OpenIA

Version 2.0

User Manual

Virtual Environments Lab, GSAIM,
Chung Ang University

OpenIA SW

Environment Setup

While the project has started in 2018 and it has only been released in the public domain under BSD license in 2019 and Apache license in 2021. As OpenIA is on open-source project, everyone is free and welcome to extend its capabilities. OpenIA is developed in C++. It is currently compiled on Windows using CMake and for 32bits or 64bits architectures. This manual gives you basic environmental setup and its technical details. Don't hesitate to ask questions and share your experiences and take a look at the Github source repository OpenIA.

Forking the GIT

The OpenIA project is hosted in the GIT under the link https://github.com/younghochai/motion-matrix (Figure1.1). The latest release is always setup in the master branch that is set as default of this github page. One can create a local copy of the entire repositorty by clicking on the fork option available in the right top corner of the github page. It is recommended to fork the repository before bringing the code base into your local machine.

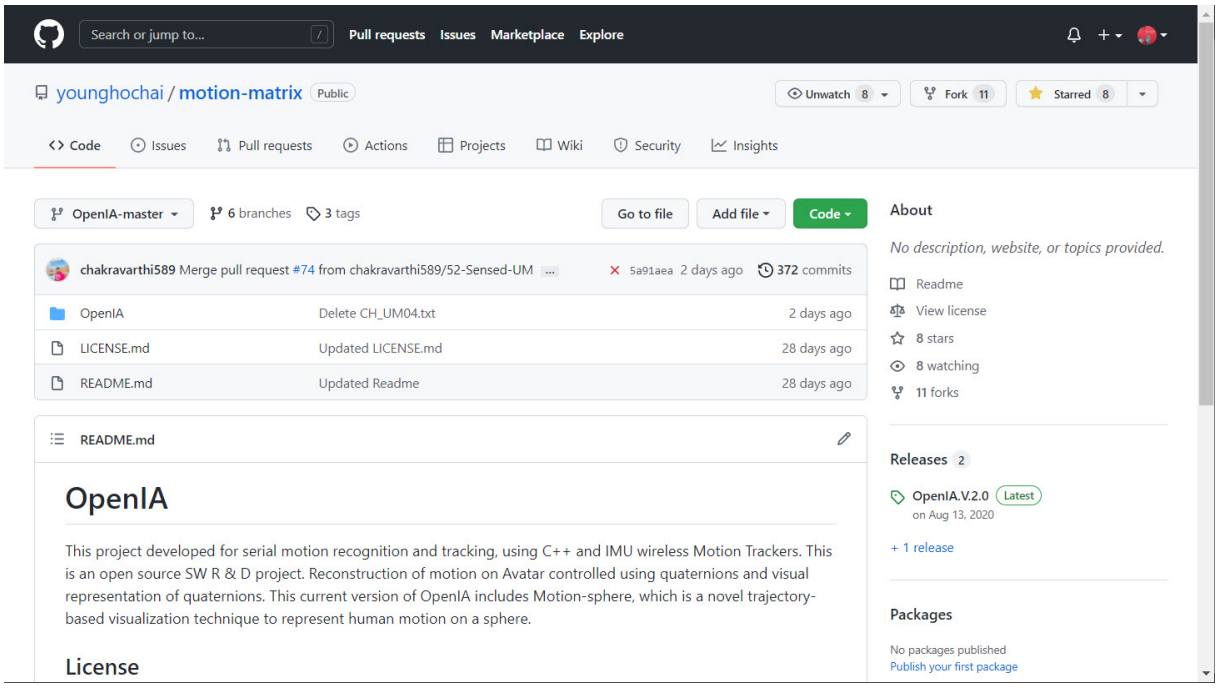

FIGURE 1.1: OpenIA project Github home page.

The GIT provides three options (Figure 1.2), direclty clone using https or ssh command, next using a git desktop application or directly download ZIP for cloning the remote repository into

the local machine (https://desktop.github.com/ for windows). One of use this desktop tool as an alternate for command line interface to clone the code base to the local machine.

