Bilkent University

Department of Computer Engineering

Senior Design Project

MoveIt, Indoor Manipulation : 3D Semantic Reconstruction, Display and Manipulation System

Low-Level Design Report

Barış Can

Faruk Oruç

Mert Soydinç

Pınar Ayaz

Ünsal Öztürk

Supervisor: Associate Professor Selim Aksoy, Ph.D.

Jury Members: Assistant Professor Shervin Rahimzadeh Arashloo,Ph.D.

Assistant Professor Hamdi Dibeklioğlu, Ph.D.

Low-Level Design Report

February 17, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

Low-Level Design Report

MoveIt, Indoor Manipulation : 3D Semantic Reconstruction, Display and Manipulation System

1. Introduction

The popularity of smartphones rises daily. While they have many uses, it can be argued that taking and sharing photos are among the most widespread uses of today's incredibly powerful smartphones. Photos capture a moment in nature or an indoor scene, and they all have visual and spatial information of the objects contained within them. It is possible to extract this information, and more information can be extracted as one adds additional angles and positions from which the original scene is viewed. The amount of information one can extract is directly proportional to the extra number of angles and positions, and the quality of these images. This information then may be used to model the environment captured by these images in 3D.

MoveIt: Indoor Manipulation aims to bring our homes into virtual reality by recreating 3D indoor scenes using a smartphone camera. With MoveIt, a quick scan of a room will bring the objects contained within this room into VR where they can be freely moved, interacted with, and manipulated. It is also a helpful tool that enables the user's aesthetic ability by allowing them to redecorate their room with ease while also providing functional help such as object boundary and collision detection. This 3D recreation of the room will be fully visualized in a VR environment that enables the user to freely move, scale, and rotate objects in the 3D reconstruction of the room using a VR Headset.

In this report, design trade-offs, engineering standards, packages and class interfaces will be discussed.

1.1. Object Design Trade-Offs

The following are the trade-offs of the system.

1.1.1. Scalability vs Cost

As the number of users increases, the load on the system as well which might cause high latency within the current system. The system can be scaled to serve a high number of users if the access request handling is stable and the server machine is powerful enough to handle all the simultaneous requests. Providing a powerful server machine comes with the trade-off that the cost of upgrading the server machine goes up.

1.1.2. Usability vs Functionality

The application will be available on several devices such as mobile, VR and computer, and the users will need to interact with all of the devices for the full experience. In order to enhance the user experience, we will make the interfaces as easy as possible. We will try to balance usability and functionality since user experience is of utmost importance and functionality does not mean much if the user cannot use the said functionality. There might exist some functionality that requires complex interactions from the user. In this case, we will try to provide guidance to the user in the form of tutorials or detailed instructions.

1.1.3. Accuracy vs Speed

Since segmentation and reconstruction operations need substantial computational power, the accuracy and the quality of the reconstructed scene heavily depend on the amount of time spent on the computations. The 3D reconstruction algorithm is able to produce an inferior reconstruction if the time is limited but given enough time, it is able to produce much more accurate results. We will try to balance the accuracy of these operations such that the user is satisfied with the outcome and the time spent waiting is not extremely high.

1.1.4. Functionality vs Complexity

Since MoveIt will run with many devices such as mobile, VR and computer. We will develop interfaces and functionalities for each environment separately. For instance, we will develop a workshop functionality for the computer and VR so that the user will be able to add any furniture to their rooms from the furniture database. Also, the user would be able to scan individual objects and add them to the furniture database which would enable them to access them during the user experience. Therefore, all components' features should work synchronously and this will increase the complexity of our application.

1.2. Interface Documentation Guidelines

Class Name	
Class Description	
Attributes	
Attribute Name	Attribute Description
Methods	
Method Name	Method Description

1.3. Engineering Standards

The following are the engineering standards followed during the project.

1.3.1. Unified Modeling Language (UML)

We generated several diagrams such as sequence, decomposition and the class diagram. In all these diagrams, we used Unified Modeling Language guidelines and standards. We also used UML to describe the diagrams like use-case and class interfaces [2].

1.3.2. IEEE

We used IEEE standards while both citing and referencing in all our reports [1].

1.4. Definitions, acronyms, and abbreviations

Semantic Segmentation: The operation of linking each pixel of an image to a class label. In our project, this corresponds to linking the pixels of the detected furniture in the image to the correct label of that furniture.

