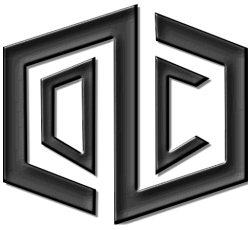
CS 491/2 Senior Design Project

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

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.

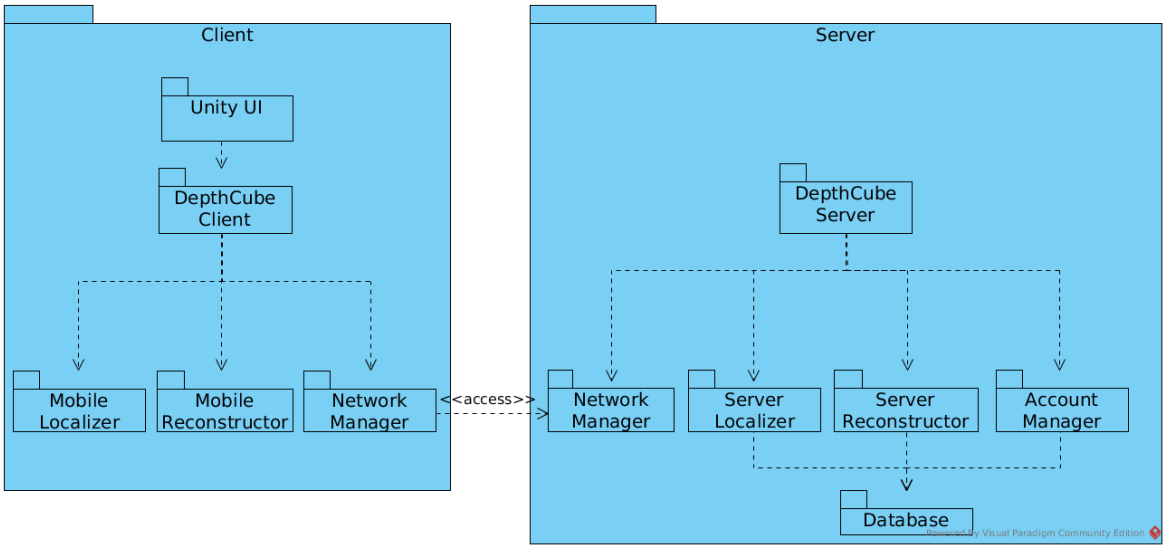


Figure 1: Package Diagram.

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.

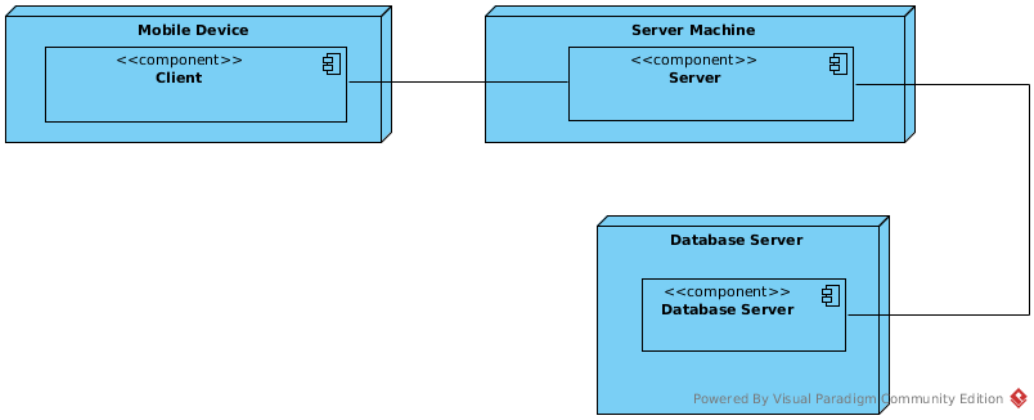


Figure 2: Deployment Diagram.

