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

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.

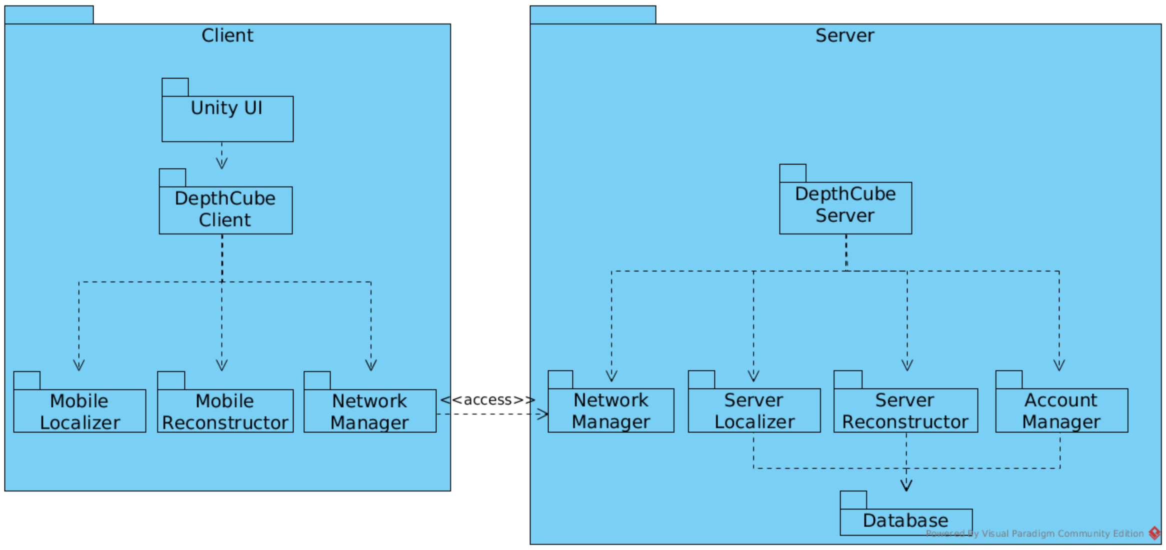


Figure 1: Package Diagram.

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.

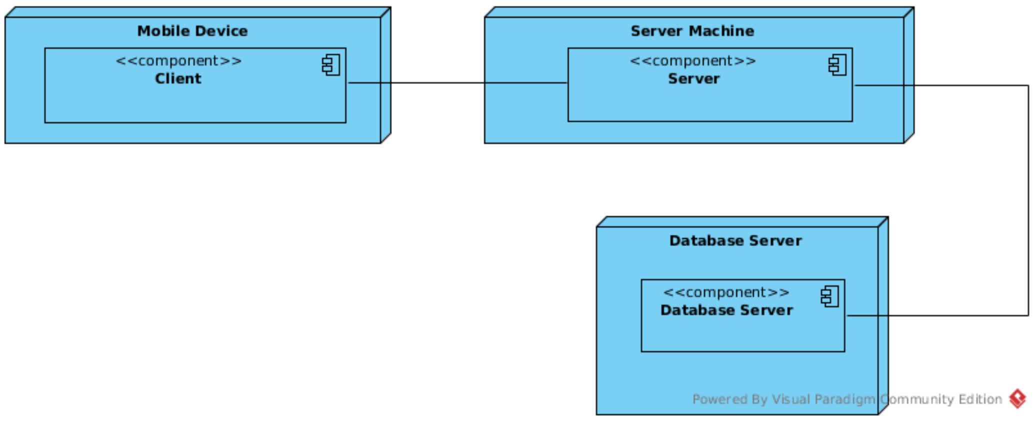


Figure 2: Deployment Diagram.

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 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.

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]

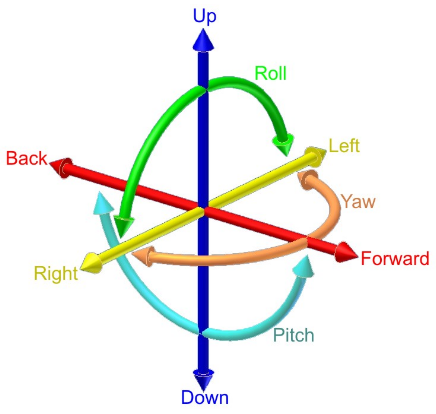
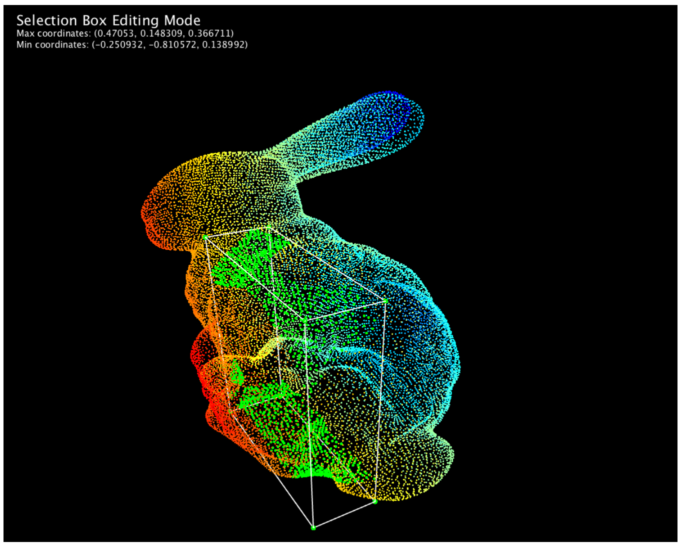


Figure 3: Six Degrees of Freedom.

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



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

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.

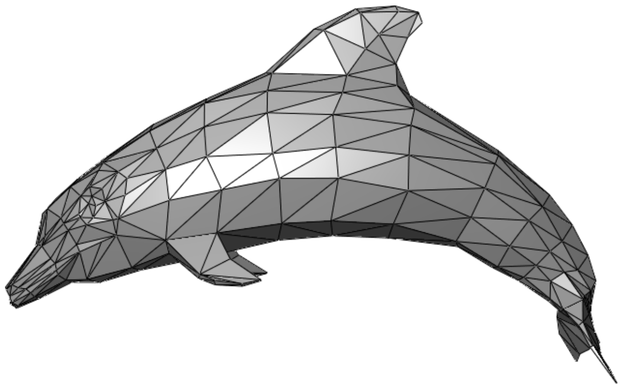


Figure 5: A triangle mesh representing a dolphin.

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/`