3D Reconstruction: The process of capturing the shape and appearance of real objects. In our project, we are trying to reconstruct the geometry of captured indoor scenes in 3D accurately [3][4].

Semantic Label: The label assigned to the pixels in an image after performing semantic segmentation. For example, the label 'chair' assigned to the pixels in the image that belong to a chair.

Geometric 3D Mesh: The structural build of a 3D model consisting of polygons. 3D meshes use reference points in X, Y and Z axes to define shapes with height, width and depth. In our project, these will be used to construct a 3D model for the detected objects from the indoor scenes.

Texture Analysis: Refers to the characterization of regions in an image by their texture content.

VR Scene Rendering: Producing a view of a scene for humans in a VR environment that aims to be as close to the actual scene in reality as possible by using the 3D data. This will be performed to display the 3D reconstructed indoor scenes in our project[5].

2. Packages

Below, packages of MoveIt will be discussed.

2.1. Client

- Client Wrapper: Provides an abstraction for Client operations. Subclasses implement hardware specific code for file I/O, UI rendering, controls, control polling, and network interfaces.
- PC Wrapper: Provides hardware specific code for MoveIt operations on PC.
- Mobile Wrapper: Provides hardware specific code for MoveIt operations on mobile devices, in particular Android.
- VR Wrapper: Provides hardware specific code for MoveIt operations on VR headsets.
- Client UI: This component is the parent component of all the user interface components.
- PC UI: Provides the user interface functionality for the PC client.
- Mobile UI: Provides the user interface functionality for the Mobile application client.
- VR UI: Provides the user interface functionality for the VR client.
- Video Sampler: Samples the uploaded videos.
- Video: Contains the actual video data uploaded by the users.
- Video Metadata: Handles all the metadata related to the uploaded user videos such as their names, resolutions, dimensions etc.

2.2. Renderer

- Light Manager: Contains the light map of the scene. Used to add light sources to the scene or remove light sources from the scene.
- Shader Manager: Contains the shader map and manages the shader programs for the scene.

- Light: Represents the light in a given scene. It contains position, intensity and color information.
- Camera Manager: Manages the interactions between the camera and the scene.
- Camera: Used to view the scene. It contains the camera position, aperture size, focal length and its projection matrix.
- Modify Command Builder: Used to easily generate instances of ModifyCommand with a specific command.
- Modify Command: A wrapper object encapsulating operations on the geometry of the scene or meshes. Used to rotate, scale, slice, replicate, shear, translate objects.
- Scene: Representation of a real scene. It contains an object manager to keep track of the object in it. It can also modify the objects in the scene, light and camera settings. It also contains a render function.
- Object Manager: Contains all the objects. Used to pass commands to objects and store the objects.
- Object: Representation of a real object. It contains the object mesh and texture information.