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

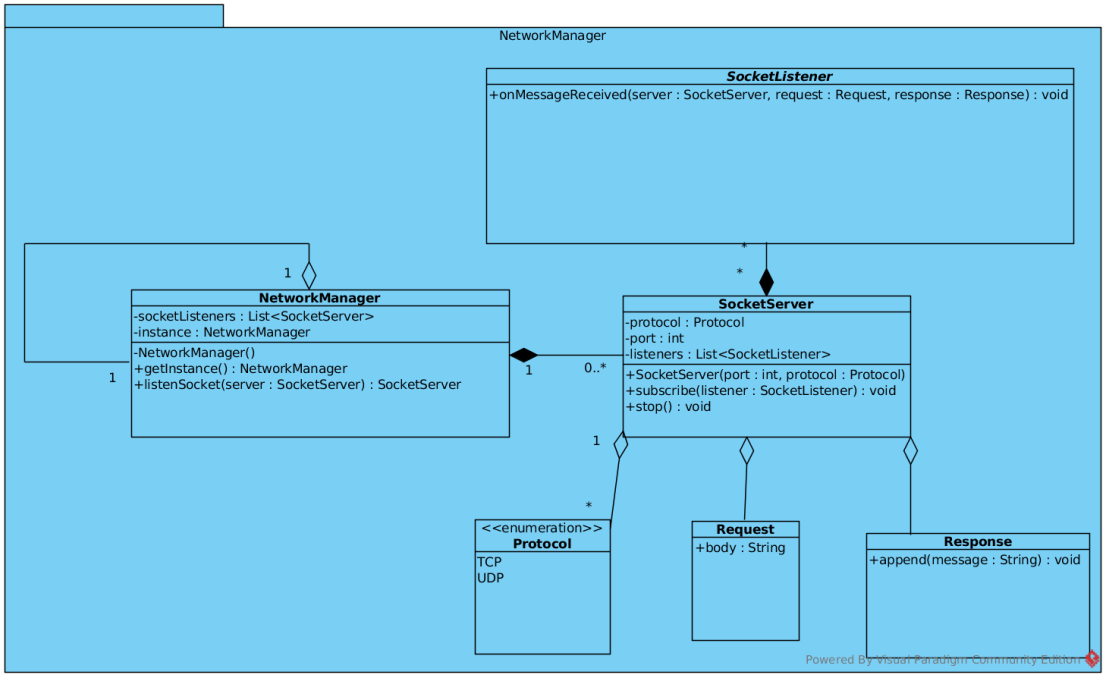


Figure 3: Network Manager Package.

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.