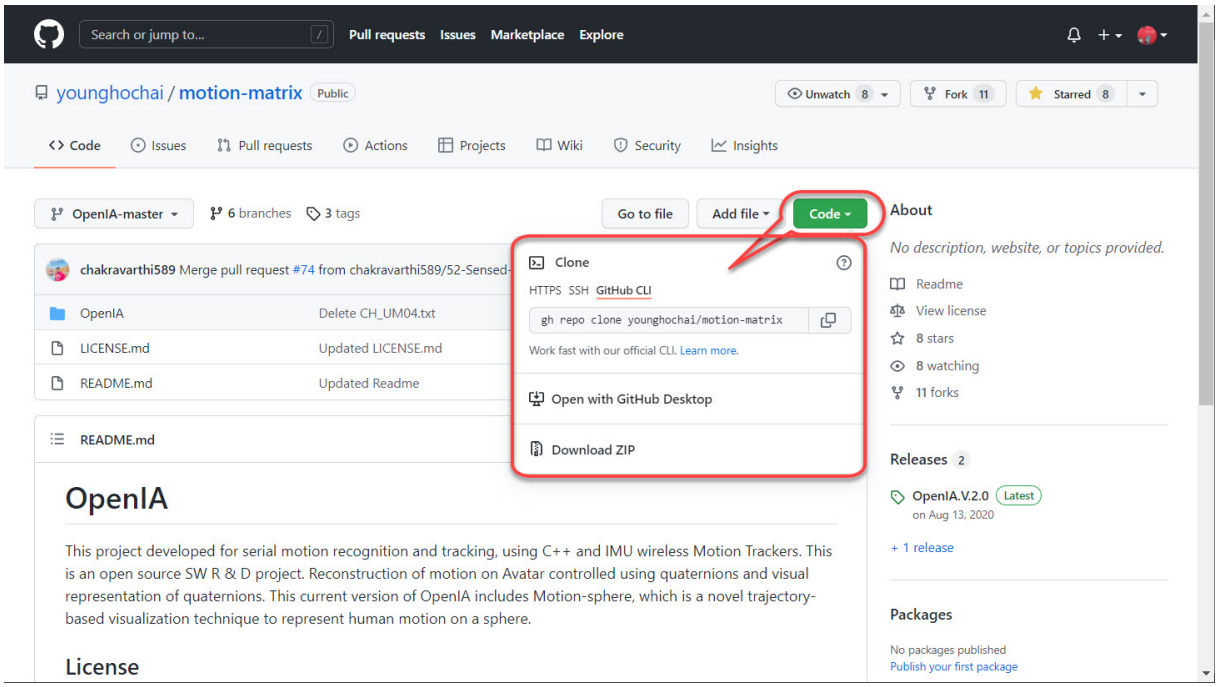

FIGURE 1.2: OpenIA Project clone window

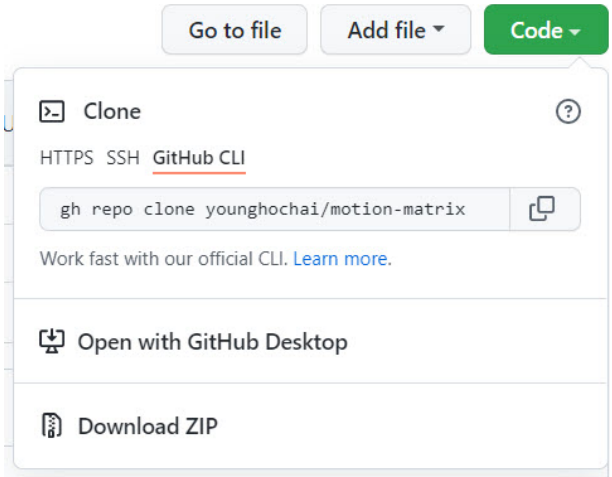

FIGURE 1.3: OpenIA Project clone options

Licenses

This work is dual-licensed under BSD-3 and Apache License 2.0. You can choose between one of them if you use this work. For more detailed description of each license follow the links.

- BSD-3: https://choosealicense.com/licenses/bsd-3-clause/
- Apache License 2.0: https://choosealicense.com/licenses/apache-2.0/

Prerequisites

OpenIA is implimented in C++. Various 3rd party APIs and tools are used within the motion sphere. A list of all those pre-requisite tools and libraries are given in the table 1.1

TABLE 1.1: Software Requirements

Sl. No	Tool / Libraries	Usage in OpenIA
1	Visual Studio for C++	IDE for all programming, building and execution activities
2	OpenGL	Graphical Library for visualizing Human Motion
3	Visualization Tool Kit	Graphical Library for Motion Reconstruction on 3D avatar
4	Xsense drivers	Used for establishing connection with the Xsense IMU sensors
5	Point Cloud Library	Used for establishing the position of joints in the human body
6	CMake	To build the project solution

Likewise there are some specific hardware requirements for capturing the human motion. Table 1.2 details the hardware that is used to capture human motion. it must be noted that all the software requirements are must have requirements. Whereas the minimal hardware is the computing system. While Xsense and LiDARs are used for motion capture feature. Even without the Xsense IMU sensors and the LiDAR the OpenIA can be used to visualize the pre-captured motion data.