2.3. Reconstruction

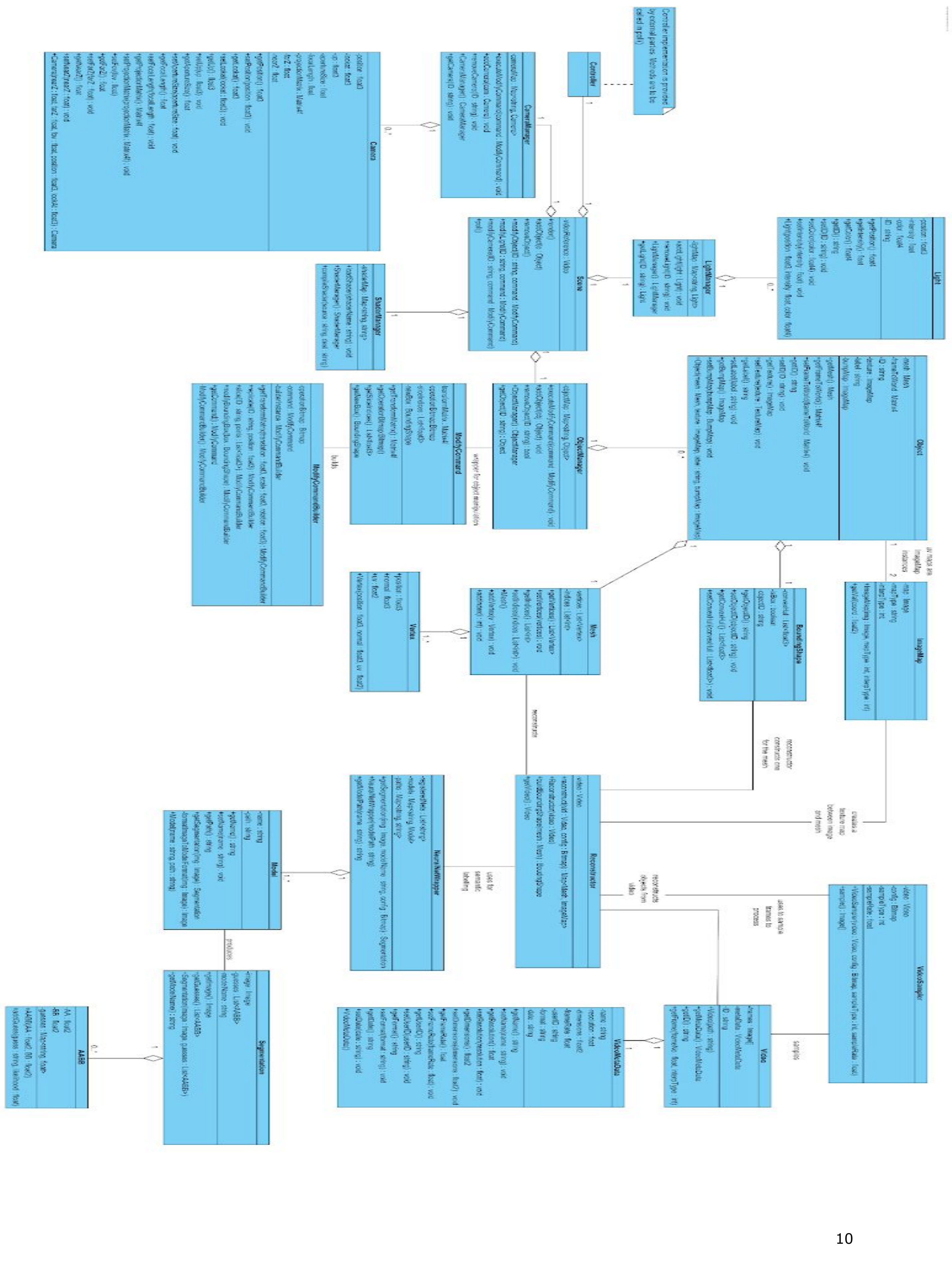
- ImageMap: ImageMap is an image containing the texture map, displacement map, bump map, or any kind of parametric map that can be represented as a 2D image.
- BoundingShape: It contains the bounding shape of an object. Used to determine collisions with other objects.
- Mesh: Provides an application specific representation of a 3D mesh using index and vertex sets.
- Vertex: Provides an application specific representation of a 3D vertex in Cartesian coordinates.

- NeuralNetWrapper: Used to communicate between the neural net model and the reconstructor.
- Model: Neural net model that produces the segmentations.
- Segmentation: Output of the model. It contains the guesses produced by the model for a given image.
- AABB: Represents the boundaries of the model’s guesses.