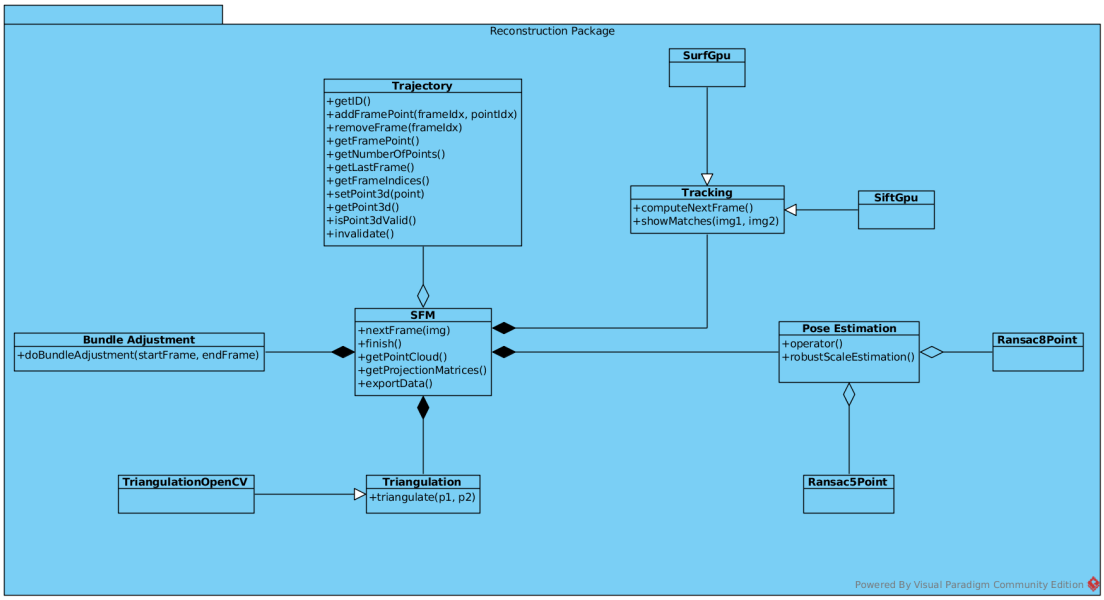


Figure 4: Reconstruction package

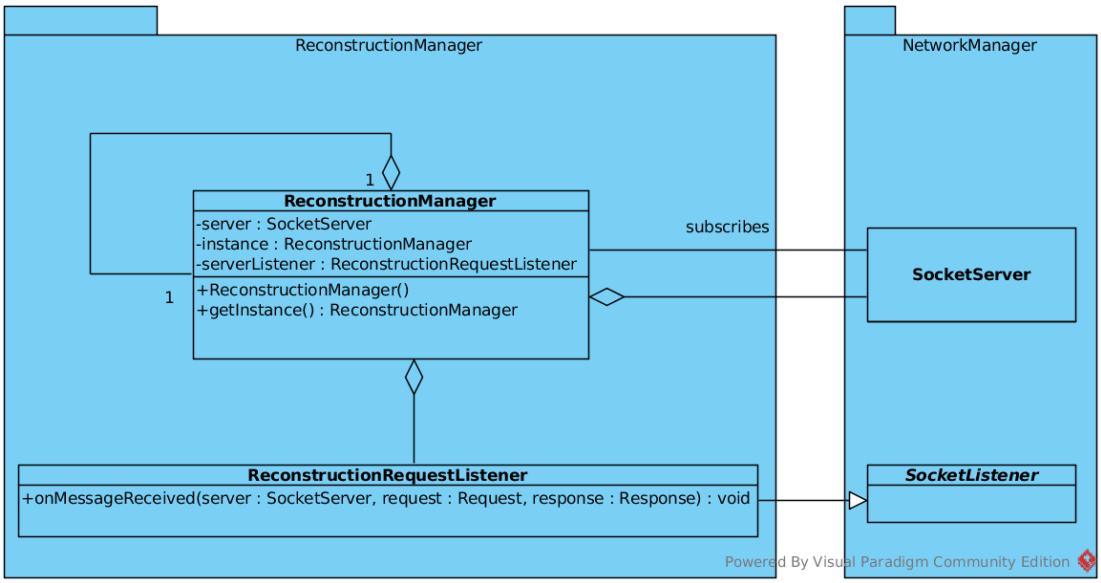


Figure 5: Reconstruction Manager with its relation to Network Manager

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.

