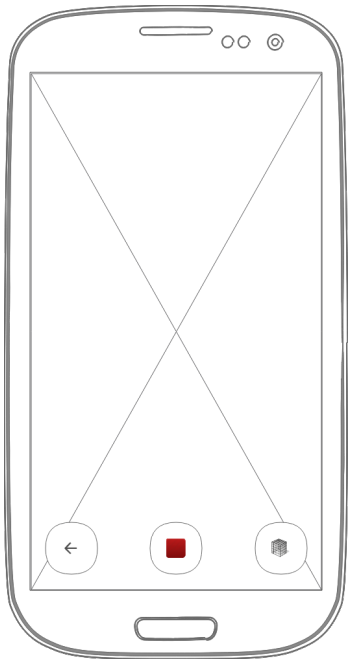


Figure 3.12: Record Reconstruction Screen.

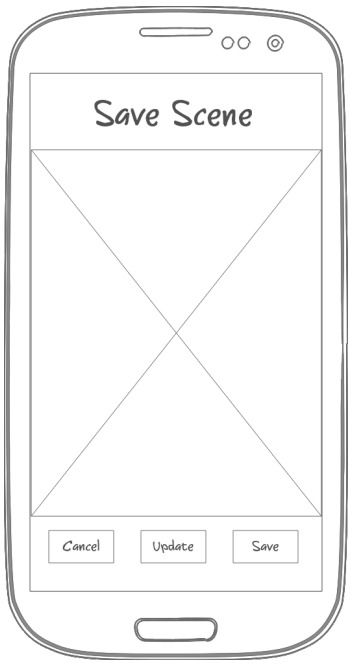


Figure 3.13: Record Dialog Screen.

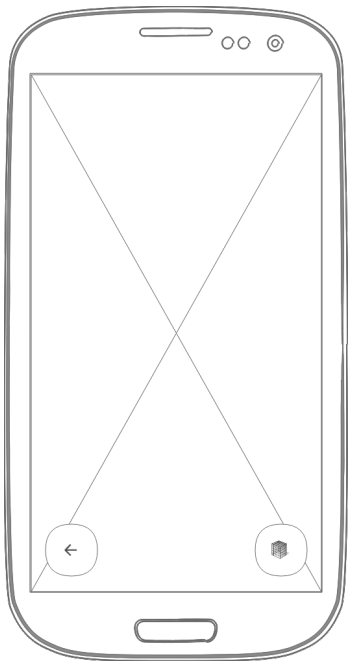


Figure 3.14: Demonstration Screen.

References

Project Tango:
http://get.google.com/tango/

Structure From Motion:
http://bit.ly/2dSxD7L

OpenCV Camera Motion Estimation:
http://docs.opencv.org/3.1.0/d5/dab/tutorial_sfm_trajectory_estimation.html

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

ORB-SLAM, structure from motion implementation for monocular cameras:
http://webdiis.unizar.es/~raulmur/MurMontielTardosTRO15.pdf

Kinect with Skeletal Tracking:
https://msdn.microsoft.com/en-us/library/jj131025.aspx

Kinect Sensor:
https://msdn.microsoft.com/en-us/library/hh438998.aspx

Kinect Sensor Hardware Specifications:
https://developer.microsoft.com/en-us/windows/kinect/hardware

Structure Sensor:
https://store.structure.io/store

CS 491 Senior Design Project

Specification 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)

October 10, 2016

Contents

1 Introduction **3**

1.1 Description 3

1.2 Similar Products / Technologies 4

1.3 Constraints 6

1.3.1 Technical Constraints 6

1.3.2 Social Constraints 6

1.3.3 Implementation Constraints 6

1.3.4 Sustainability Constraints 7

1.3.5 Economic Constraints 7

1.3.6 Security Constraints 7

1.3.7 Language Constraints 7

1.4 Professional and Ethical Issues 8

2 Requirements **9**

2.1 Functional Requirements 9

2.2 Non-Functional Requirements 9

3 References **11**

Introduction

1.1 Description

DepthCube is a portable game console that extends 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 interacting with the real formation of your environment. In this sense, players will interact with the game console using their mobile devices as gamepads.

In the core application, DepthCube will provide device/application with the precise location of the device relative to the room with the fine details about formation of the environment. We want to achieve this goal using state-of-the-art computer vision algorithms, with minimizing the need of an additional hardware. 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 this obstacle with the additional computing power provided by DepthCube and better algorithms.

Our ultimate goal is to create an API-like infrastructure, where many interesting ideas can be built upon. We can see many innovations originated from the Kinect console, which provided developers with the depth sensor and skeletal tracking. It is possible to do the same thing with the minimum cost, but this time on the mobile devices, with a new idea. Customers will be willing to buy DepthCube in order to extend their mobile devices capabilities on the environmental understanding and enable potential applications exploiting this area awareness.

To further clarify, assume that we have a game that is already developed and ready to play for the users developed using the API of DepthCube. Our exemplary new game will be very similar to Pong, but this time balls bounce on the real walls and objects, rather than on rectangular borders. Users will interact with the virtual ball using their mobile device as a virtual surface where virtual ball bounce on. Here is how the system will work:

- First user will scan her environment using the camera present on DepthCube. Simultaneously DepthCube will create a sparse representation of the environment, and show the 3D model on the mobile devices. It is sufficient to scan a room only once, when the console is newly acquired.
- When scanning phase has completed, DepthCube will create a dense representation of the room in the offline mode. This construction will take several hours. At the end DepthCube will create the environment in Unity model, which correctly represents the room.

Figure 1.1: DepthCube's understanding the room after scan phase. Each vertex corresponds to a 3D point.

- Then, Unity model will be transferred to the mobile device which runs the DepthCube Android application, which is built upon Unity Engine.
- Once Unity model is received and wireless connection between the mobile device and DepthCube is console is established, users can run the Pong game on their phones.
- During the game phase, wireless connection between DepthCube and device will provide the information of precise localization of mobile device.The information will be extracted from key-points, intensity information using already calculated dense 3D structure of the room.
- Since Unity engine has the fine details of the room with the precise location of the mobile device, Unity engine will draw the virtual objects in the correct location and size without disrupting the perspective.

1.2 Similar Products / Technologies

There are some products that use similar technologies to the DepthCube. There are game consoles such as Kinect, or additional sensors like Structure Sensor use depth information to build 3D structure of the environment. Although, they can perform camera tracking and 3D model construction faster, these sensors have limited range for their depth measurements. They cannot construct 3D model of the large environments. Moreover, their price is drastically more expensive than DepthCube's targeted price. Additionally, there are some AR frameworks that run for mobile devices. With these frameworks, developers can build AR applications that can interact with the environment without need of any external hardware. However, they cannot perform well due to low computational power of the target mobile devices. In addition, they use a lot of battery for their calculation.

The most similar technology with DepthCube is Project Tango, currently developed by Google. Project Tango wants to solve the problem of area awareness in real time, and their ultimate goal is the simultaneous localization and mapping. Tango devices include a powerful NVIDIA GPU, a depth sensor, and RGB camera with large field of view. All these to support real-time computations. However these which makes the Tango device relatively expensive, compared to main-stream mobile devices, and it is almost stands like specifically built for AR and VR purposes, rather than serve as a general mobile device. Tango device is heavy and thick compared to the main-stream phones and tablets.

1.3 Constraints

1.3.1 Technical Constraints

- Raspberry-Pi will be used as the backbone of our gaming console. It will be responsible for taking care of the computationally heavy part of the system, as well as storing scenery data.
- The wireless card of the Raspberry-Pi will be utilized for communicating between mobile devices. Raspberry-Pi will also act as a server in this manner.
- A monocular camera that will be mounted to the Raspberry-Pi will be used for scanning the environment.
- Mobile device on which the augmented reality experience will be held needs to have a back camera.
- The AR applications will be available on Android operating system.

1.3.2 Social Constraints

- The developer framework will be open-source, allowing everyone to fork, develop their new ideas or improve existing ones.
- Multiplayer option of the developed games in the gaming console will help people to build social relations.

1.3.3 Implementation Constraints

- The structure of environment will be turned into game scene using Unity3D.
- Game developers will use Unity3D to develop apps for DepthCube.
- Network between mobile devices and console will be established using TCP/IP protocol.
- User's security details will be stored in a MySQL database.
- GitHub will be used as the version control repository.

1.3.4 Sustainability Constraints

- Our gaming console should have low energy consumption.
- Distributing the computationally-heavy part of the process to our gaming console will decrease the power consumption of the mobile devices, therefore increasing their battery life.

1.3.5 Economic Constraints

- The cost of the gaming console will not exceed $100.
- The framework that will be developed will be under GNU General Public License 2.0 and open-source. It will not require any payment for other developers to use.
- RGB-D sensor or stereo camera will not be used in acquiring scene data, due to their high prices increasing the overall cost of the product.
- Powerful GPUs that would normally increase the computation capabilities of the console will not be used, due to its high price.

1.3.6 Security Constraints

- Any kind of user data such as usernames, emails or passwords that is required by the application that uses our framework will be encrypted and secure.
- The latest 3D models that the user created will be encrypted and secure.
- Secure connection will be established between the console and the mobile devices, to prevent any information leak.

1.3.7 Language Constraints

- The console will be used worldwide hence it will have English as the default language.
- Other languages may be added depending on the console's popularity.

1.4 Professional and Ethical Issues

The secrecy of the users will be of the utmost importance. As users share their information of their private rooms, we are obligated to protect what is already theirs. We want to achieve this goal firstly using a username/password pair. Since we will already support multiple users, this will be important to prevent each user from accessing the information of the other corresponding users.

One other problem is with the possible danger of reverse-engineering, where adversary can access the memory after opening the case of the DepthCube. If we decide to leave the structure information in the game console without any protection, it can be easily decoded, bypassing the need for a username/password pair. To prevent this we need to encrypt the room information and decrypt it only when username/password is correctly identified. Once the game session has ended, decrypted information should be deleted until a new session.

On the other hand a professional problem is the usage of already licensed software. Since our design will heavily rely on the state-of-the-art computer vision algorithms, we have to use already existent software, since writing it from scratch is virtually impossible given the complexity of the algorithms. However since each software relies on many dependencies, we should be cautious of the possible violations of the licenses.

We will take the best available security measurements in order to protect user's secret about their rooms. Formation and location of the rooms of users should be strictly private, and DepthCube should be unalterable for any possible adversarial use. At the end of the day no adversary should be able to exploit the trust users have placed in DepthCube.

Requirements

2.1 Functional Requirements

- Players should have their own unique username/password.
- DepthCube should not require anything other than an Android device to run.
- Games should be controlled using a mobile device(smartphone/tablet).
- There should be two types of users for the product: developers and players.
- Players should be able to experience the Augmented Reality, given an already developed and ready-to-play game.
- Players should be able to upload videos for 3D construction.
- DepthCube should process user videos to create 3D structure
- Players should be able to download games from the online library.
- Developers should be able to design and develop their own games using our API that provides the location and structural information about the environment.

2.2 Non-Functional Requirements

- Each player should have access to the last 5 3D structures they created.
- DepthCube should be pocket-size.
- DepthCube should support Android 4.4 and above.
- A local and fast connection between DepthCube and mobile device is required. The communication latency between the DepthCube and the mobile device should be less than 40 ms.
- DepthCube should not require any kind of internet connection to work.
- Mobile interface of the DepthCube application should be user-friendly, clean and simplistic. Users must not have any problems with navigating themselves inside the application.
- Games developed for DepthCube should have at least 30 FPS.

- The provided API for developers should be well-documented, precise and succinct.
- DepthCube should scan user video and create a 3D structure in less than 1 hour.
- The provided API should be developed on Unity3D.
- DepthCube should support games developed using Unity3D version 5.0.0 and above.
- The server code should be written in C++.