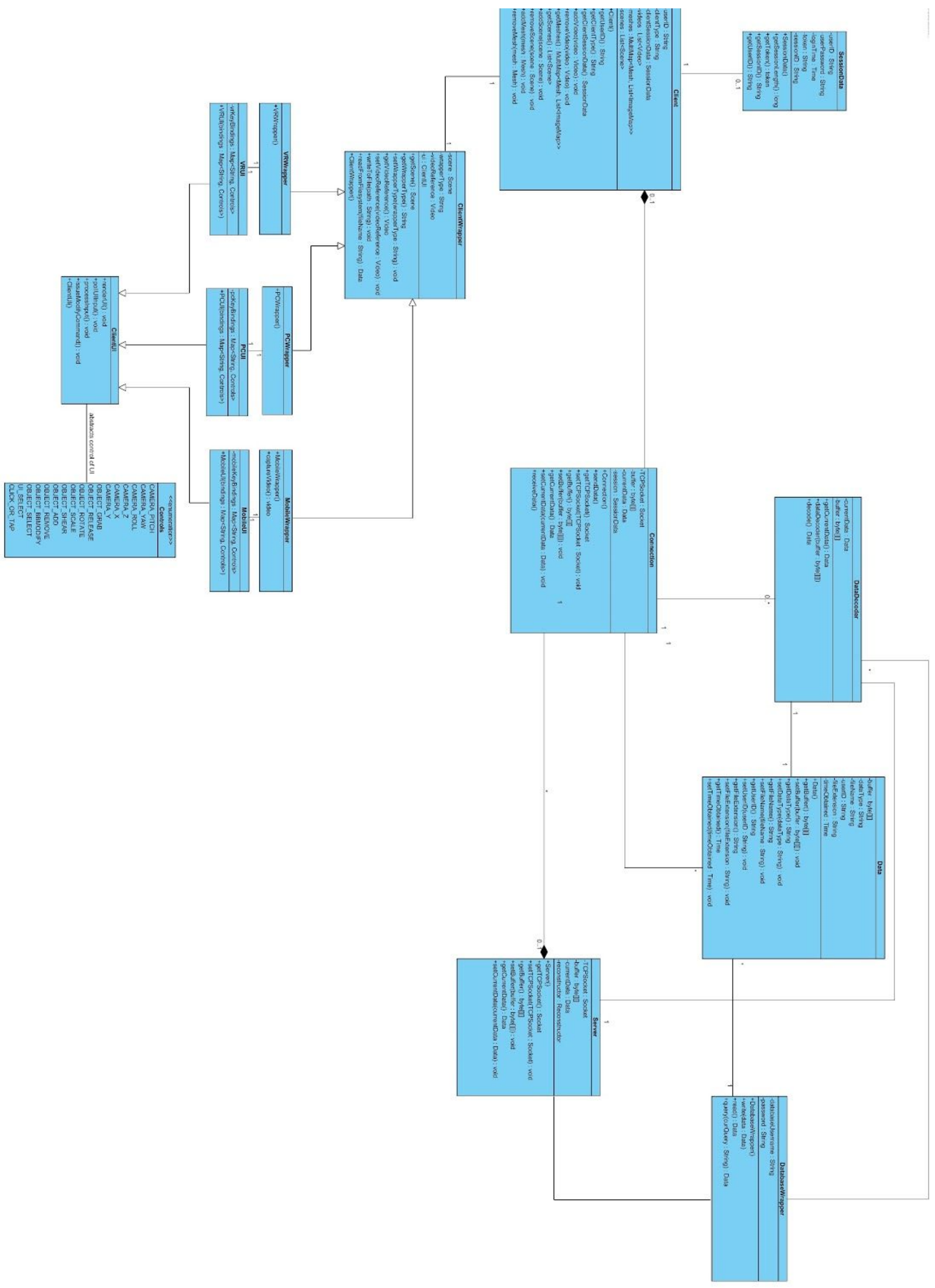
2.4. Server

- Connection: Represents the connection between a Server and a Client object. Establishes the connection between these two using a Socket. Allows the transfer of raw bytes of data.
- Server: Main server of the application. It is used as a bridge between the reconstructor, database and client.
- DatabaseWrapper: Bridge between the server and the database. Handles the data connection between them.
- DataDecoder: Decodes the raw bytes received through a connection. Has hard-coded instructions to decode application specific headers for particular objects and files. Produces instances of the Data class.
- Data: Provides a neat representation of data for application specific purposes. Has fields to determine the header to be used if the data is to be sent over a network, along with file type and metadata.

3. Class Interface



Controller
Controller implementation is provided by external parties. Methods are to be called in poll()
CameraManager
Camera
Scene
LightManager
Light
ShaderManager
ObjectManager
Object
wrapper for object manipulation
ModifyCommand
builds
ModifyCommandBuilder
Mesh
Vertex
BoundingShape
ImageMap
reconstructs
reconstructor constructs one for the mesh
creates a texture map between image and mesh
Reconstructor
reconstructs objects from video
uses to sample frames to process
uses for semantic labeling
NeuralNetWrapper
Model
produces
Segmentation
AABB
VideoMetaData
Video
samples
VideoSampler



https://imgur.com/a/O0Duor2 Higher quality class diagrams

3.1. Client

3.1.1. View

ClientUI	
ClientUI provides a common abstraction for user interfaces over all platforms and operating systems the software runs on.	
Attributes	
-	-
Methods	
renderUI()	Renders the UI of the software.
pollUIInput()	Waits for user input on the user interface, converts this input into the common format used by the software.
processInput()	Processes the input obtained through processInput() and updates the user interface according to the modifications in the model.
issueModifyCommand()	Issues a ModifyCommand object for the manipulation and deformation of the meshes in the 3D scene.