TABLE 1.2: Hardware requirments

Sl. No	Hardware	Description
1	Xsens Maven IMU (Required for MoCAP)	10 numbers for full body motion capture
2	LiDAR (Required for MoCAP)	1 or 2 depending on application requirment
3	A Computing environment	A system with decent computing capabalities and graphics card

Building the project

Once the project is cloned on the local system. The project has the following folder tree structure.

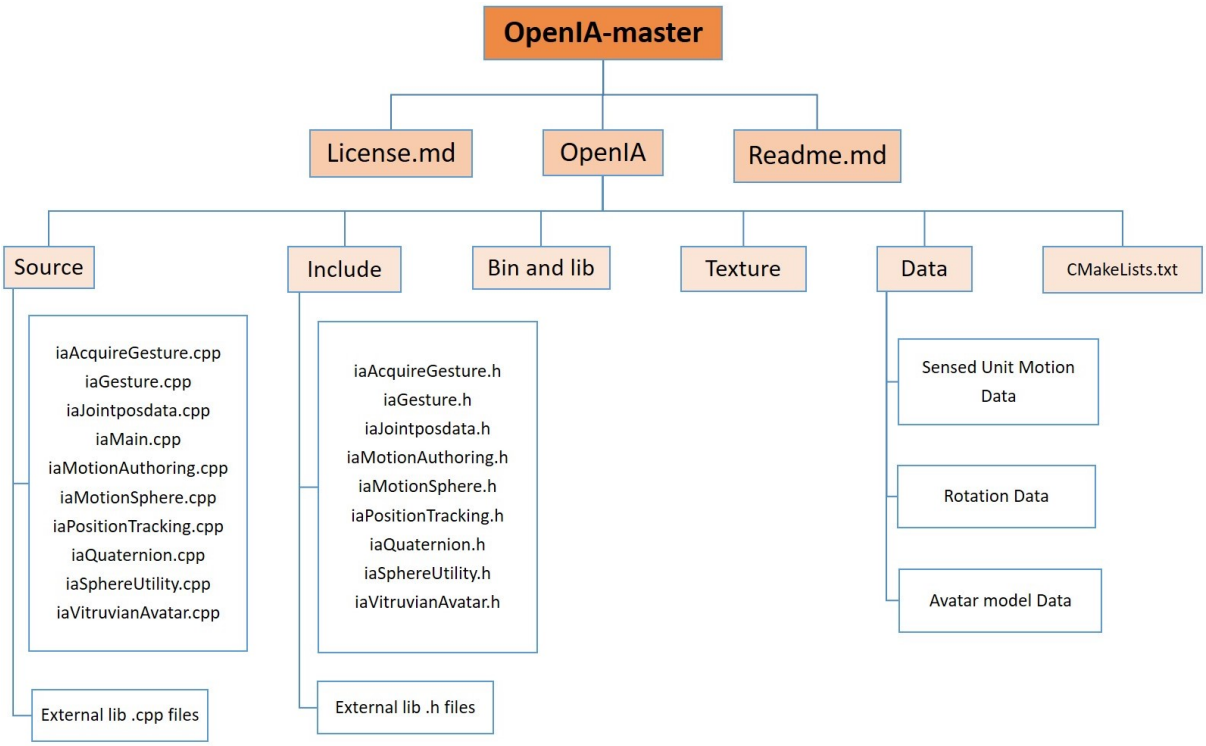

FIGURE 1.4: OpenIA Project folder structure tree in Github repository

Note: *The folder structure may be slightly different depending on the version of the OpenIA, but the proceedure to build the project remains the same.*

The Cmake tool is used to build the solution as shown in Figure 1.5. The solution file thus created in a separate build folder can be used for building the project that can be executed.