References

Project Tango:
http://get.google.com/tango/

Structure From Motion:
http://bit.ly/2dSxD7L

OpenCV Camera Motion Estimation:
http://docs.opencv.org/3.1.0/d5/dab/tutorial_sfm_trajectory_estimation.html

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

ORB-SLAM, structure from motion implementation for monocular cameras:
http://webdiis.unizar.es/~raulmur/MurMontielTardosTRO15.pdf

Kinect with Skeletal Tracking:
https://msdn.microsoft.com/en-us/library/jj131025.aspx

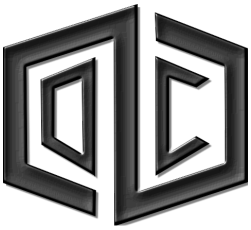
Kinect Sensor:
https://msdn.microsoft.com/en-us/library/hh438998.aspx

Kinect Sensor Hardware Specifications:
https://developer.microsoft.com/en-us/windows/kinect/hardware

Structure Sensor:
https://store.structure.io/store

CS 491 Senior Design Project

High-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)

December 30, 2016

Contents

1 Introduction **2**

1.1 Purpose of the System 2

1.2 Design Goals 2

1.3 Definitions, Acronyms and Abbreviations 4

2 Proposed Software Architecture **5**

2.1 Overview 5

2.2 Subsystem Decomposition 5

2.3 Hardware-Software Mapping 6

2.4 Persistent Data Management 7

2.5 Access Control and Security 8

2.6 Global Software Control 9

2.7 Boundary Conditions 9

3 Subsystem Services **10**

3.1 Unity UI 10

3.2 DepthCube Client 10

3.3 DepthCube Server 11

3.4 Account Manager 11

3.5 Network Manager 11

3.6 Reconstructor 12

3.7 Localizer 12

3.8 Database 12

4 Glossary **13**

5 References **15**

1 Introduction

1.1 Purpose of the System

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.

1.2 Design Goals

Low 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.

Minimum Number of Errors: 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 gameplay 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.

Reliability: We want to maximize the success us 3D reconstruction and localization, which stands as the single most important criteria of our implementation. Since our software will work as an API, our core success criteria will be working reliability and accurately in all circumstances. This will be the core deciding criteria for the third parties. To ensure that our system is reliable, we will make sure we carry the important computations to our server, which will perform state-of-part algorithms, which stands as reliable, yet require lots of computational power.

Ease of learning & Good Documentation: The API documentation will provide concise explanation of each function provided to developers. We want to handle most of the complex operations inside our API, thus granting the developer simple, yet powerful functions to create their applications. Although we will use some complex algorithms to create the environment and handle player movement, the API's documentation will maintain a set of simple functions for developers to use in their applications. Since our main users will be developers, a good documentation and tutorials is the key concept for making developers like us. We will emphasize commenting and documentation through our design.

Reusability: We will thrive to make important part of our code reusable, especially the computer vision algorithms, which are dealing with 3d reconstruction and localization. Since there is no working pipeline dealing with our work in the market, we are planning to publish our source code to make it available for future developers to reuse our code.

Efficiency: Mobile devices lack powerful batteries. We implement our system such that limited power is only drained by important calculations like localization. It is especially important since we are going to implement an API, therefore when third parties select our system, efficiency and its effect on the battery power will be highly effective.

Portability: We want our system to work in many platforms. Our common goal will be implementing the core algorithms in low-level languages like C, which will enable us carry all the computation to other platforms. Although we initially aim for a Android API, carrying it to other platforms should be feasible. More than that, we make most of the computations on our server, which makes carrying our implementation into other devices even more easy. At last, our GUI and interaction design in Unity, which is available in every major platform. While adapting our implementation into other systems, we will only need to make minor adjustments.

Traceability of Requirements: Our requirements will be achieving certain accuracies in popular datasets in localization community. Those that are important stand as UJIIndoor Loc Dataset [6] and Navvis [7] indoor localization and mapping dataset. Accuracy is one of the most important requirements for our project and therefore traceability stands as the only way of ensure we achieve certain accuracy criterion. Having an easily adaptable API structure is already is one of our goals, therefore making our task easier.

High-performance: We are compelled to deliver a high-performance API since a real-time system becomes undesirable if the application freezes/lags often. We strive to provide robust servers and cutting-edge processing algorithms to tackle these obstacles. The algorithms will be stress-tested to confirm DepthCube applications operate with adequate performance.

1.3 Definitions, Acronyms and Abbreviations

API: Application Programming Interface, is a set of subroutine definitions, protocols, and tools for building application software [1]

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 [2]

GUI: Graphical User Interface is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. [3]

OS: Operating System is system software that manages computer hardware and software resources and provides common services for computer programs.[4]

DB: Database is a collection of information that is organized so that it can easily be accessed, managed, and updated.[5]

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

2 Proposed Software Architecture

2.1 Overview

Proposed architecture of DepthCube is comprised of three main parts: client, server and the database. Client is the mobile device that is the subject of the DepthCube experience. The user will be navigated by the user interface that the DepthCube application has for the client.

When the application starts, client and server firstly will communicate to verify the username and password to validate the login process. Then, when the user starts recording a video using his/her mobile device's camera, frames that are recorded by the camera will be processed and the resulting point cloud, as well as mesh data and several point descriptor data will be sent from mobile device to server. Server will be responsible for storing these data into the database in an encrypted form. Server will also process these data to provide localization as well as the reconstruction of the 3d environment; and will send the desired results back to client. Client will receive reconstruction data from the server, or adjust itself and correct its results according to server's feedback on the localization task. Database will both store user credentials data and associated localization and reconstruction data (i.e. FAST descriptors, point cloud, mesh data of the scene) for each user.

2.2 Subsystem Decomposition

Our system can be decomposed into two subsystems that represent client and server logic, which will be further decomposed into the meaningful pieces of actions that are performed by the respective subsystems.

Client subsystem consists of the *GUI* that is generated using Unity, and three subsystems that possess the functionality of the clients: *Localizer* which is responsible for localization function, *Reconstructor* which is responsible for reconstructing the scene in realtime on the mobile device's screen, and the *Network Manager* which regulates the network traffic between client and server.

Server subsystem consists of four subsystems that possess the functionality of the server machine: *Localizer* which is responsible for providing feedback to clients on their localization task, *Reconstructor* that is responsible of reconstructing scene when inputted a point cloud from the client, *Network Manager* that is the server-end of the Network logic, and *Account Manager* that is responsible of managing and protecting user data as well as handling login requests.

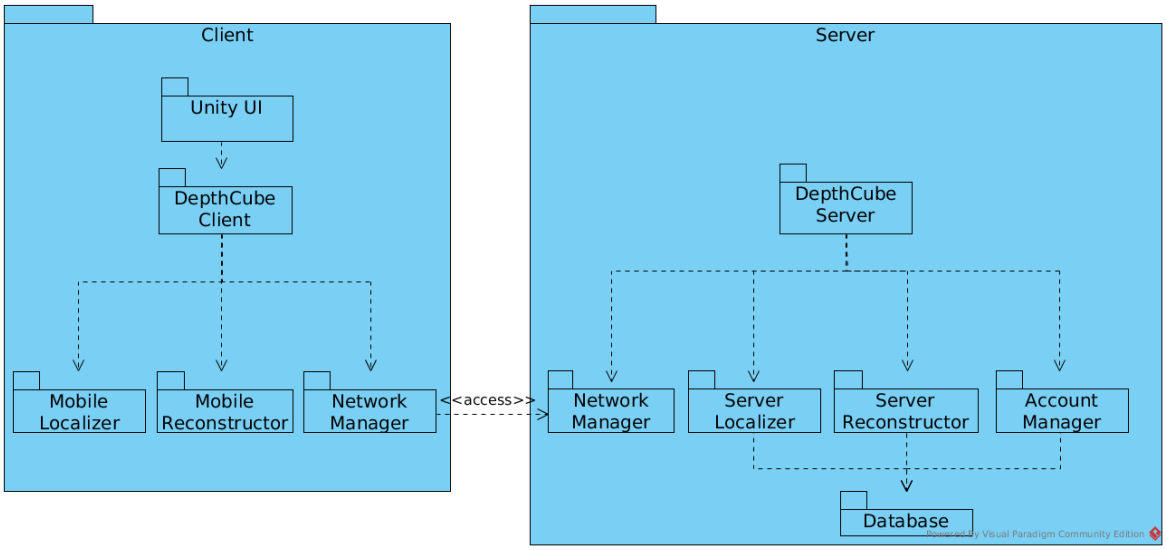


Figure 1: Package Diagram.

2.3 Hardware-Software Mapping

In the simplest terms, hardware-software mapping of DepthCube will consist of mainly two parts: A mobile device in which the client will run; and a server machine in which both our server application and database will run, although database does not necessarily need to run in the same machine that server application runs. Mobile device will be responsible for the client logic which will comprise of the localization and reconstruction tasks that can be navigated through the client GUI. On the other hand, server machine will be responsible of storing data in the DB, responding to DB queries and improvement on the localization and reconstruction tasks.

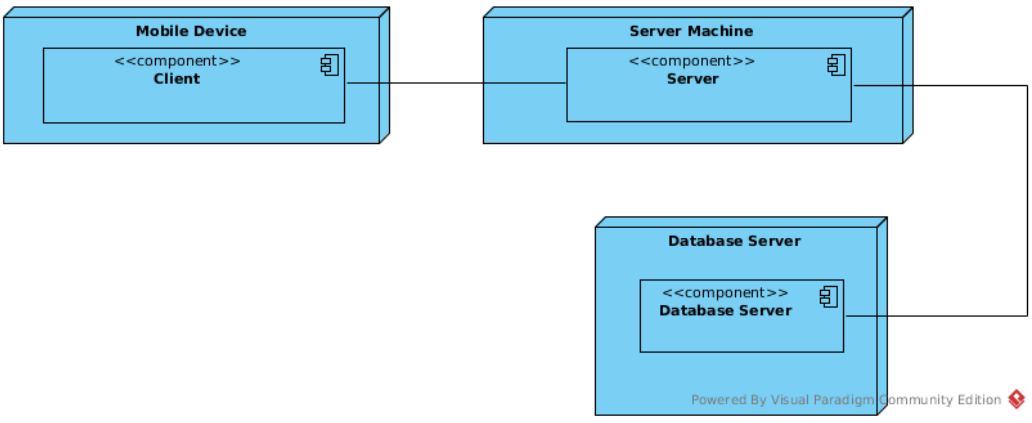


Figure 2: Deployment Diagram.

2.4 Persistent Data Management

DepthCube has a single source of persistent data, point clouds obtained after 3D Reconstruction and their corresponding mesh construction. This data will be composed of simple features, yet its vast size and fast query time provide our application with fast results. Furthermore, since even the simplest models include thousands of points, usually consisting of megabytes of data, we need to be able to handle the size of it efficiently.

In our point cloud storage, we will have the following features in order to represent and localize our images with respect to some known reference world coordinate system:

- 3D Point Cloud, consists of listed 3D locations of calculated keypoints
- Mesh data of the constructed from corresponding point cloud
- Raw pixel value for each point
- BRIEF Descriptor for each point
- FAST Descriptor for each point

While BRIEF descriptors will be used for mobile device for fast tracking, FAST Descriptors will be used by server to enable global localization of the camera frame. Therefore there will be constant communication between the mobile device and the server, where BRIEF descriptors will be communicated. On the other hand FAST Descriptor communication will be one-way, therefore its storage will be static and will only be used for queries coming from the mobile device.

Our aim in storing our persistent data in the database is to enable fast query response time, which will result in fast localization. This approach requires aggressive preprocessing. This pre-processing comes in different forms, but mostly indexing camera frames and their respective descriptors in a way such that we have strong priors about the location of the camera. In the literature there are several different approaches such as clustering the camera for known locations, or basically for its color histogram or storing frames as they form a graph, considering the number of matches between FAST Descriptors each frame. How to store this data in our database will be a design choice, where we need to consider performance in order to enable fast localization.

Our access policy for persistent data is quite straightforward. With the exception of mesh data, all data will be stored in the Server side. This data will not interact with the user in any way. Therefore the relevant computations will be done in the server. The only data client side will have is the mesh information, which will be used for visualizing the environment in the mobile device. This data will be transferred to the mobile device in

the beginning of the session. Since we will simplify the mesh data as a pre-processing part, the cost of data communication will be relatively low.

It is also a necessity to make our data distributed. Since users will potentially have great geographical variety, our data also needs to be close to our users. This will eventually increase our accuracy, since the communication time between user and server naturally constitutes an important bottleneck. When communication time decreases, more correction can be made in the global position of the mobile device. However we do not consider implementing a distributed system for the initial phase of the project, which will create unnecessary workload despite providing little improvement.

2.5 Access Control and Security

Before using DepthCube and downloading protected information users are required to register to the system. This feature is required in order to protect the sensitive data of our users, including the 3D construction of their rooms in both point cloud and mesh information, which perfectly describes the details of the target room.

To prevent any potential leak of information to malicious users and other third-party applications we will use the following policy:

- Give as little data as you can to third party applications
- Always require explicit confirmation from users before sharing their information to other applications
- Encrypt the stored information, yet abstain from affecting the runtime performance of the system.