VRUI	
Provides the user interface functionality for the VR client.	
Attributes	
Map<String, Controls> vrKeyBindings	Static hard-coded map for VR specific input format. Maps the controls provided by a VR headset to software specific control schemes.
Methods	
-	-

PCUI	
Provides the user interface functionality for the PC client.	
Attributes	
Map<String, Controls> pcKeyBindings	Static hard-coded map for VR specific input format. Maps the controls provided by peripheral devices (i.e. keyboard, mouse) to software specific control schemes.
Methods	
-	-

MobileUI	
Provides the user interface functionality for the Mobile application client.	
Attributes	
Map<String, Controls> mobileKeyBindings	Static hard-coded map for mobile device specific input format. Maps the controls provided by a mobile device to software specific control schemes.
Methods	
-	-

3.1.2. VideoManagement

VideoSampler	
Samples frames at desirable points from videos uploaded by the user.	
Attributes	
Video video	In memory reference to the object representing a video as a series of image frames.
Bitmap config	Determines the configuration for the sampler, i.e. whether the sampler will also perform spatial sampling along with temporal sampling upon initialization.
int sampleType	Determines the type of sampling used by the sampler object. Sampling types include uniform sampling, adaptive sampling, and importance sampling.
float sampleRate	Determines the temporal sampling rate for the sampler. While this variable is straightforward for

	uniform sampling, for other sampling types, increasing this parameter increases the number of samples taken, but not in a uniform manner.
Methods	
sample()	Produces a sampling of the provided video given the configuration of the sampler, and returns the samples as an array of images and timestamps.

Video	
Provides an implementation for the in-memory object representing a video as a set of image frames, unique video ID and related metadata.	
Attributes	
Image[] frames	Representation of the video as an array of images.
VideoMetaData metaData	In-memory representing the relevant metadata regarding several properties and attached information to the video.
string ID	Unique ID assigned to the video. Acts as primary key.
Methods	
get methods for all attributes	-

VideoMetaData	
Handles all the metadata related to the uploaded user videos such as their names, resolutions, dimensions etc.	
Attributes	
String name	Name of the video.
float resolution	Resolution of the video.
float2 dimensions	Dimensions of the video in width x height.
float frameRate	Number of frames per second.
String userID	User ID attached to the video (i.e. a handle to the user submitting the video)
String format	Format of the video in string format.
String date	Date on which the video is captured.
Methods	
get and set methods for all attributes	-

3.1.3. Hardware Level Subsystem

ClientWrapper	
Provides an abstraction for Client operations. Subclasses implement hardware specific code for file I/O, UI rendering, controls, control polling, and network interfaces.	
Attributes	
Scene scene	Handle to the main interface of the renderer.
String wrapperType	Stores the type of the wrapper in string format (values include

	VRWrapper, PCWrapper, MobileWrapper)
Video videoReference	Handle to the in-memory representation of a video.
ClientUI ui	Reference to the user interface object for user interaction with the application and the Scene object.
Methods	
writeToFile(String path)	Writes the necessary information to recreate the 3D scene after the program has been terminated, i.e. serializes the in-memory object to a given file path.
readFromFilesystem(String fileName)	Reads the scene information stored in a file in binary or human-readable format and reconstructs the scene.
get methods for scene, wrapperType and videoReference and set methods for wrapperType and videoReference	-

PCWrapper	
Provides hardware specific code for MoveIt operations on PC.	
Attributes	
-	-
Methods	
-	-

VRWrapper	
Provides hardware specific code for MoveIt operations on VR headsets.	
Attributes	
-	-
Methods	
-	-

MobileWrapper	
Provides hardware specific code for MoveIt operations on mobile devices, in particular Android.	
Attributes	
-	-
Methods	
captureVideo()	Captures a new video from the mobile application.
playVideo()	Plays the captured video if it exists from the mobile app.

3.2. Renderer

3.2.1. Shading Subsystem

LightManager	
Contains the light map of the scene. Used to add light sources to the scene or remove light sources from the scene.	
Attributes	
Map lightMap	Provides a mapping between an abstract representation of a light object and a unique string identifier.
Methods	
addLight(Light light)	Inserts a light to the light map with a procedurally generated key.
removeLight(String ID)	Removes a light from the light map.
getLight(String ID)	Returns a Light object with the provided key.