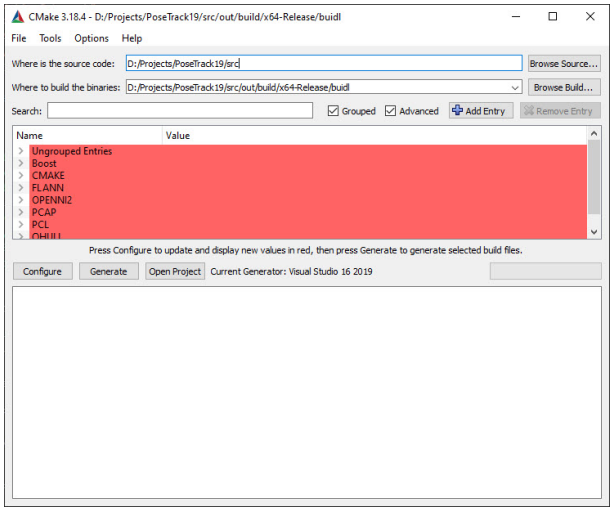

FIGURE 1.5: OpenIA Project folder structure tree in Github repository

Execution and Running

Building the project is as simple as selecting INSTALL in the folder structure window of the visual studio and executing build by right clicking (Figure 1.6).

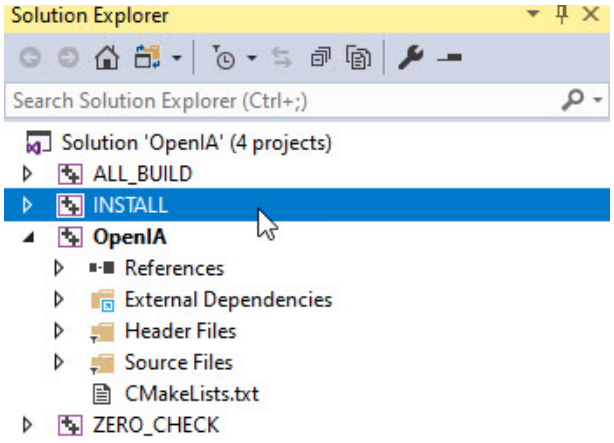

FIGURE 1.6: Building the Project from the Visual Studio.

The project can be run after the build is completed. Three windows appear, one with the 3D avatar that is rendered from the Visualization Tool Kit (Figure 1.7), second a Heterogeneous sensing motion capture window (Figure 1.8) and the other is a Unit Sphere rendered from the OpenGL renderer window(Figure 1.9).

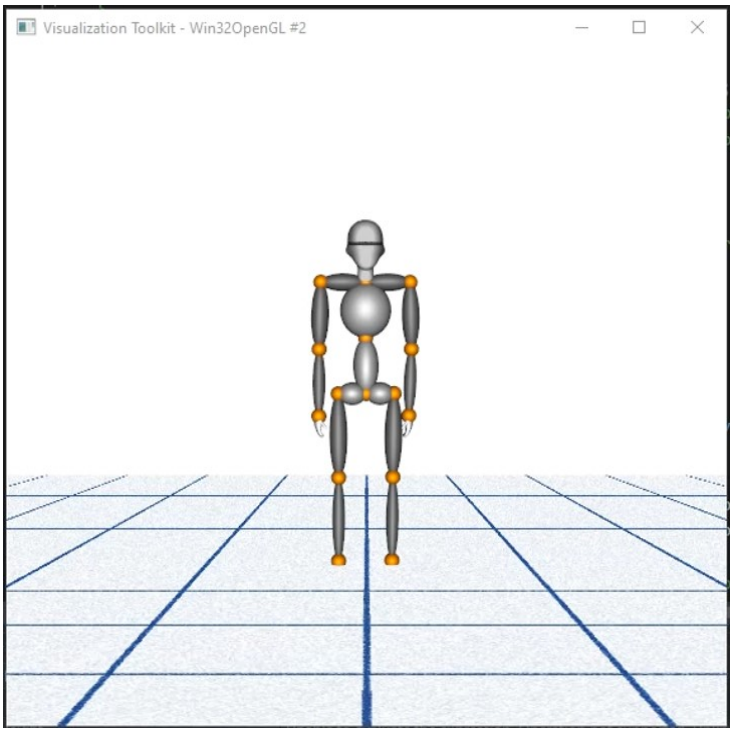

FIGURE 1.7: The first opening window of the OpenIA: 3D Avatar.