To materialize the first principle, we will only share the mesh information with other applications, which gives the information of visual appearance of the room. Other information is bound to be unnecessary, as our servers are capable of retrieving any other query regarding localization information. We cannot abstain from giving mesh information, since this is one of the promises of our project. On the other hand, we can perfectly refrain from leaking other information with pushing computations to the server, and only returning to localization and reconstruction results to the device. We are aware of the trade-off between security and speed, therefore we want to maximize our performance by protecting user data in the server, instead of protecting it in the mobile device which is naturally prone to errors. We are aware that it is practically infeasible to protect user data where it is bound to be decrypted and extensively processed in an operating system like Android.

Once the user is registered, they can enter their login information to access their account. The database stores user's info in a hashed form and checks whenever a login

attempt occurs if credentials are correct. If login request is granted, the user can use Depth Cube applications and access their stored rooms. The rooms that users scan are also kept in the database. These are private and we are obligated to keep them secure. This information will be protected via user designated password.

2.6 Global Software Control

Internal Control: The subsystems will use asynchronous callbacks for inter-process communication. Services will use their internal methods to notify other subsystems of their own status and needs. Of course, while doing these, it is essential for the system to avoid deadlocks so that the callbacks shall not block their callers. Each service will have their respective threads that will communicate with other services.

External Control: Two systems, client and server, communicate with each other using network protocols. Server listens to the clients and responses them according to their needs and requests, whereas clients request a service from the server. This service might be about a login activity for a user, a query to the DB that server has control over, or a reconstruction task for the point cloud that is sent to server.

Concurrency Control: With multiple subsystems working simultaneously, concurrency control in the system becomes more of an issue. Therefore, services need to be managed so that there shall be no race conditions that might cause inconsistencies within the system. For instance, when the server localizer subsystem is sending data to client about the localization, it both requires the location data from the client meanwhile trying to update it. This issue, if not resolved properly, might cause residual error to increase in localization.

2.7 Boundary Conditions

Initialization: To bring system to initialization state, we need to sync our client with the server. There are two models to be synced; the mesh data which will be sent to the mobile device for a single time -and it will stay synced since there will not be any more computation-, also the initial localization of the device. The latter is especially important for SLAM systems, where the system initialization is an important problem. Although there are many solutions to this problem in the literature, the particular one we will use especially depends on the implementation we are going to choose. This topic will be further discussed in coming reports. The other important part of initialization process is the decryption of the data, which is originally held protected in the database. We assume, after data is communicated, before localization and construction process has begun, data

is decrypted so that we can play with the data in a regular fashion. Therefore decryption is an important part of the initialization process.

Termination: The most important termination condition is getting rid of the encrypted files, both on the server and client side. Yet this activity is as simple as using free command to deallocate necessary memory. The termination further requires a notice to send to server, so that it will also free the resources to that are allocated to server that particular user. There also stands several implementation oriented terminations cases like closing the Android camera and closing the socket connection, which are not important enough to mention here.

Failure: There are two failure cases, which will be handled separately. Firstly, DepthCube needs to deal with occasional failure during the localization step. If image based localization fails, we need to rely on other primitive sources of information, like gyroscope and accelerator. If this failure state remains too long, we need to warn user that video input does not involve sufficient information. This failure might result in dropping the connection between the server and the user. The other scenario involves where server and user cannot have a healthy communication. This will result in very sparse corrections between server and the user, which will drastically decrease the accuracy of the localization, since all the decisions will be done in the local context. In this case we will not stop the localization, but we will warn the user that results might be unreliable due to the poor Internet connection.

3 Subsystem Services

3.1 Unity UI

UnityUI subservice is used to demonstrate the capabilities of the DepthCube. It will be used for developers as a template application for DepthCube API.

3.2 DepthCube Client

DepthCube Client will provide the following services:

- Login
- RegisterUser
- GetScenes

- Locate
- Reconstruct

DepthCube client subsystem provides services that DepthCube API provides. It controls subsystems that runs in the mobile device.

3.3 DepthCube Server

DepthCube Server subsystem is the controller subsystem that controls services that runs in the server.

3.4 Account Manager

Account Manager will provide the following services:

- Login
- RegisterUser

Account manager provides authentication and registration services. This subsystem handles user specific data.

3.5 Network Manager

Network Manager will provide the following services:

- Send
- Receive
- Listen

Network Manager handles server - client communication.

3.6 Reconstructor

Mobile Reconstructor:

Mobile Reconstructor subservice will handle temporary reconstruction for user to see the reconstruction progress on the mobile device. It will provide the 'reconstruct' service.

Server Reconstructor:

Server Reconstructor will provide the following services:

- ReconstructPointCloud
- PointCloud2Mesh

Server Recunstructor subservice handles reconstruction of the scene. This service is used to generate point cloud and mesh of the scene.

3.7 Localizer

Localizer subsystem handles localization service. It finds the location of the user using the reconstructed scene information. Localization service is distributed in 2 subservice.

Mobile Localizer:

Mobile Localizer will provide the 'Locate' service. It will be used to find approximate location of the user on the mobile device. Since it runs on the mobile device, it is fast and lightweight.

Server Localizer:

Server Localizer will provide the 'Locate' service. It will provide accurate and precise location of the user. It runs slower compared to mobile localizer while performing more accurate and precise localization.

3.8 Database

Database subsystem will be used to store persistent data such as user authentication information and scene information.

4 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.[6]

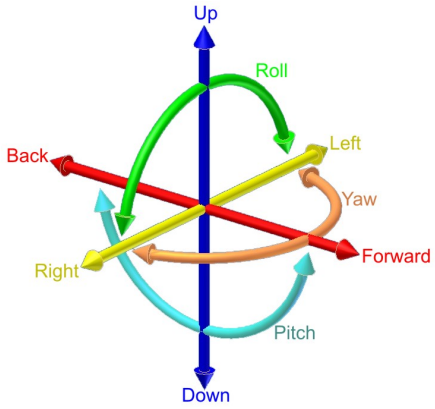


Figure 3: Six Degrees of Freedom.

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

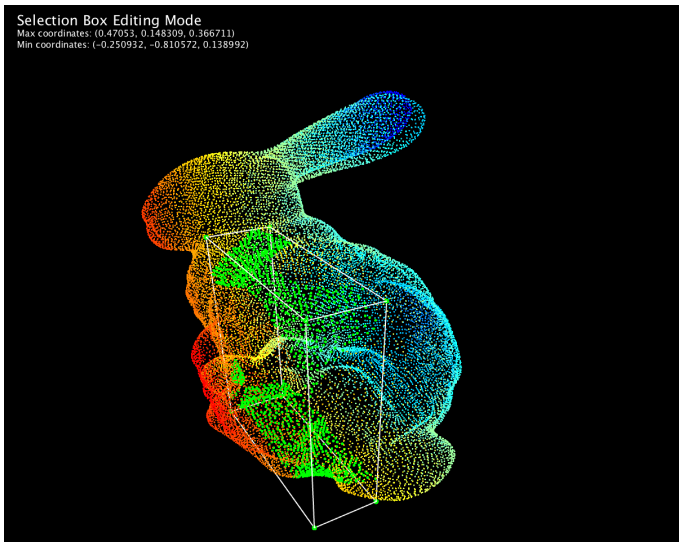


Figure 4: A sample colorized point cloud [5].

[6]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.

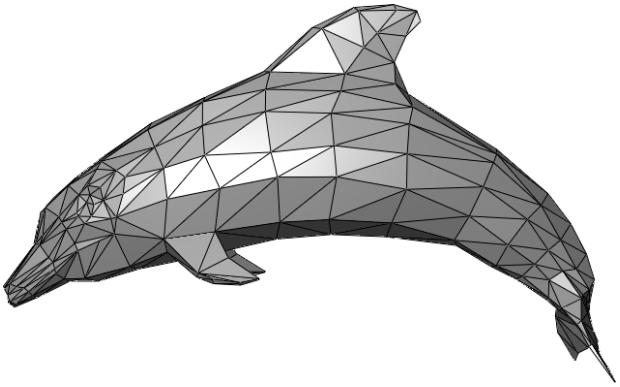


Figure 5: A triangle mesh representing a dolphin.

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. [7]

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. [8]

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

5 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).

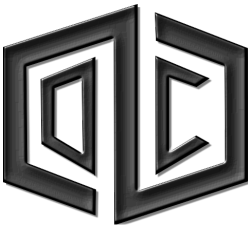
[5] 3D Point Cloud Editor `http://paradise.caltech.edu/~yli/software/pceditor.html`

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

[7] Navigation based on Visual Information (NAVVIS) Dataset. `http://www.navvis.lmt.ei.tum.de/dataset/`

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

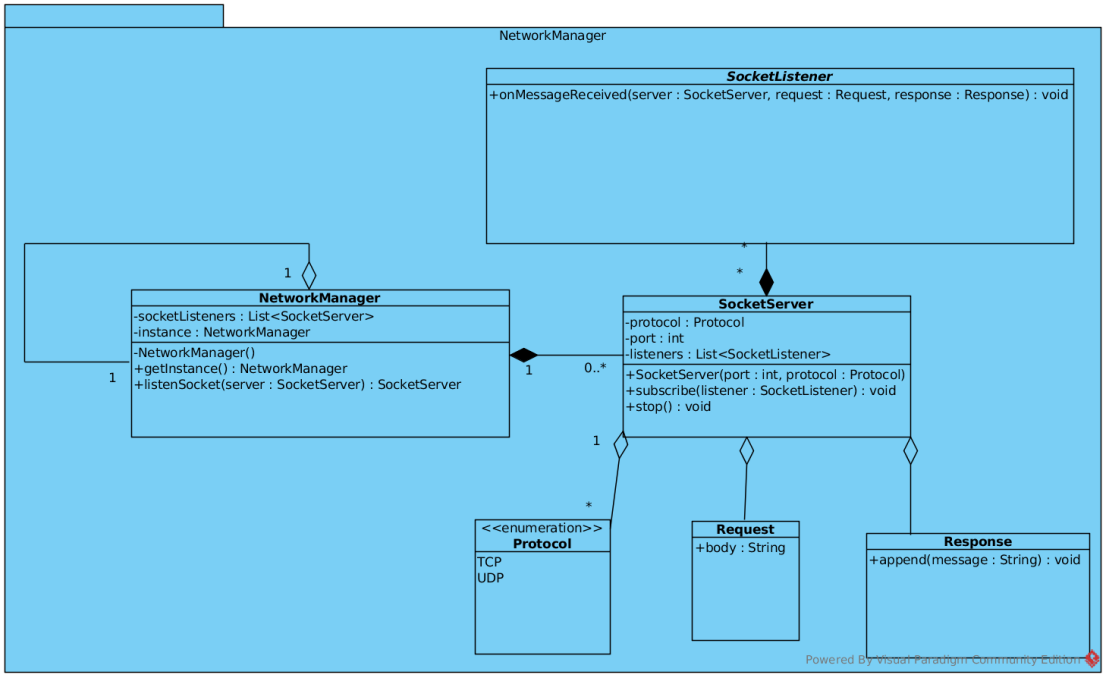


Figure 1: Network Manager Package.

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.