ShaderManager	
Contains the shader map and manages the shader programs for the scene.	
Attributes	
Map shaderMap	Provides a mapping between shaders and their keys, where keys are determined arbitrarily as the name of the shader (e.g "phong_shader").
Methods	
loadShader(String shaderName)	Loads a shader program and runs the shader program via the shader map, given the name of the shader.
compileShader(String source, String dest)	Compiles a shader registered to the shader map. The parameters require

	the source directory of the file containing the shader program and the destination directory for the compiled shader program.

Light	
Represents the light in a given scene. It contains position, intensity and color information.	
Attributes	
float3 position	Position of the light in world coordinates.
float intensity	Intensity of the light in W/m^2.
float4 color	Color value of the light in RGBA form. The last channel is not used.
String ID	ID of the light.
Methods	
get methods for all attributes and set methods for ID, color and intensity	-

3.2.2. Camera Subsystem

CameraManager	
Manages the interactions between the camera and the scene.	
Attributes	
Map cameraMap	Provides a mapping between IDs and the handles of camera objects.
Methods	
executeModifyCommand(ModifyCommand command)	Executes a command concerning a camera object. The command can be

	the rotation or a translation of the camera.
addCamera(Camera camera)	Adds a camera to the mapping with an ID specified in the Camera object.
removeCamera(String ID)	Removes a camera with a given ID.
getCamera(String ID)	Returns a handle to the camera object with a given ID.

Camera	
Used to view the scene. It contains the camera position, aperture size, focal length and its projection matrix.	
Attributes	
float3 position	Defines the position of the camera in world coordinates.
float3 lookat	Defines the lookat vector.
float3 up	Defines the up vector of the camera.
float apertureSize	Defines the size of the aperture of the camera.
float focalLength	Defines the focal length of the camera.
Matrix4f projectionMatrix	Projection matrix of the camera, derived from other quantities.
float farZ	Defines the far depth cutoff for the render.
float nearZ	Defines the near depth cutoff for the render.
Methods	
get and set methods for all attributes	-

3.2.3. Control Subsystem

ModifyCommandBuilder	
Used to easily generate instances of ModifyCommand with a specific command.	
Attributes	
Bitmap operationBitmap	Bitmap representing the operations, the order of operations, and the allowed types of operations to be applied on a particular object in the scene.
ModifyCommand command	An abstraction for a command that modifies a given object.
ModifyCommandBuilder builderInstance	An instance of the ModifyCommandBuilder in accordance with the singleton pattern.
Methods	
getTransformMatrix(float3 translation, float3 scale, float3 rotation)	Produces a 4D matrix representing a linear transformation in homogeneous coordinates.
replicate(String ID, float3 position)	Replicates an object with a given ID at a provided position.
slice(String ID, List<float3> points)	Slices the mesh of an object with a given ID at the convex hull of the points provided in the parameter list.
modifyBoundingBox(BoundingShape box)	Modifies the bounding box of an object such that the new bounding box of the object is the one in the parameter list.
getCommand()	Returns the instance modify command currently associated with the builder.

ModifyCommand	
A wrapper object encapsulating operations on the geometry of the scene or meshes. Used to rotate, scale, slice, replicate, shear, translate objects.	
Attributes	
Matrix4f transformMatrix	Transformation matrix representing the linear transformation in 4D homogeneous coordinates.
Bitmap operationBitmap	Bitmap representing the operations, the order of operations, and the allowed types of operations to be applied on a particular object in the scene.
List<float3> sliceIndices	Indices through the convex hull of which the mesh is to be sliced.
BoundingShape newBox	The new bounding box of the object.
Methods	
get methods for all attributes	-

3.2.4. Object Subsystem

ObjectManager	
Contains all the objects. Used to pass commands to objects and store the objects.	
Attributes	
Map objectMap	Provides a mapping between the IDs and the handles to abstract representation of an object.
Methods	
executeModifyCommand(ModifyCommand command)	Executes a 'modify' command on a specified object queried through the

	object map.
addObject(Object obj)	Adds an object to the object map and the scene.
removeObject(String ID)	Removes an object from the object map and the scene.
getObject(String ID)	Returns the handle to an object with the specified ID through a query on the map.