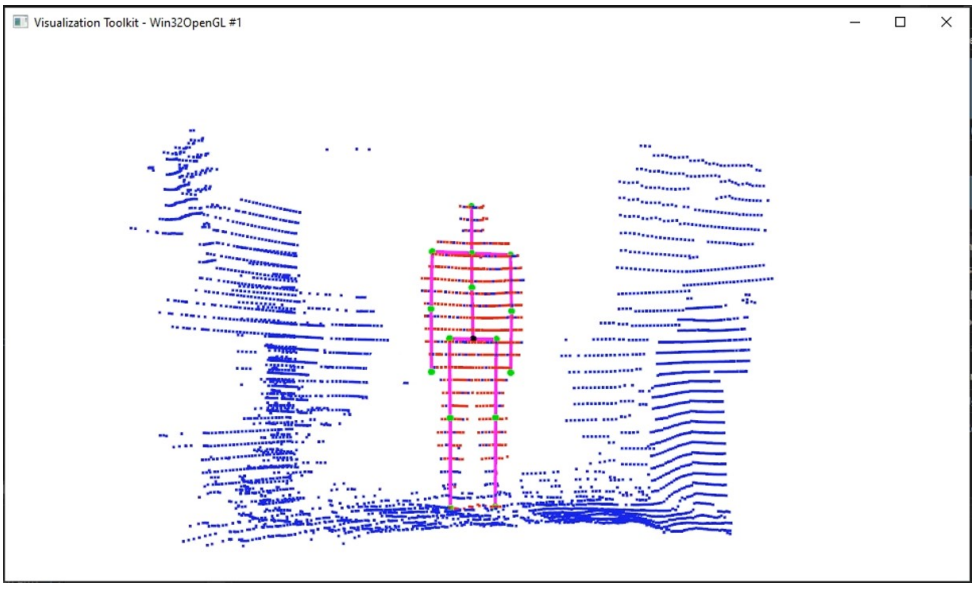

FIGURE 1.8: The second opening window of the OpenIA: Heterogeneous Sensing.

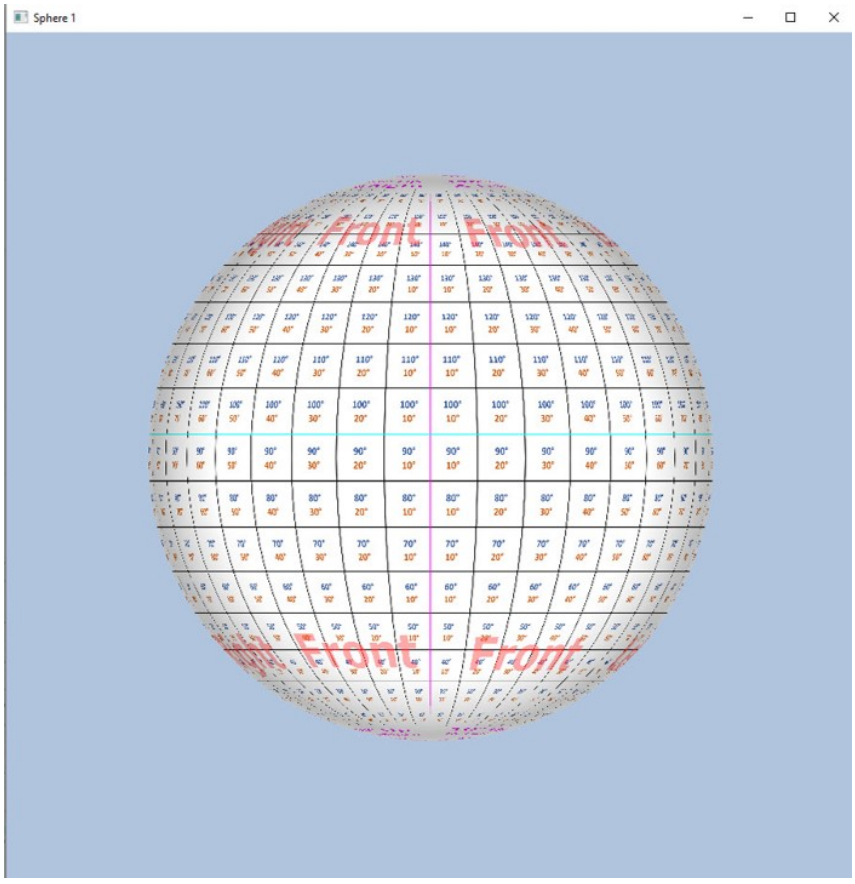

FIGURE 1.9: The third opening window of the OpenIA: Motion-Sphere.