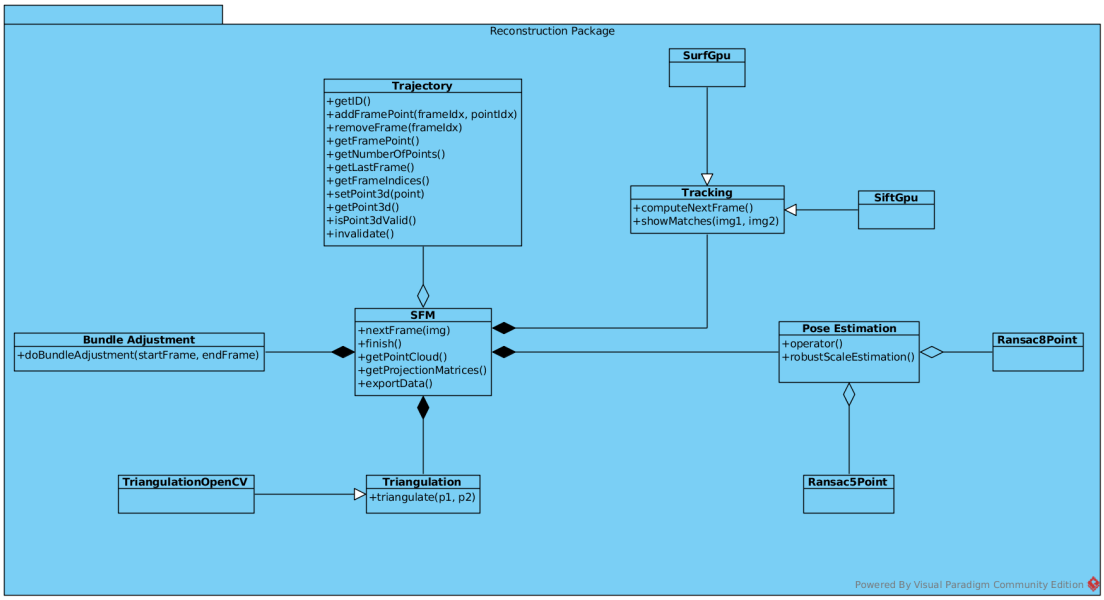


Figure 2: Reconstruction package

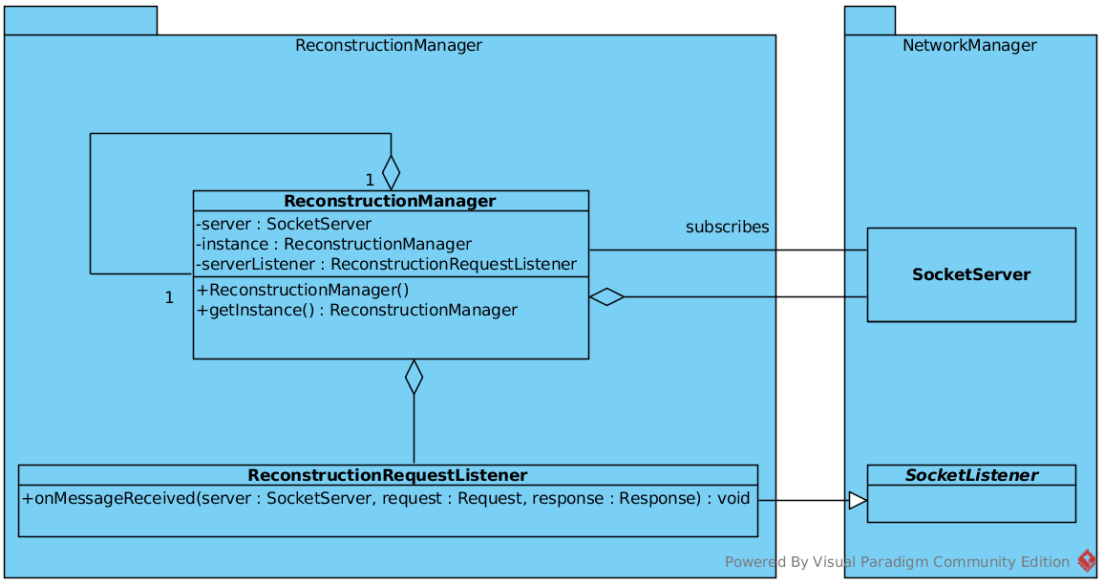


Figure 3: Reconstruction Manager with its relation to Network Manager

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.

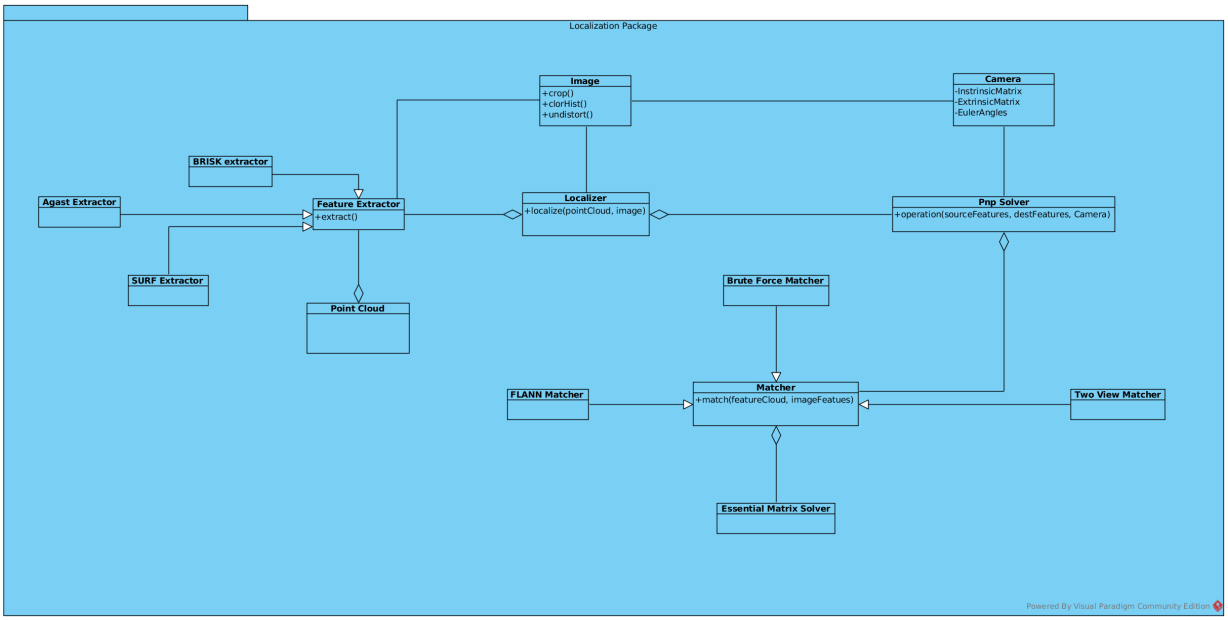


Figure 4: Localization package

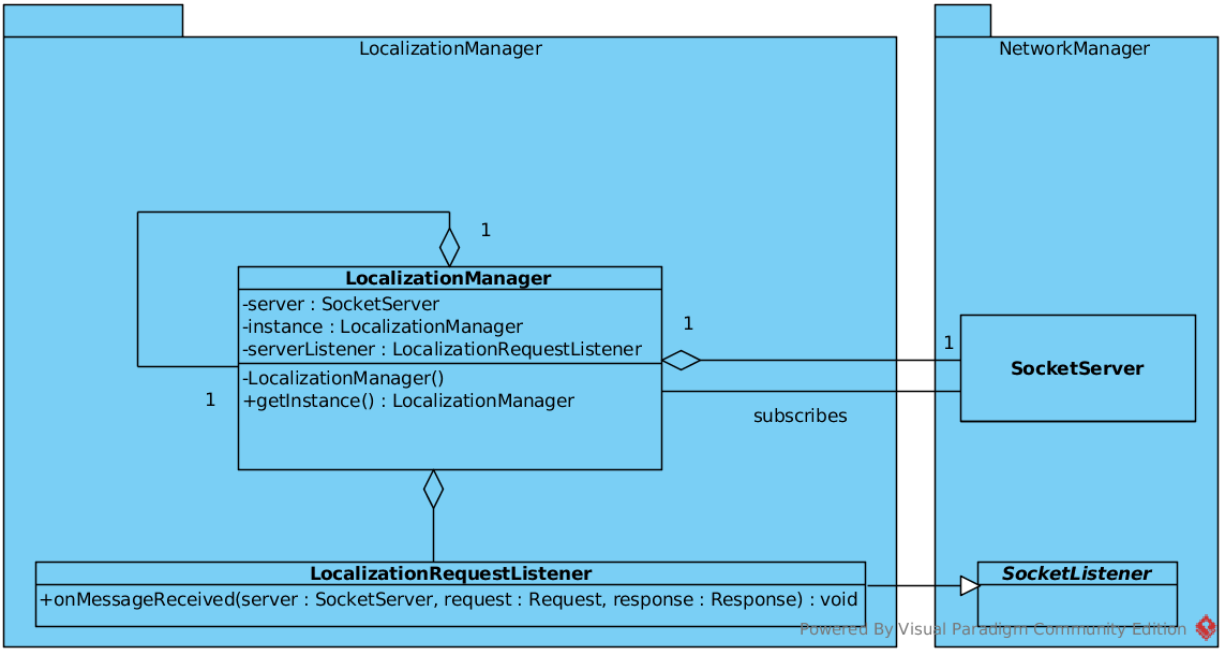


Figure 5: Localization Manager with its relation to Network Manager

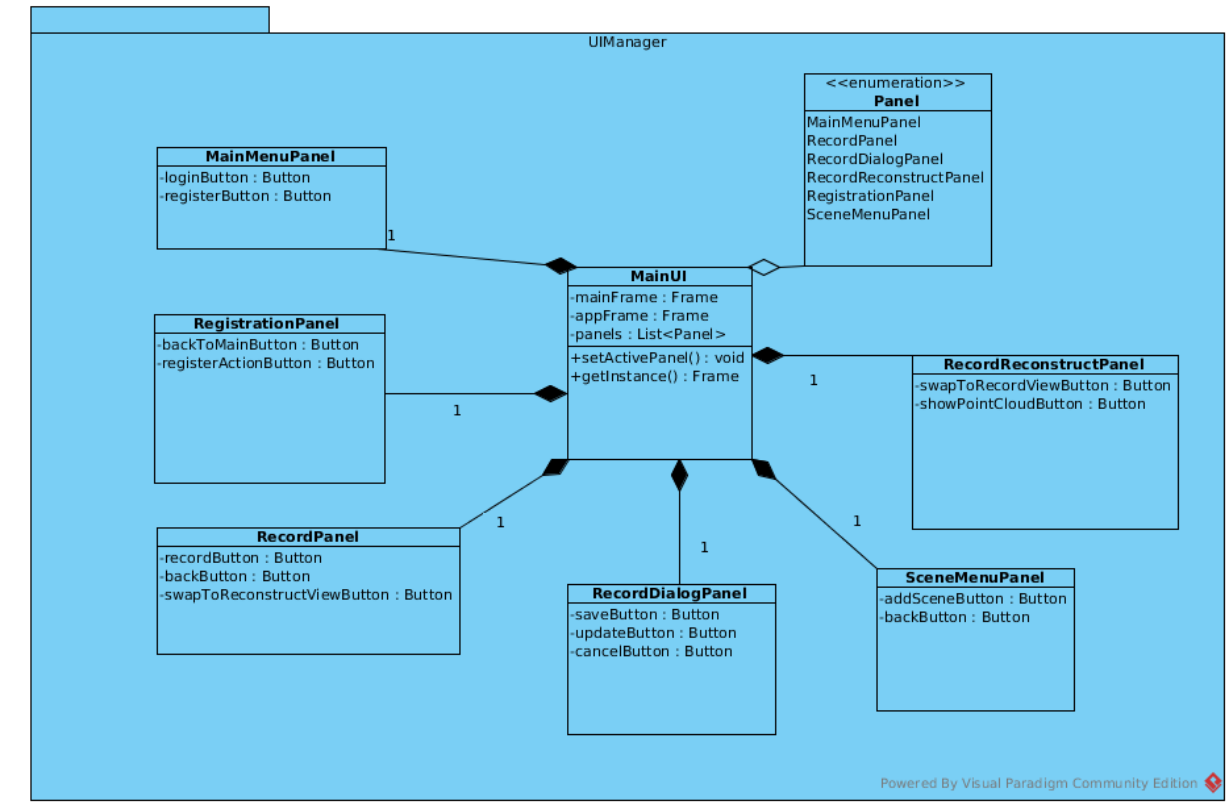


Figure 6: UIManager package

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.