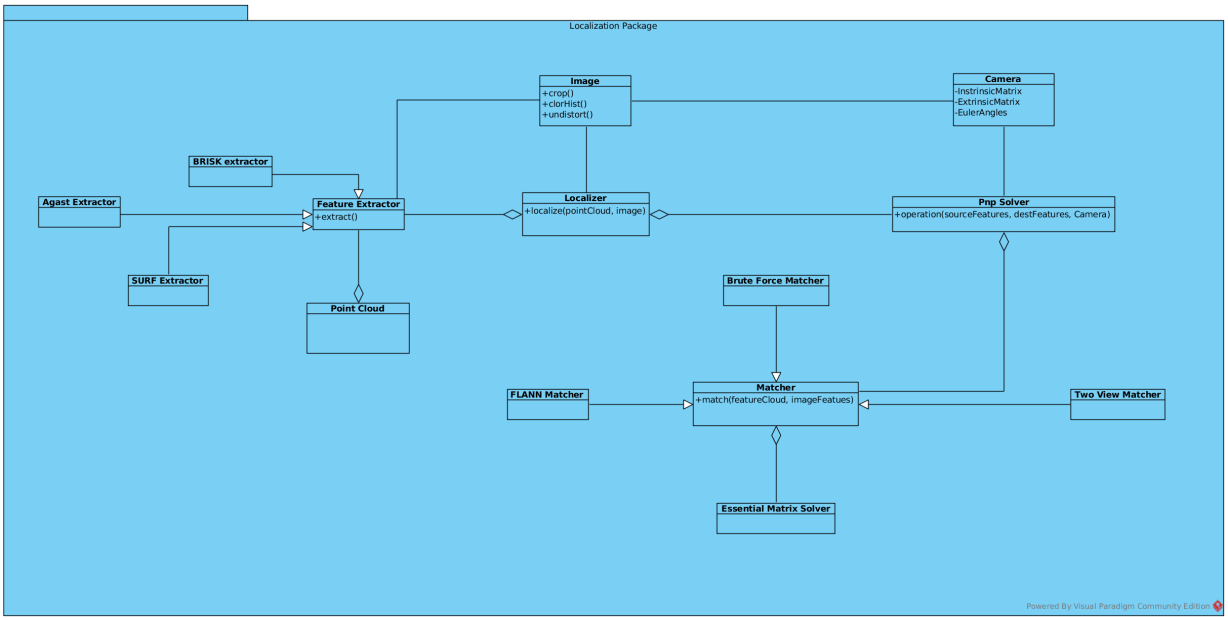


Figure 6: Localization package

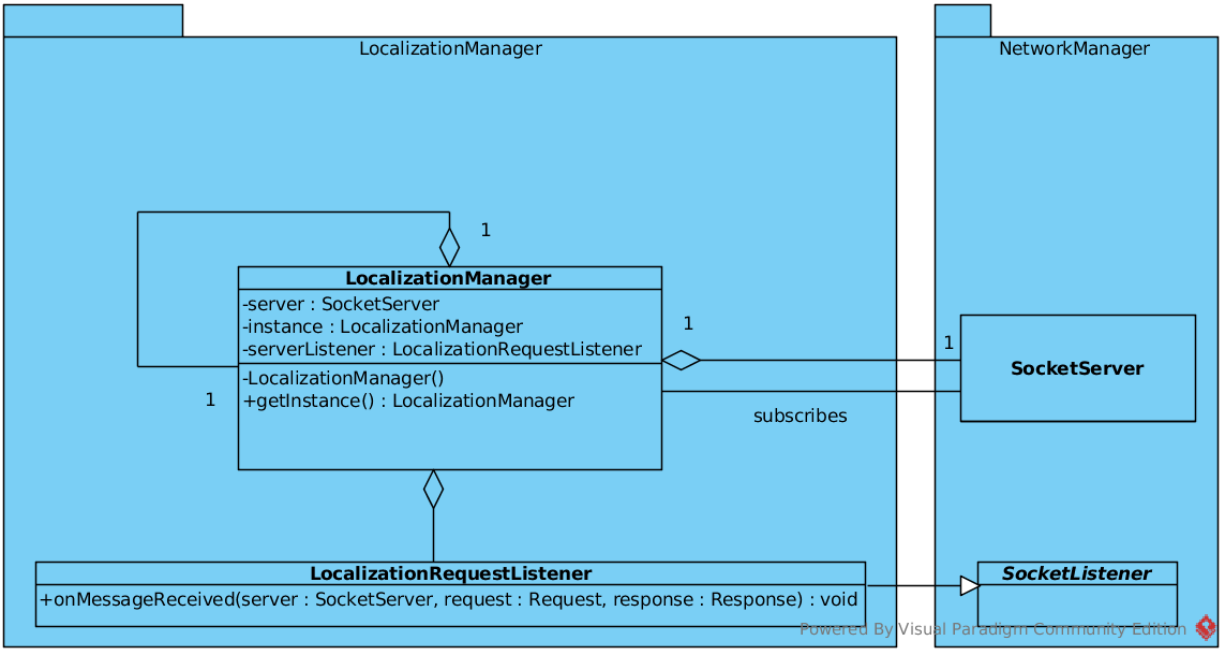


Figure 7: Localization Manager with its