Primary Modules

Motion sphere is composed of the following primary modules. 1) iaSphereUtility 2) iaQuaternion 3) iaMotionSphere 4) iaAcquireGesture 5) iaVitruvianAvatar 6) iaPositionTracking

Class Description

Figure 1.10 shows a description of all the C++ structures that is used in the OpenIA. The *struct Avatar* is the main data structure that is used throughout the code base for accessing the data of a motion frame. A motion frame consists of orientation data of 10 bone segments starting from b_0–b_9 in form of quaternions.

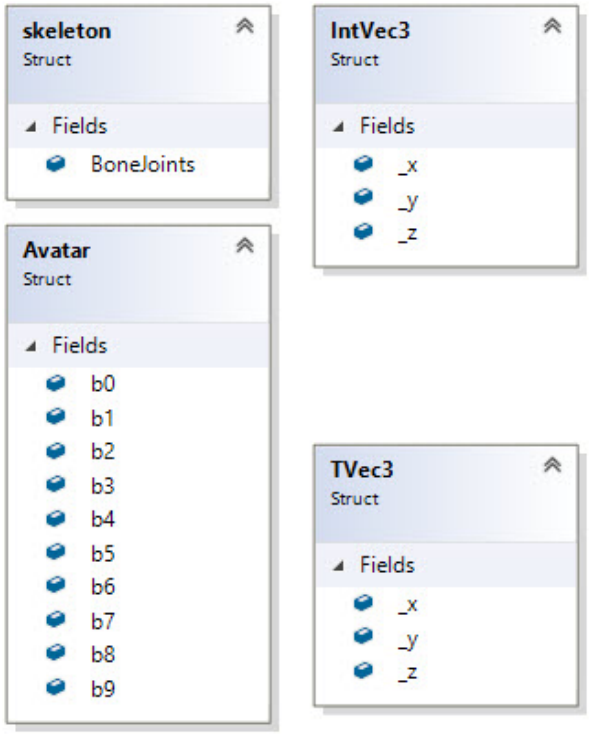

FIGURE 1.10: C++ Structures that are handy in OpenIA

Figure 1.11 shows the description of all the classes in the OpenIA tool's code base.

iaMain

The iaMain.cpp is the function called when the program is run. The execution of all class functions begins in main function with multiple threads and have unique id for each class functions.

iaPositionTracking

The PositionTracking uses the point cloud library and the LiDAR to compute the subjects height. various point cloud trasformations like the segmentation of point cloud, the estimation of pelvis position, saving the quaternions in to files, updating the bone segment data in real time and so on are implimented in this class.