Object	
Representation of a real object. It contains the object mesh and texture information.	
Attributes	
Mesh mesh	A handle to the in memory abstract representation of a 3D mesh.
Matrix4 frameToWorld	Stores the transformation matrix that transforms the geometry of the object from local to world coordinates.
String ID	Unique ID for the object.
ImageMap texture	(u,v) Texture map for the object
String label	Semantic label for an object.
ImageMap bumpMap	An image representing the bump map for the object, if available.
Methods	
get methods for all attributes and set methods for frameToWord, ID, texture, label and bumpMap	-

Scene	
Representation of a real scene. It contains an object manager to keep track of the object in it. It can also modify the objects in the scene, light and camera settings. It also contains a render function.	
Attributes	
Video videoReference	Handle to the video for which the 3D reconstruction was done.
Methods	
render()	Renders a given scene configured with the camera, lights, and the objects.
addObject(Object o)	Adds an object to the scene by registering the necessary information to the object map (i.e. the handle to the in-memory representation of the object)
removeObject()	Removes a specified object from the scene, and deallocates the related memory.
modifyObject(String ID, ModifyCommand command)	Modifies an object with the specified ID according to the specified ModifyCommand instance.
modifyLight(String ID, ModifyCommand command)	Modifies a light object with the specified ID according to the specified ModifyCommand instance.
modifyCamera(String ID, ModifyCommand command)	Modifies a camera object with the specified ID according to the specified ModifyCommand instance.
poll()	Polls inputs for user interaction with the scene (e.g. camera and object movement).

3.3. Reconstruction

3.3.1. Geometric Processing

ImageMap	
ImageMap is an image containing the texture map, displacement map, bump map, or any kind of parametric map that can be represented as a 2D image.	
Attributes	
Image map	Contains either color or numerical information aimed at visually enhancing a mesh to which the ImageMap is attached. Acts as a lookup table for (u,v) texture, normal, displacement, or bump map.
String mapType	String storing the type of the map.
int interpType	Integer value denoting the type of interpolation performed by the lookup. Available interpolation types include bilinear interpolation, bicubic interpolation, and Lanczos interpolation.
Methods	
getVal(float2 coord)	Returns the value interpolated at the coordinates specified.

BoundingShape	
It contains the bounding shape of an object. Used to determine collisions with other objects.	
Attributes	
List<float3> convexHull	Stores the boundary of the bounding shape as a convex hull.

boolean isBox	Stores if the bounding shape is a box.
String objectID	ID of the current object.
Methods	
get and set methods for convexHull and objectID	-

Mesh	
Provides an application specific representation of a 3D mesh using index and vertex sets.	
Attributes	
List<Vertex> vertices	Stores the handles to Vertex objects that make up the mesh.
List<int> indices	Stores the indices of the connected vertices. The first three indices are considered to form the first triangle, the second triplet of indices are considered to form the second triangle and so on.
Methods	
addVertex()	Adds a new vertex to the vertices list.
addIndex()	Adds a new index to the indices list.
get and set methods for all attributes	-

Vertex	
Provides an application specific representation of a 3D vertex in Cartesian coordinates.	
Attributes	
float3 position	Stores the position of a vertex in the local coordinate frame.
float3 normal	Stores the normal at a vertex. May be overridden at render time by an appropriate parametric map.
float2 uv	The (u,v) coordinate of the vertex obtained via the planar embedding of the mesh.
Methods	
-	-

3.3.2. Semantic Segmentation

NeuralNetWrapper	
Used to communicate between the neural net model and the reconstructor.	
Attributes	
List<string> registeredNets	Names of the trained neural networks used in the application.
Map<string, Model> models	Dictionary of the neural network models matched with their names.
Map<string, string> paths	The path of the used model on the used machine.
Methods	
getSegmentation(Image img, String modelName, Bitmap config)	Applies the semantic segmentation algorithm on the provided image

	with the provided model and returns the segmentation of the image.
getModelPath(String name)	Returns the path of the model on the used machine using the name of the model.

Model	
Neural net model that produces the segmentations.	
Attributes	
String name	Name of the model.
String path	Path of the model.
Methods	
formatImageToModelFormat(Image img)	Formats the image to the required parameters for the specified model such as changing the dimensions or the color channels.
get and set methods for all attributes	-

Segmentation	
Output of the model. It contains the guesses produced by the model for a given image.	
Attributes	
Image image	The image used for the segmentation.
List<AABB> guesses	The guesses of the neural net stored in a list format.
String modelName	The used model for the

	segmentation.
Methods	
get methods for all attributes	-

AABB	
Represents the boundaries of the model’s guesses.	
Attributes	
float2 AA	The top left coordinate of the bounding box for the guess.
float2 BB	The bottom right coordinate of the bounding box for the guess.
Map<string, float> guesses	A map between the guessed labels and their corresponding bounding boxes.
Methods	
addGuess(String guess, float likelihood)	Adds the predicted label to the guesses map with the specified bounding box.