relation to Network Manager

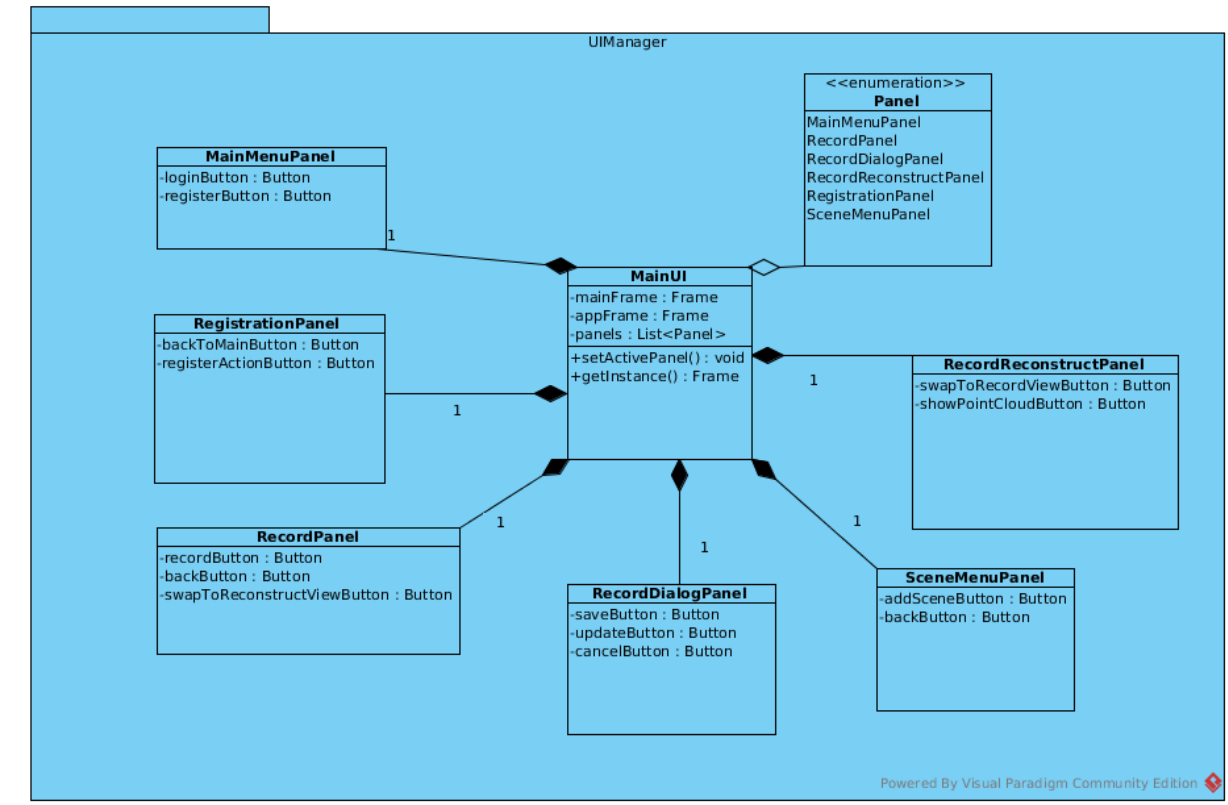


Figure 8: UIManager package

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.