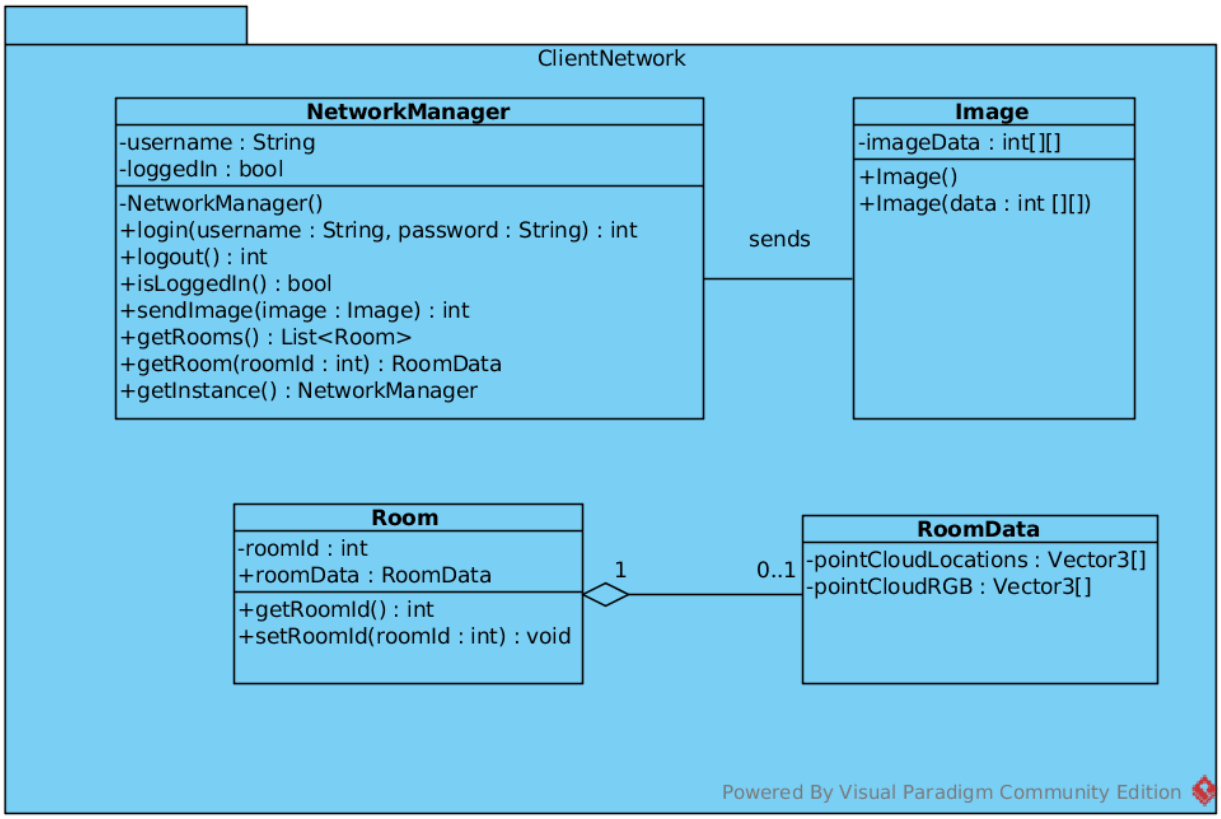


Figure 7: Client Network Manager package

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 reconstruction 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.