3.4. Server

Client	
This component is the parent component of all the user interface components.	
Attributes	
String userID	The unique ID of the user that is stored within the database.
String clientType	The device the user is currently connected such as PC, Mobile or VR.

SessionData clientSessionData	Metadata generated during the current session of the user.
List<Video> videos	The videos taken by the user.
MultiMap<Mesh, List<ImageMap>> meshes	The maps of each mesh such as bump map, displacement and normal map.
List<Scene> scenes	Scene that were processed from the users videos.
Methods	
addVideo()	Adds a video to the list of videos.
removeVideo()	Removes a video from the list of videos.
addScene()	Adds a scene to the list of scenes.
removeScene()	Removes a scene from the list of scenes.
addMesh()	Adds a mesh to the list of meshes.
removeMesh()	Removes a mesh from the list of meshes.
get methods for userID, clientType, clientSessionData, meshes and scenes	-

Connection	
Represents the connection between a Server and a Client object. Establishes the connection between these two using a Socket. Allows the transfer of raw bytes of data.	
Attributes	
Socket TCPSocket	TCP socket created for the connection between the server and the client.

byte[][] buffer	Buffer used for the data transfer.
Data currentData	Current data that has been processed from the buffer.
SessionData session	Metadata generated during the current session of the user.
Methods	
sendData()	Send data to the server.
get and set methods for TCPSocket, buffer and currentData	-

SessionData	
Represents the connection between a Server and a Client object. Establishes the connection between these two using a Socket. Allows the transfer of raw bytes of data.	
Attributes	
String userID	ID of the current user.
String password	Password of the current user.
Time loginTime	Login time of the current user.
String token	Token of the current user.
String sessionID	SessionID of the current user.
Methods	
get methods for userID, token and sessionID	-
getSessionLength()	Returns the duration of the current session.

Server	
Main server of the application. It is used as a bridge between the reconstructor, database and client.	
Attributes	
Socket TCPSocket	TCP socket created for the connection between the server and the client.
byte[][] buffer	Buffer used for the data transfer.
Reconstructor reconstructor	Representation for the reconstructor object which reconstructs.
Data currentData	Current data that has been processed from the buffer.
Methods	
get and set methods for TCPSocket, buffer and currentData	-

DatabaseWrapper	
Bridge between the server and database. Handles the data connection between them.	
Attributes	
String databaseUsername	Username of the current user.
String password	Password of the current user.
Methods	
write(Data data)	Writes into the database.
read()	Reads from the database.
query(String curQuery)	Inserts a query to the database which would be used for

	administrative purposes.

DataDecoder	
Decodes the raw bytes received through a connection. Has hard-coded instructions to decode application specific headers for particular objects and files. Produces instances of the Data class.	
Attributes	
Data currentData	Decoded version of the raw bytes buffer as an instance of the Data class.
byte[][] buffer	Buffer of the raw bytes received through a connection.
Methods	
getCurrentData()	Returns the current data.
decode()	Decodes the raw bytes received through a connection.

Data	
Provides a neat representation of data for application specific purposes. Has fields to determine the header to be used if the data is to be sent over a network, along with file type and metadata.	
Attributes	
byte[][] buffer	Buffer containing the bytes that are sent through the network.
String dataType	String denoting the type of the data communicated through the network.
String fileName	Name of the file sent through the network.

String userID	ID of the current user.
String fileExtension	Extension of the file in string format.
Time timeObtained	Time at which the data is obtained. E.g. if the type data is a video, this field contains on which date the video was uploaded to the server.
Methods	
get and set methods for all attributes	-

4. Reference

[1] “The world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.,” IEEE. [Online]. Available: https://www.ieee.org/. [Accessed: 15-Feb-2020].

[2] “Welcome To UML Web Site!,” Welcome To UML Web Site! [Online]. Available: https://www.uml.org/. [Accessed: 15-Feb-2020].

[3] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1262, 2017.

[4] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, “BundleFusion,” ACM Transactions on Graphics, vol. 36, no. 3, pp. 1–18, Jan. 2017.

[5]“Oculus Unity Getting Started Guide,” Oculus Developer Center. [Online]. Available:https://developer.oculus.com/documentation/unity/book-unity-gsg/. [Accessed: 15-Feb-2020].

5. Website

https://barisc22.github.io/MoveIt/ MoveIt website