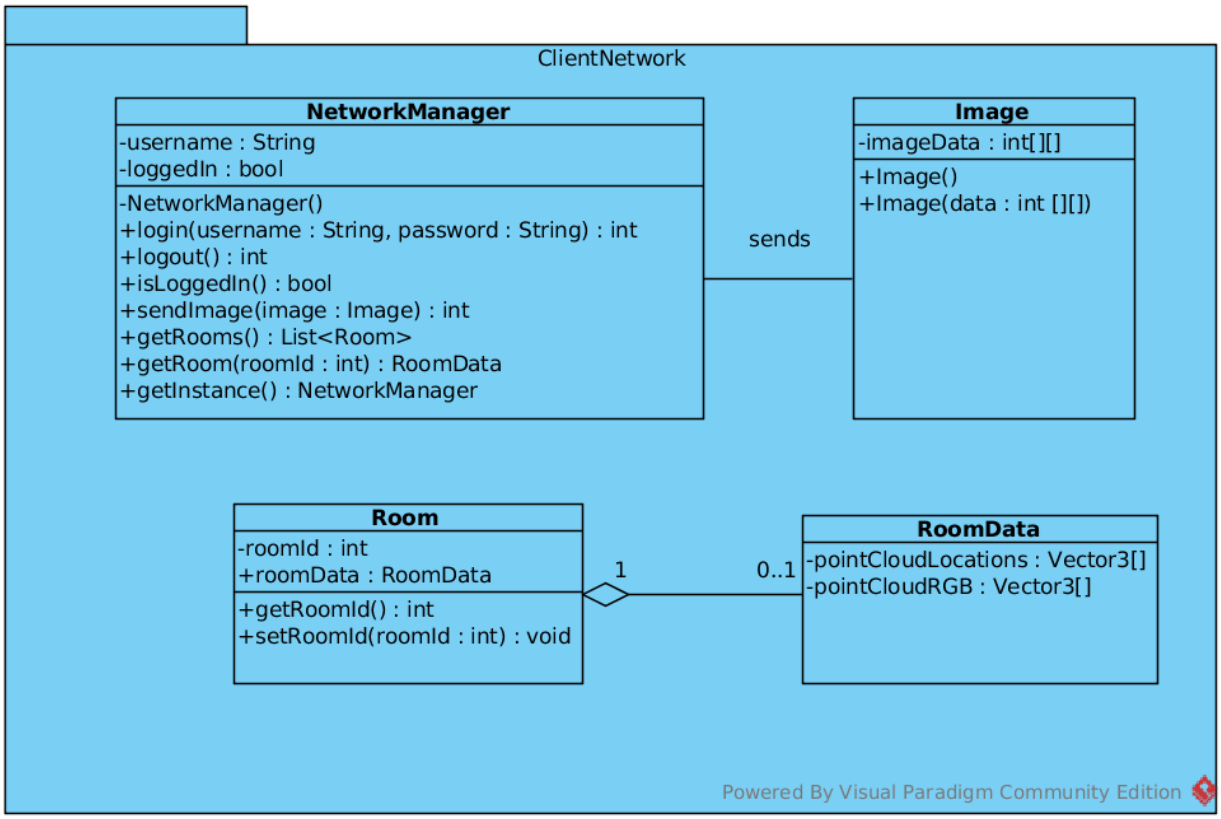


Figure 9: Client Network Manager package

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.

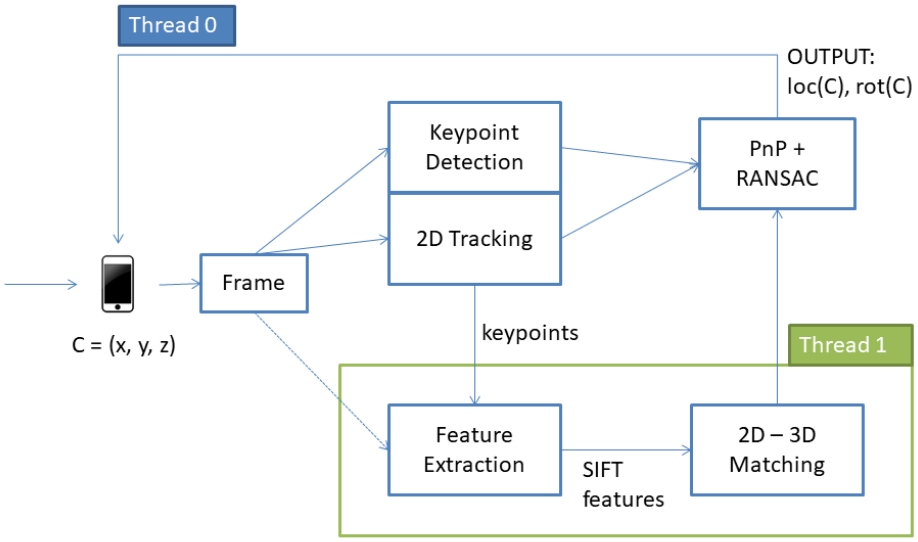


Figure 10: Flow of Work in Server-side

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.