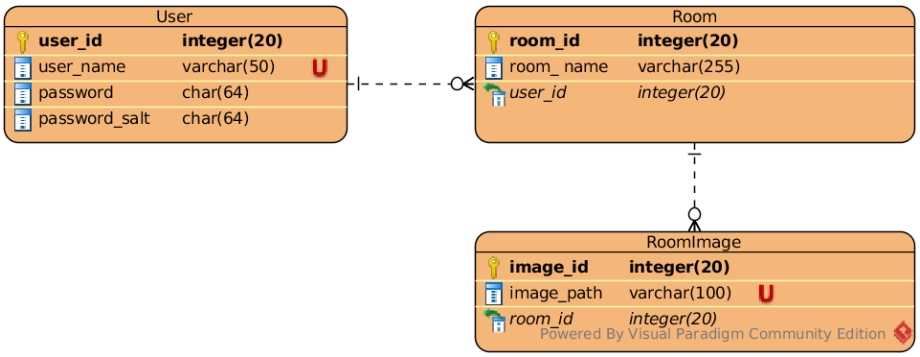


Figure 8: ER Diagram for database constructed in the server.

```
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 weaknesses-

are 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]

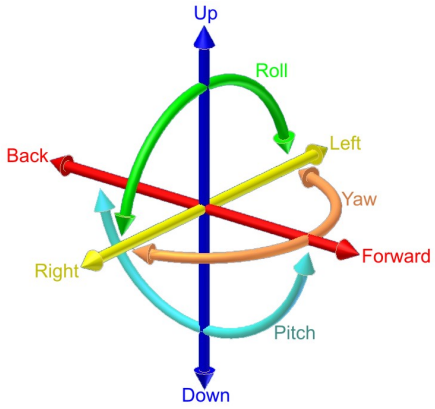


Figure 10: Six Degrees of Freedom.

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

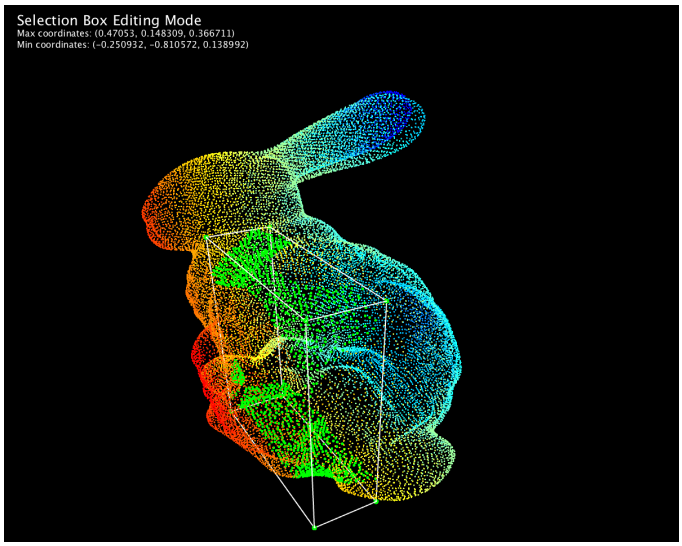


Figure 11: A sample colorized point cloud [5].

[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.

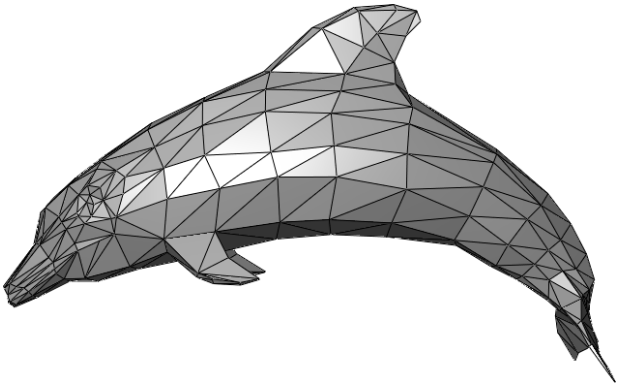


Figure 12: A triangle mesh representing a dolphin.

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/`

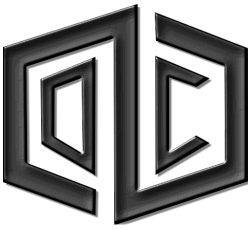
[9] COLMAP - Structure-From-Motion and Multi-View Stereo `http://people.inf.ethz.ch/jschoenb/colmap/`

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

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

CS 491/2 Senior Design Project

Final 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)

May 11, 2017

Contents

1 Introduction **3**
1.1 Overview . . . 3
1.2 Object Design Trade-offs . . . 4
1.3 Engineering Standards . . . 5
1.4 Definitions, Acronyms and Abbreviations . . . 5
1.4.1 Technical Abbreviations . . . 6

2 Final Architecture and Design **7**
2.1 Subsystem Decomposition . . . 7
2.2 Hardware/Software Mapping . . . 7
2.3 Persistent Data Management . . . 8

3 Packages / Class Diagrams **10**
3.1 Server . . . 10
3.1.1 Network Manager . . . 10
3.1.2 Reconstruction Manager . . . 10
3.1.3 Localization Manager . . . 11
3.2 Client . . . 13
3.2.1 UI Manager . . . 13
3.2.2 Client Network Manager . . . 13

4 Final Status of the Project **15**
4.1 Overview . . . 15
4.2 Flow of Work - Server . . . 16
4.3 Flow of Work - Client . . . 17

5 Sample Output **18**

6 Impact of the Engineering Solution **19**

7 Related Contemporary Issues **20**

8 Used Tools and Technologies **21**

8.1 Utility Tools 21

8.2 Third Party Libraries and Frameworks 21

8.3 Internet Resources 22

9 Software / Hardware System **23**

10 Conclusion **23**

11 References **24**

1 Introduction

1.1 Overview

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.

1.2 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.

1.3 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]

1.4 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]

---

[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`

[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`

1.4.1 Technical Abbreviations

SfM: Structure from Motion. SfM is the process of creating the model (structure) of the environment by utilizing the movement / motion of the device [9].

PnP: Perspective-n-Point Problem is the problem of estimating the pose of a calibrated camera from N known correspondences between space control points and image points[10].

RANSAC: Random Sample Consensus. RANSAC is an iterative parameter estimation method for a mathematical model where the outliers in the observed data does not influence the model. Therefore RANSAC is used to detect outlier data [11].

SIFT: Scale-Invariant Feature Transform. SIFT is an algorithm to detect and extract features from images. It can detect **keypoints** and generate **descriptors** from these keypoints [12].

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. [13]

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. [14]

[9] SfM - http://mi.eng.cam.ac.uk/~cipolla/publications/contributionToEditedBook/2008-SFM-chapters.pdf
[10] PnP - http://nlpr-web.ia.ac.cn/2006papers/gjkw/gk2.pdf
[11] RANSAC - http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FISHER/RANSAC/
[12] SIFT - https://www.inf.fu-berlin.de/lehre/SS09/CV/uebungen/uebung09/SIFT.pdf
[13] Structure From Motion - https://en.wikipedia.org/wiki/Structure_from_motion
[14] SLAM - https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

2 Final Architecture and Design

2.1 Subsystem Decomposition

Our system can be decomposed into two subsystems that represent client and server logic, which will be further decomposed into the meaningful pieces of actions that are performed by the respective subsystems.

Client subsystem consists of the *GUI* that is generated using Unity, and three subsystems that possess the functionality of the clients: *Localizer* which is responsible for localization function, *Reconstructor* which is responsible for reconstructing the scene in realtime on the mobile device's screen, and the *Network Manager* which regulates the network traffic between client and server.

Server subsystem consists of four subsystems that possess the functionality of the server machine: *Localizer* which is responsible for providing feedback to clients on their localization task, *Reconstructor* that is responsible of reconstructing scene when inputted a point cloud from the client, *Network Manager* that is the server-end of the Network logic, and *Account Manager* that is responsible of managing and protecting user data as well as handling login requests.

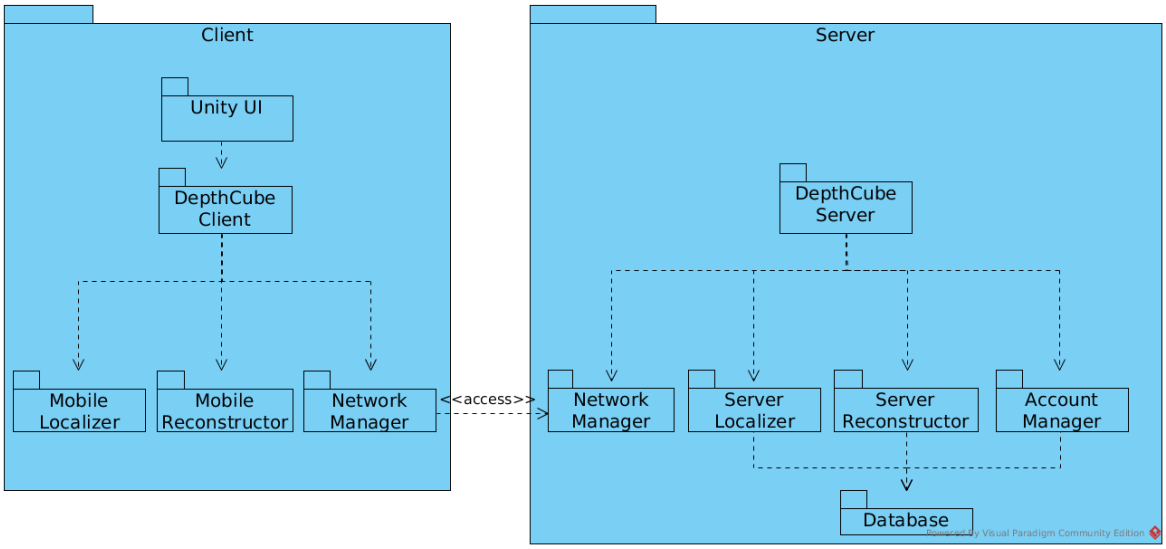


Figure 1: Package Diagram.

2.2 Hardware/Software Mapping

In the simplest terms, hardware-software mapping of DepthCube will consist of mainly two parts: A mobile device in which the client will run; and a server machine in which both our

server application and database will run, although database does not necessarily need to run in the same machine that server application runs. Mobile device will be responsible for the client logic which will comprise of the localization and reconstruction tasks that can be navigated through the client GUI. On the other hand, server machine will be responsible of storing data in the DB, responding to DB queries and improvement on the localization and reconstruction tasks.

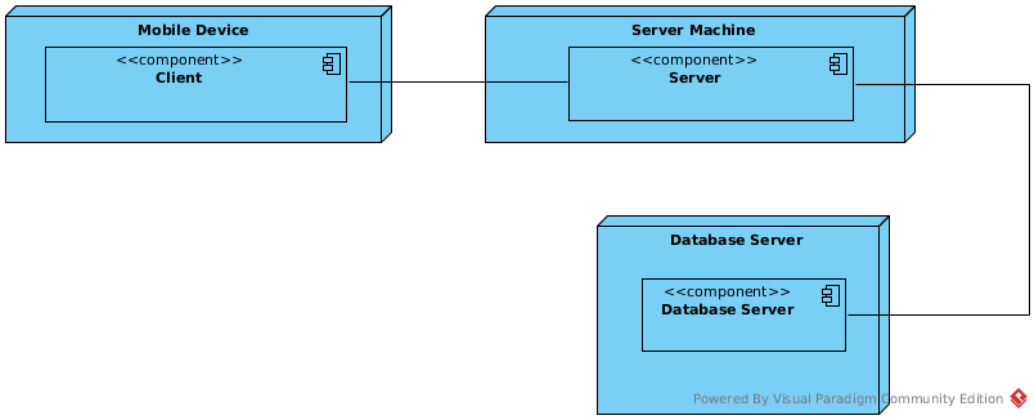


Figure 2: Deployment Diagram.

2.3 Persistent Data Management

DepthCube has a single source of persistent data, point clouds obtained after 3D Reconstruction and their corresponding mesh construction. This data will be composed of simple features, yet its vast size and fast query time provide our application with fast results. Furthermore, since even the simplest models include thousands of points, usually consisting of megabytes of data, we need to be able to handle the size of it efficiently.

In our point cloud storage, we will have the following features in order to represent and localize our images with respect to some known reference world coordinate system:

- 3D Point Cloud, consists of listed 3D locations of calculated keypoints
- Mesh data of the constructed from corresponding point cloud
- Raw pixel value for each point
- BRIEF Descriptor for each point
- FAST Descriptor for each point

While BRIEF descriptors will be used for mobile device for fast tracking, FAST Descriptors will be used by server to enable global localization of the camera frame. Therefore there will be constant communication between the mobile device and the server, where BRIEF descriptors will be communicated. On the other hand FAST Descriptor communication will be one-way, therefore its storage will be static and will only be used for queries coming from the mobile device.

Our aim in storing our persistent data in the database is to enable fast query response time, which will result in fast localization. This approach requires aggressive pre-processing. This pre-processing comes in different forms, but mostly indexing camera frames and their respective descriptors in a way such that we have strong priors about the location of the camera. In the literature there are several different approaches such as clustering the camera for known locations, or basically for its color histogram or storing frames as they form a graph, considering the number of matches between FAST Descriptors each frame. How to store this data in our database will be a design choice, where we need to consider performance in order to enable fast localization.

Our access policy for persistent data is quite straightforward. With the exception of mesh data, all data will be stored in the Server side. This data will not interact with the user in any way. Therefore the relevant computations will be done in the server. The only data client side will have is the mesh information, which will be used for visualizing the environment in the mobile device. This data will be transferred to the mobile device in the beginning of the session. Since we will simplify the mesh data as a pre-processing part, the cost of data communication will be relatively low.

It is also a necessity to make our data distributed. Since users will potentially have great geographical variety, our data also needs to be close to our users. This will eventually increase our accuracy, since the communication time between user and server naturally constitutes an important bottleneck. When communication time decreases, more correction can be made in the global position of the mobile device. However we do not consider implementing a distributed system for the initial phase of the project, which will create unnecessary workload despite providing little improvement.

3 Packages / Class Diagrams

3.1 Server

3.1.1 Network Manager

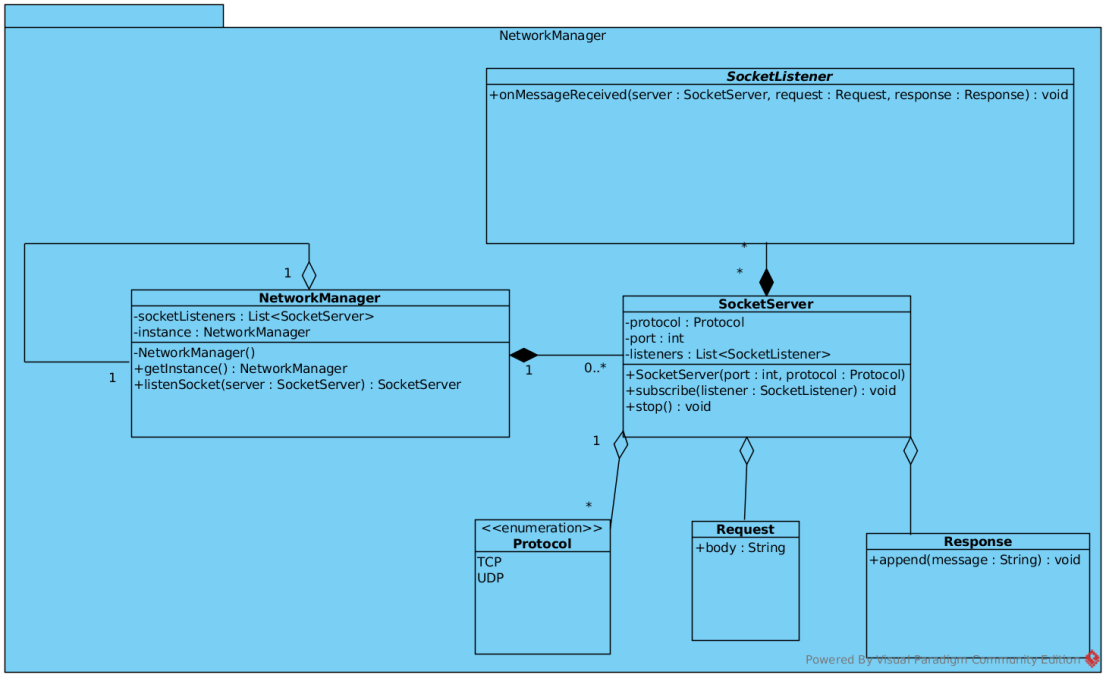


Figure 3: Network Manager Package.

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.

3.1.2 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.

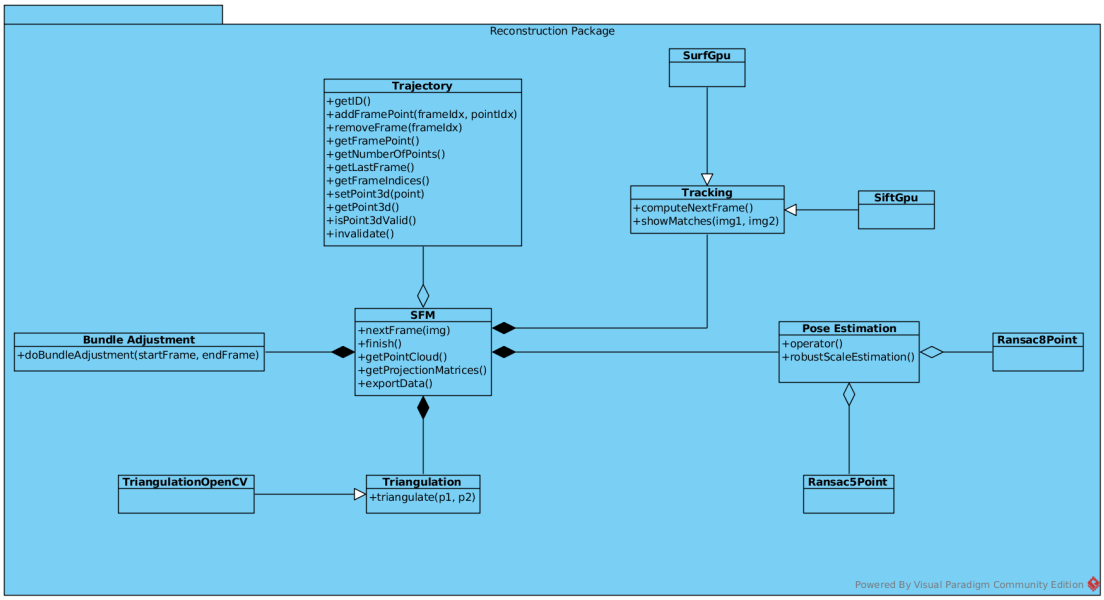


Figure 4: Reconstruction package

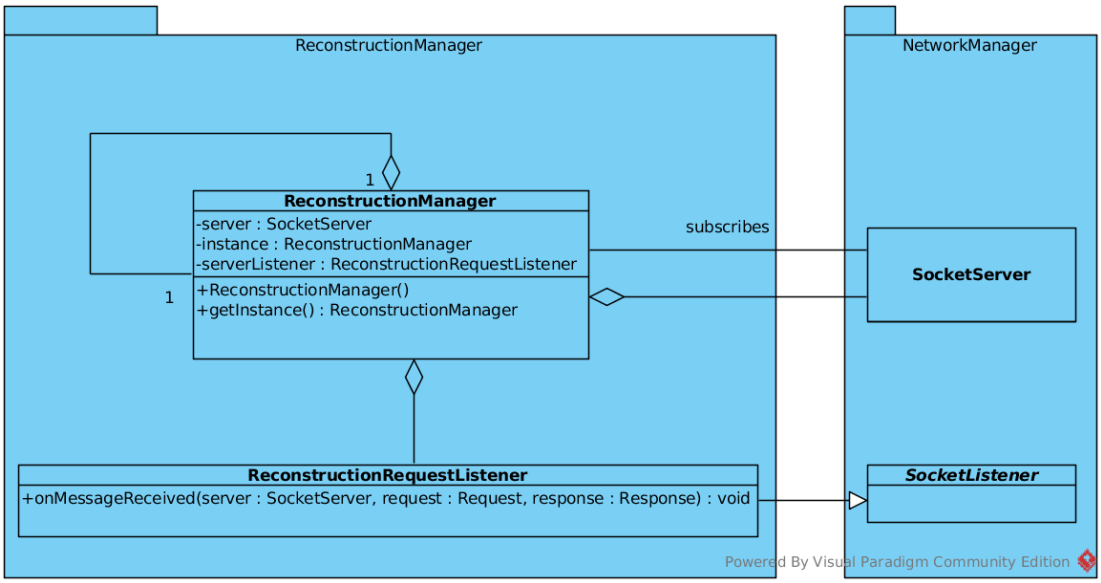


Figure 5: Reconstruction Manager with its relation to Network Manager

3.1.3 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.

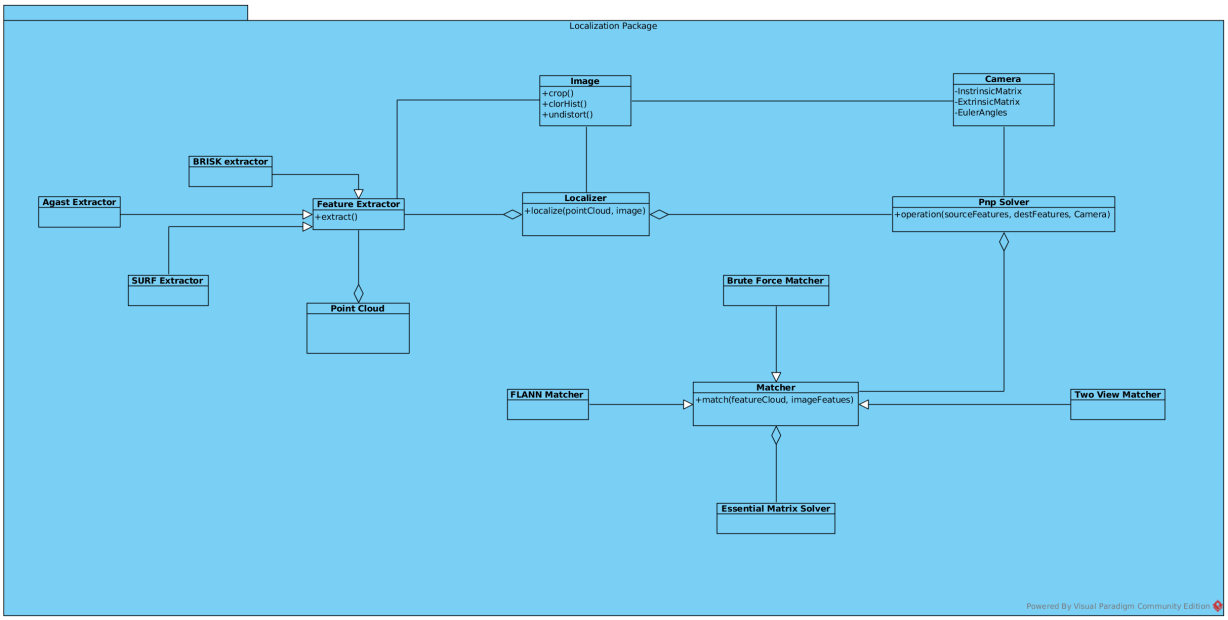


Figure 6: Localization package

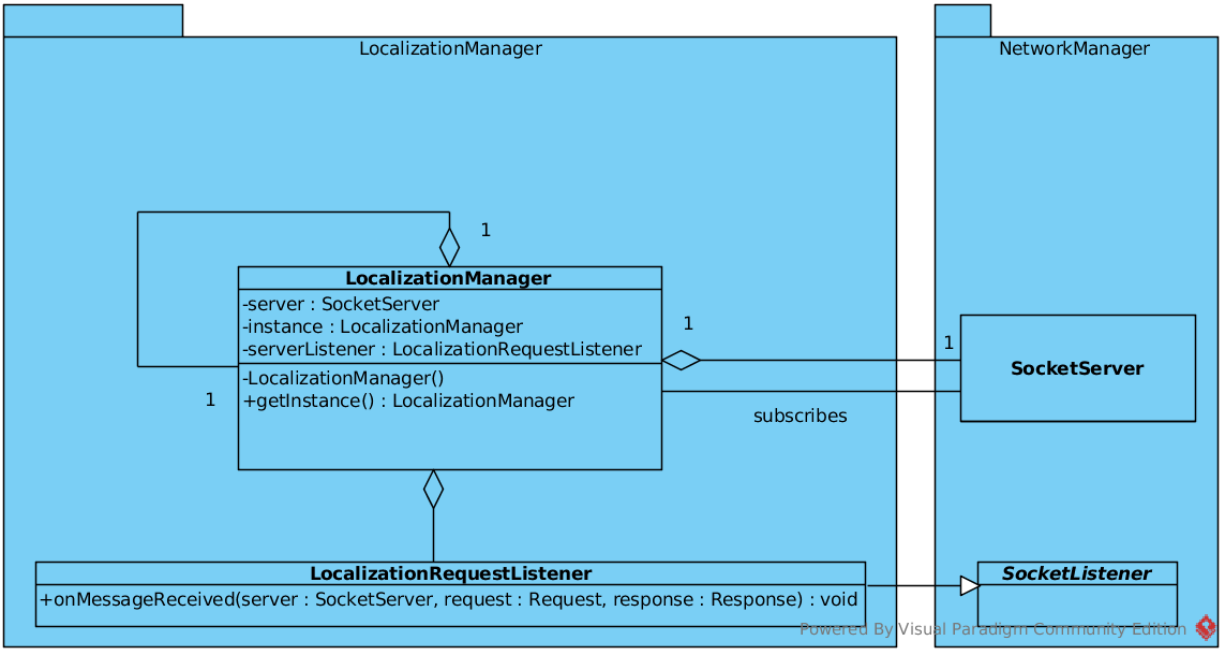


Figure 7: Localization Manager with its relation to Network Manager

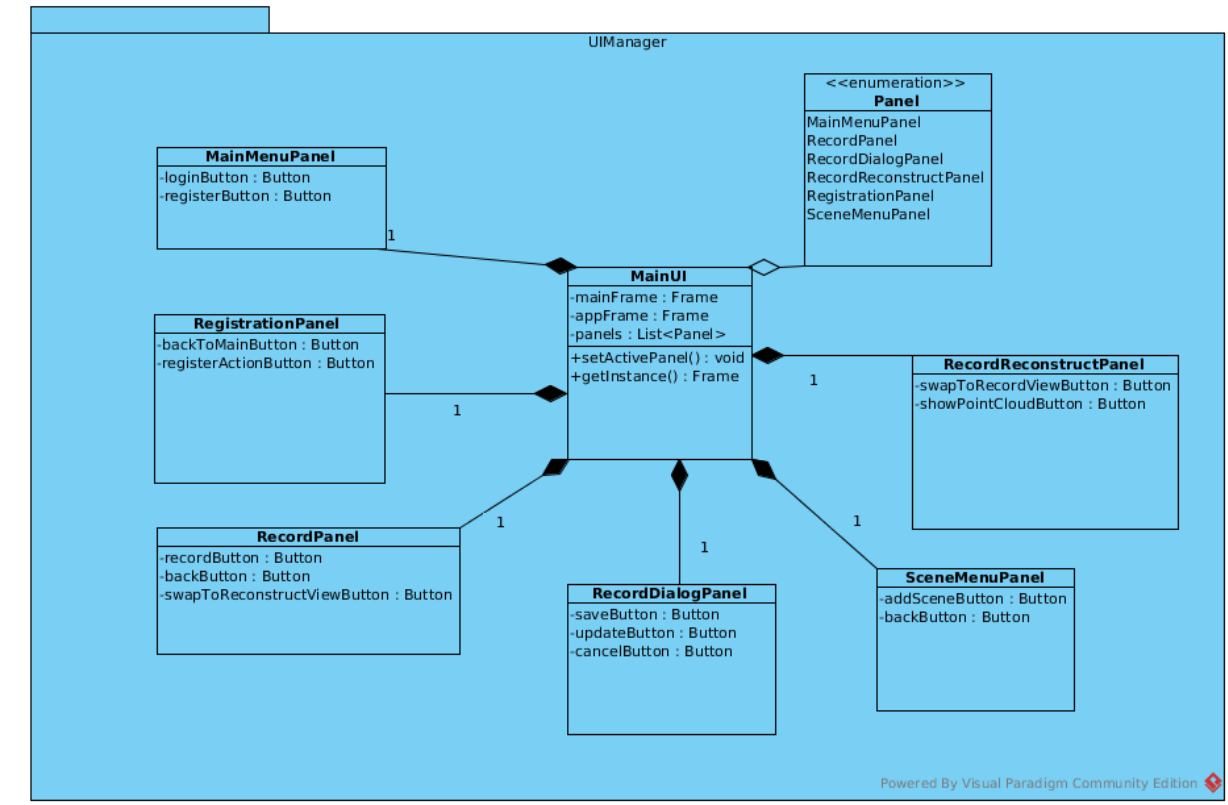


Figure 8: UIManager package

3.2 Client

3.2.1 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.

3.2.2 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.

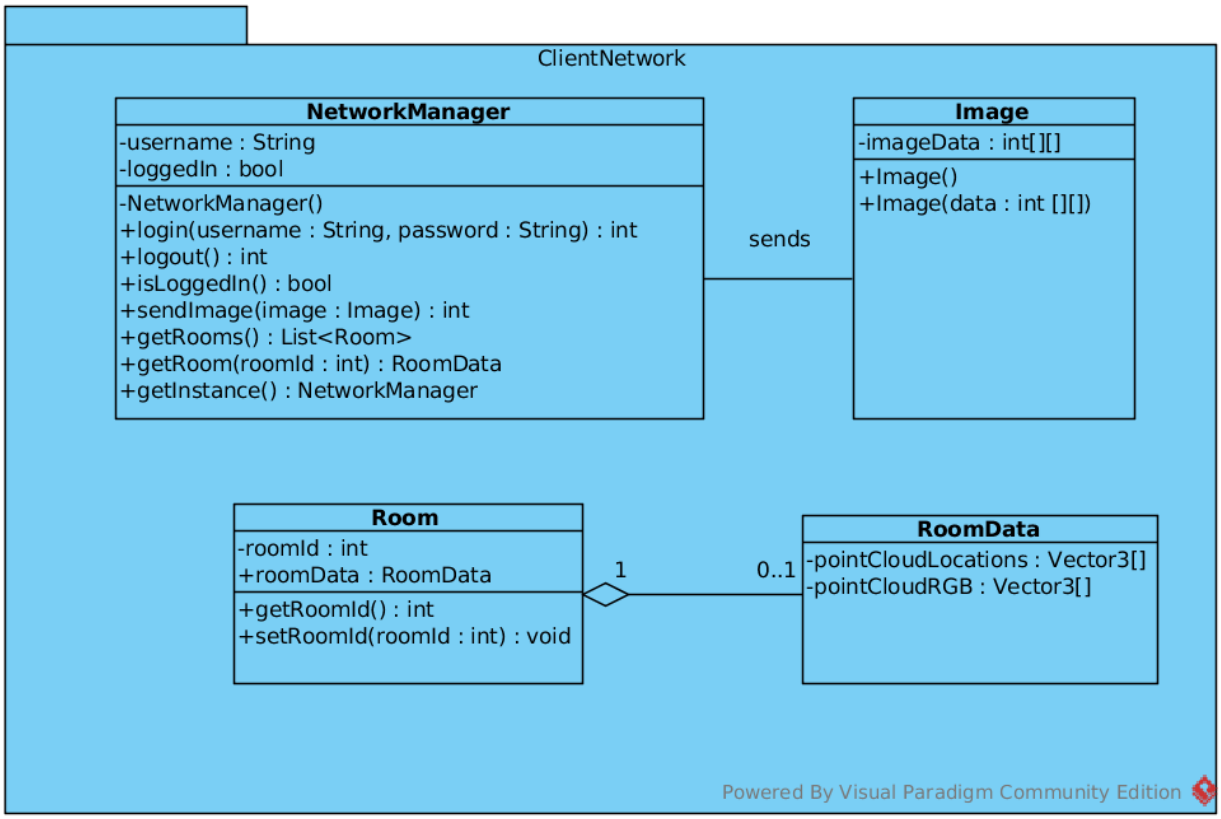


Figure 9: Client Network Manager package

4 Final Status of the Project

4.1 Overview

We finished implementing 90% of the back-end of the application. This includes the following:

- COLMAP (which is a SFM pipeline)'s output, a database chunk, is translated into text files for the server to read the SFM output more easily.
- Implementation of the feature matching code that is used for tracking the camera location. This is a 2d-tracking that does not take the depth into account (later utilized with PNP to take depth into account).
- Utilization of PNP that matches 2d and 3d tracking points and finds the position of the camera.
- Establishment of the connection between main server and a server that is implemented in Unity. This is the connection that receives client's frames to server to process them and sends the camera information to the main server.
- Implementation of the optimization code that searches only the points in the perspective (FOV) of the camera so that it searches smaller space rather than the whole space (brute-force). This improves the running time as well as the accuracy of the feature point matching.

The front-end of the application is still in progress. Following are the progress that has been made in front-end portion of the project:

- Successfully running the Unity application in an Android device, which includes forming the dependencies between Unity, Android and Java libraries.
- Utilization of the mobile device's camera so that the features of the frames that feed through the camera are extracted.
- Showing a 3d object in an augmented environment and several experiments regarding how 3d object's position changes with respect to the camera's location.

Nonetheless, we do not have the user-friendly and intuitive user interface that we designed and hoped for yet. We are currently working on improvements in UI as well as the visual of the 3d object so that we make sure that it looks seamless in the camera's texture.

According to the current state of the project, below are how the activities and actions are handled in server and client side respectively:

4.2 Flow of Work - Server

In the following figure, flow of work for the server-side is described.

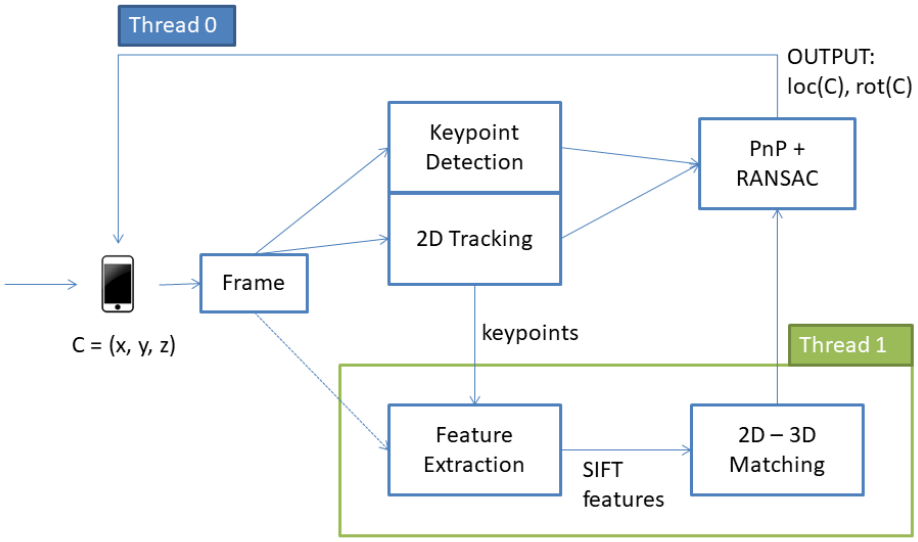


Figure 10: Flow of Work in Server-side

The camera of the mobile device is up and running, i.e. the stream of images from the Android Webcam Texture is ready whenever requested. For each of the frames that is sent to server-side, first of all, SIFT keypoint detection is performed. Over these keypoints, as more frames come, Lucas-Kanade Algorithm [15] is performed to accomplish 2D-tracking of these points. When a point is unable to be tracked, then that point is removed from the list of the tracked points. After that, these keypoints as well as the current frame itself is sent to SIFT Algorithm to extract features. Later, the output, i.e. SIFT keypoints and descriptors, are sent for matching 2D image points and 3D coordinates in the space. The last two actions are, as can be seen in Figure 10, done in a different thread due to their computational cost. The matched 2D and 3D points are sent to PnP algorithm to find the camera's current pose and orientation. RANSAC is applied with PnP in this step to smooth the output of the process, by removing outlier data. The camera's location and rotation information is sent back to the mobile device back as the final step.

[15] Lucas-Kanade Algorithm: `http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/tutorials/Lucas-Kanade2.pdf`

4.3 Flow of Work - Client

In the following figure, flow of work for the client-side is described.

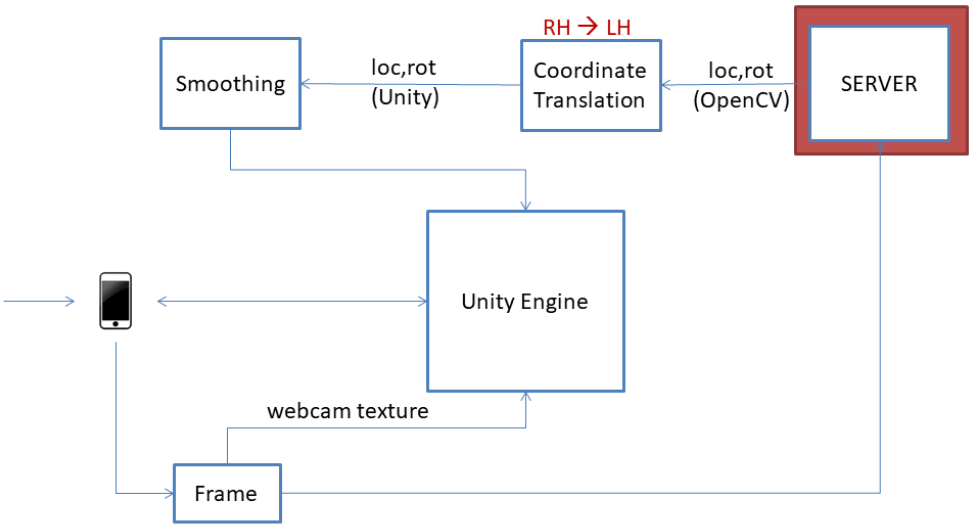


Figure 11: Flow of Work in Client-side

For the client-side, first of all, it is assumed that the server-side is sending location and rotation information to the client-side. However, the location and rotation information that server-side sends is in a Right-Handed Coordinate System (OpenCV's) whereas the client-side application is written in Unity which is in Left-Handed Coordinate System. Therefore, there needs to be a coordinate translation right after the client receives the information. After this translation is done, Slerp [16] is applied to further smooth the location and rotation data before it enters the Unity Engine and used in the client-side application. The client-side application, so far, has an animation model Unity-Chan [17], a humanoid that has several preset animations for developers to utilize, that responds to touching-tapping the screen (by jumping, saluting etc.); and it is placed in the augmented space manually (for now) by entering the exact coordinates for her to be located.

[16] Spherical Linear Interpolation - URL: `https://docs.unity3d.com/ScriptReference/Quaternion.Slerp.html`

[17] UnityChan - `http://unity-chan.com/`

5 Sample Output