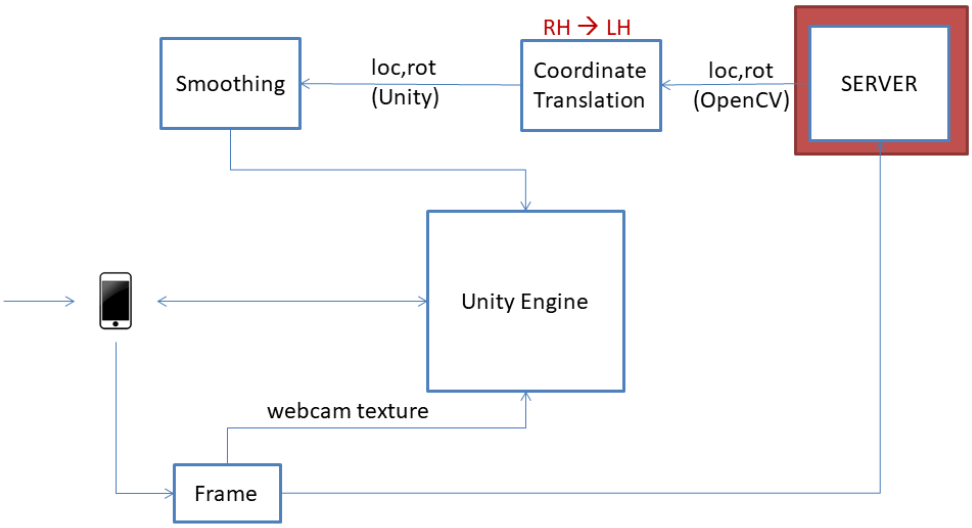


Figure 11: Flow of Work in Client-side

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.

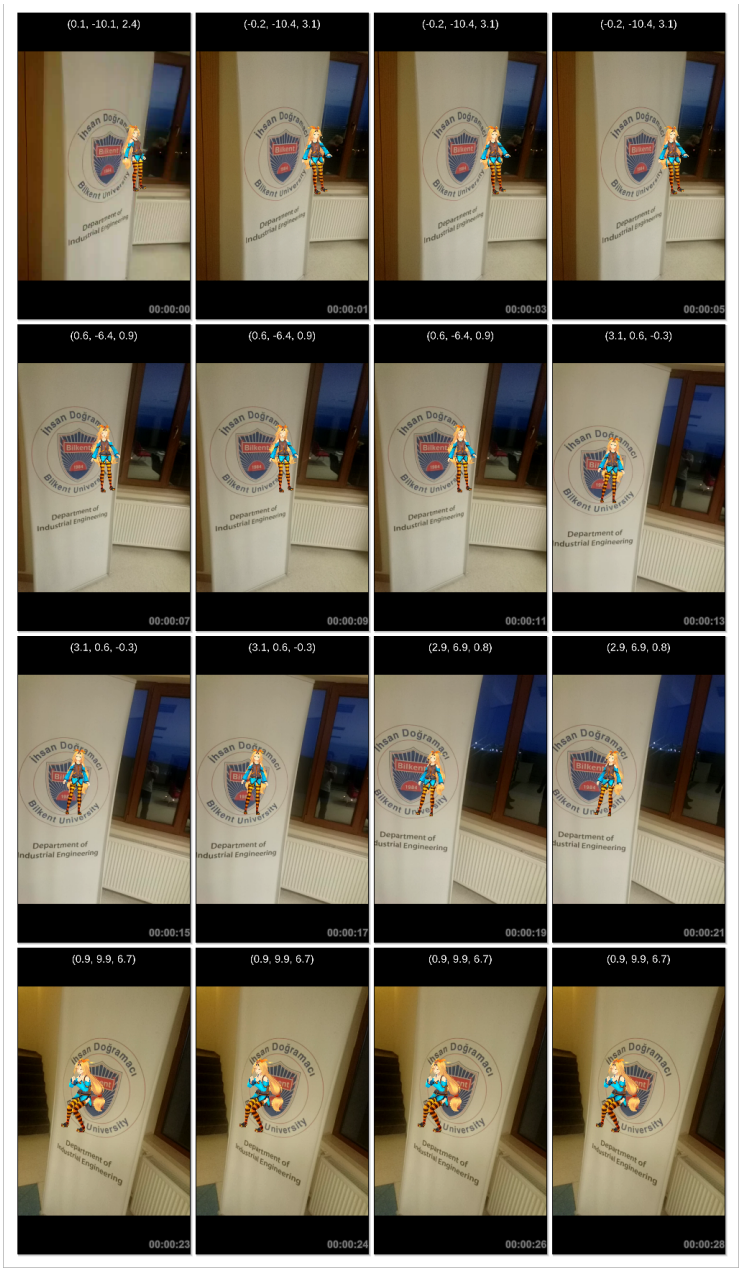


Figure 12: Demo1 - Video shot in 3rd Floor of EA Building in Bilkent University. 16 snapshots over 28 seconds.

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/`