Here are snapshots of a sample output video of DepthCube. Notice how the 3D animation model stays almost exactly the same place despite the changes in camera's location and orientation over time.

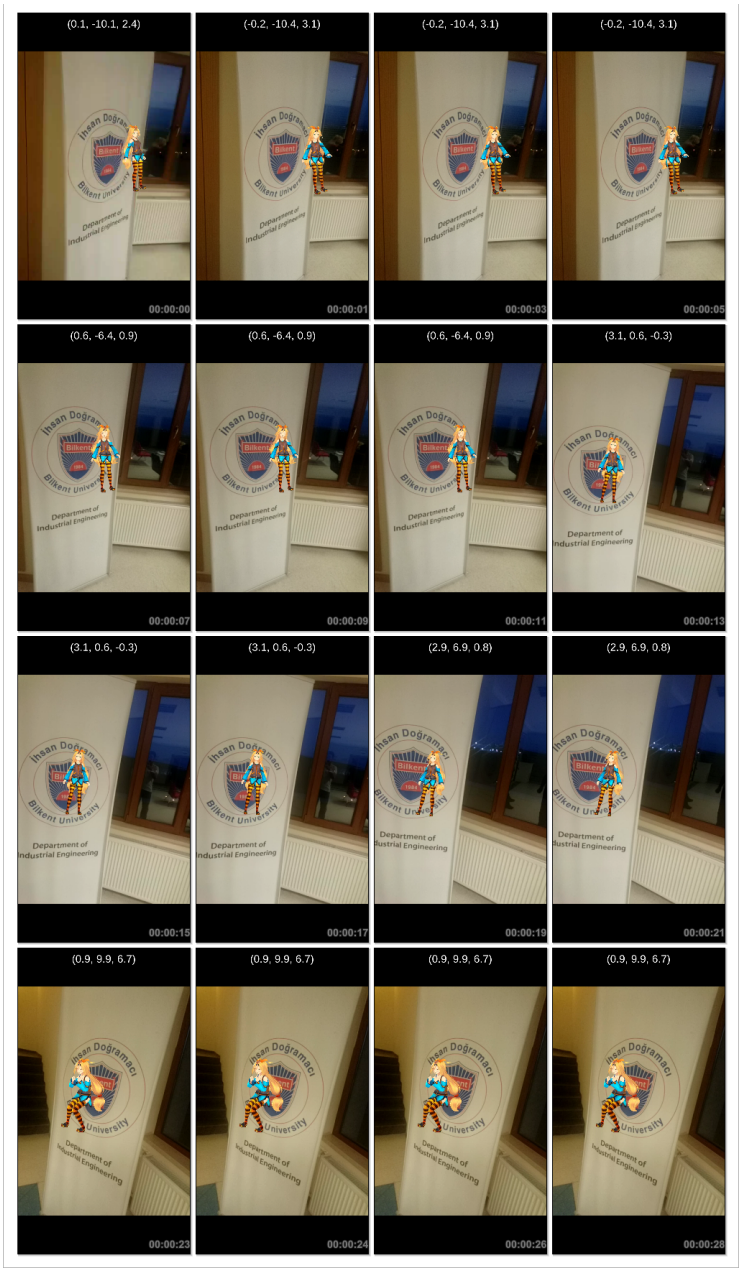


Figure 12: Demo1 - Video shot in 3rd Floor of EA Building in Bilkent University. 16 snapshots over 28 seconds.

6 Impact of the Engineering Solution

- **Global Impact:** Every contemporary mobile device in the world has a monocular camera at the back of their device. Making SFM and localization work in monocular cameras leads to new horizons in virtual and augmented reality; as it will allow mobile devices to be capable of applications that were once only available for binocular and multiocular settings. Before DepthCube (e.g. in PokemonGO), the augmented images or animations would not stay where they were first seen, although they are seemingly static and they should not change their position with respect to the actual reality. We attempted to improve this, and it will have a huge global impact if this problem is solved completely. Secondly, perhaps more importantly, the localization can be used in open-spaces. For instance, in huge shopping malls, it is a problem for people to find the store they want to go. If localization problem is solved, then the user can just take out his/her mobile phone, scan his/her surroundings and find out where exactly s/he is in a shopping mall. Not only can s/he locate herself, but also s/he can have directions to wherever s/he want to go, using path finding algorithms from there. This is a very important, highly functional and world-widely utilizable use-case.

- **Economic Impact:** Augmented Reality is an expensive entertainment because it requires extra equipments (i.e. additional devices or perks that attach to devices) for it to work. VR-Headsets such as Oculus Rift, Sulon Cortex or Morpheus are essential for most of the VR/AR-based games and other applications. DepthCube does not require anyone to buy such devices and incur cost on the users. Your mobile device is the only thing you need; which makes DepthCube a cheaper and budget-friendly alternative versus the rest of the VR/AR based applications.

- **Social Impact:** DepthCube provides a framework for AR-gaming. The games that are developed using DepthCube's framework can very well be multiplayer games. Hence, the players that are enjoying their time in a game (that used DepthCube API) while playing with or versus their friends are the embodiment of the social impact that our project can make.

- **Environmental Impact:** DepthCube is a mobile application and thus, it does not have a *direct* impact on the environment.

7 Related Contemporary Issues

There have been great improvements in Augmented Reality technologies in the recent years. Google's Project Tango, Amazon's VR Mask, Facebook's Oculus VR are all example projects and divisions which shows that the most successful IT companies in the world have been interested in VR/AR business. These divisions and projects are all formed after the start of 2010s, which shows that the area is still taking its baby steps, despite the fact that there is a strong desire to progress.

The problematic issue about the AR/VR masks is, first of all, their price. They are expensive and not worthwhile for a regular user; as there are only few applications that a user can make use of that mask. From an economical standpoint, the marginal benefit that the mask can provide to the user is proportional to the number of applications that exists, which is a low number. A framework that uses no such masks and requires no further equipment is more safe and reachable by the users. If the user did not like the application, s/he can just delete the application in DepthCube; whereas the other applications that require VR-masks have inflicted the mask's cost on the user permanently.

8 Used Tools and Technologies

8.1 Utility Tools

- **GitHub:** Git is used as a version control system. The project can be found at: `https://github.com/semihgunel/depth_cube`.
- **Unity3D Bundle:** Unity comes with its developer's bundle, which includes Unity's core functionality as well as an IDE called Unity Studio. In addition, to run and edit the scripts that are used in the scenes of Unity, Microsoft Visual Studio is installed within this developer's pack.
- **Android Bundle:** Android is installed with its bundle which has Android SDK, an IDE called Android Studio which is based on *IntelliJ*.
- **cURL:** cURL (also known as libcurl) is a client-side URL transfer library. It is used in communication during localization process between the client and the server.
- **Docker:** Docker is a technology that allows Linux servers to save and migrate server configurations. It packages applications as containers, allowing them to be deployed easily in any version of the Linux.

8.2 Third Party Libraries and Frameworks

- **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 weaknesses- are 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.

8.3 Internet Resources

- **StackOverflow.Com:** StackOverflow is the world-renowned website of programming Q/As. The questions that are asked in StackOverflow can vary from 'why is my code not working' to questions on the most subtle and delicate differences in the behaviour of certain methods or classes. Like any other project, StackOverflow is made great use of.

- **OpenCV Answers and Docs:** OpenCV has a stackoverflow-like website where users post their questions for other people to answer. It can be found at `http://answers.opencv.org/questions/`. Additionally, the documentation of OpenCV's packages are present at: `http://docs.opencv.org/3.1.0/modules.html`.

- **Unity Tutorials and Docs:** Unity is well-known for its user-friendliness. It is frequently advertised by its newbie-welcoming tutorials. These tutorials can be found at `https://unity3d.com/learn/tutorials/`. Also, the user manual and documentation of the Unity classes are documented at `https://docs.unity3d.com/Manual/index.html`.

9 Software / Hardware System

DepthCube has a Unity-based Android application in the client side and a cloud-based server in the server side. DepthCube mobile application can be installed on any Android-based mobile device with Android version 7.0.0 or higher. The server is a DigitalOcean droplet[18] where the necessary back-end libraries are installed. Unity that is used in client side is of version 5.6.0. For versions that come before 5.6.0, some compilation errors might arise due to the assets that are used in the client-side Unity application. For the demonstration purposes, the server can be one of the available local computers around the demo area; to ensure that the server-side processes are not affected by possible poor network conditions.

10 Conclusion

DepthCube is a mobile application that utilizes state-of-the-art computer vision algorithms to solve SFM and localization problems for monocular camera. SFM and localization are still open-problems in the field of computer vision, i.e. there is not a definite way to solve these yet. However, our framework attempts this problem and has some promising results. With further improvements, DepthCube has the potential to have a serious impact in the AR field in economical, social as well as global sense.

[18]DigitalOcean - Cloud Infrastructure Provider. `digitalocean.com`

11 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/`