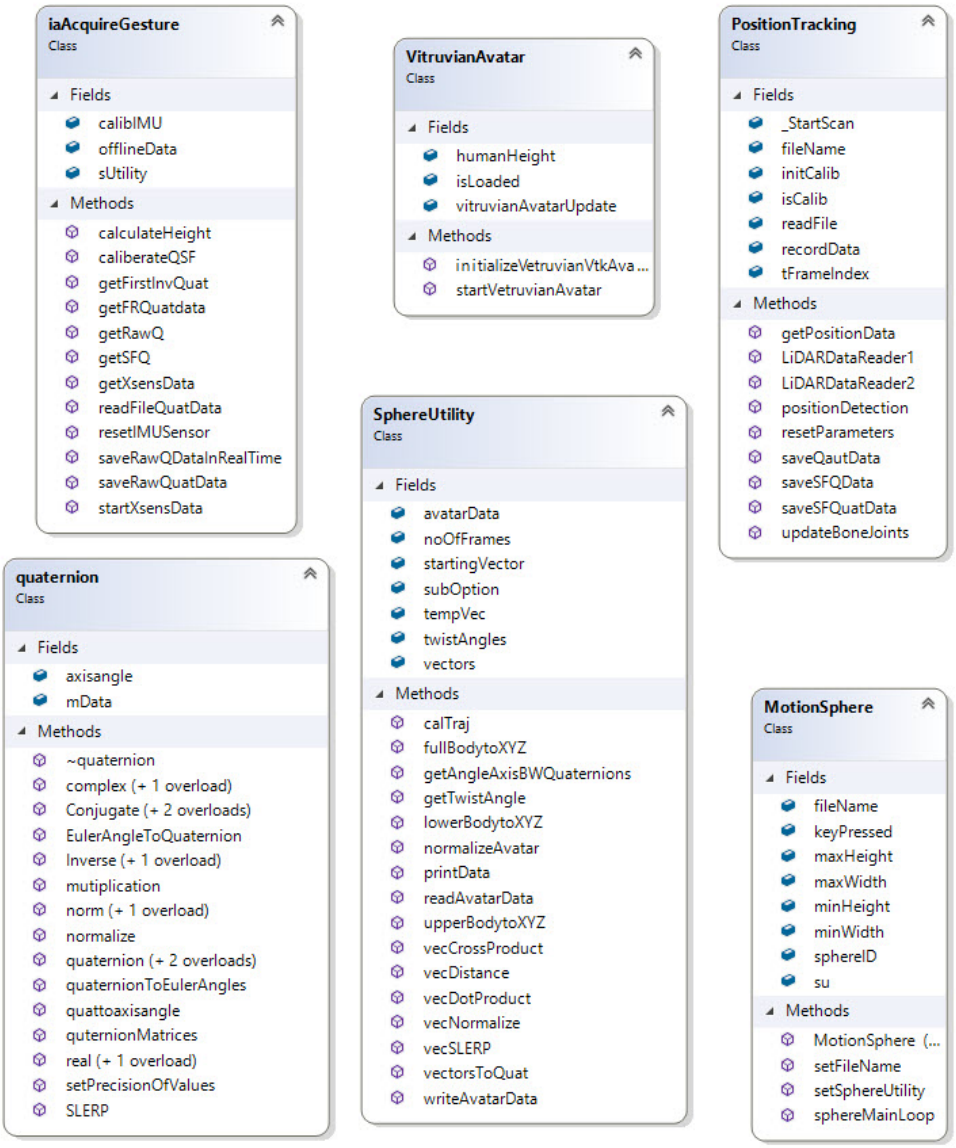

FIGURE 1.11: Class diagram of all the modules in the OpenIA

iaAquireGesture

The iaAquireGesture mainly impliments the motion capture functionality. The class starts the xsense IMU sensors (*startXsenseData()*), gets raw quaternions from the sensors (*getRawQ*), calibrates the 3dAvatar (*calibrateQSF()*), and applies a series of quaternion transformations to save the motion frame in a standard file format.

iaSphereUtility

Contains mathematical operations including vector and quaternion algebra essentials for visualizing the human motion as 3D points. SphereUtility class implements methods not limited to *readAvatarData* to read from the captured file, *fullBodyXYZ* to convert all the quaternion frames into 3D points, and *calTraj* to calculate the trajectory based on kinematic hierarchy.

iaQuaternion

All the quaternion operations are handled in the class quaternion. Operations like the converting of quaternions to euler angles, normalizing, quaternion multiplication, conversion of quaternions into axis and angles, spherical linear interpolation, inverse and so on.

iaVitruvianAvatar

VitruvianAvatar is the VTK (Visualization Tool Kit) implimentation of the 3D avatar. The avatar exposes a attribute called the *vitruvianAvatarUpdate*, which is of type *struct Avatar*. This attribute can be updated in realtime to see the animation of the avatar as and when the data is captured or rendered by the iaAquaireGesture or the MotionSphere modules respectively. The Avatar is initialized and triggered using the methods exposes in this class.

iaMotionSphere

MotionSphere is the main module that consists of a GLUT main loop. When the *sphereMainLoop()* is called the tool enters into a infinite event based loop. The MotionSphere reads the data with the help of the other utility classes and renders the visualization on the unit sphere. Mainly it holds the window parameters, an instance of SphereUtility to operate all the visualization functions that are implimented using openGL

iaMotionAuthoring

The iaMotionAuthoring.cpp class implemented for interactively edit existing motion data using 3D avatar models with user-defined scenarios and synthesize numerous variations of motion sequences.

Heterogeneous sensing system

In this project work, a human motion tracking system that uses lidar and IMU sensors to estimate 3D human pose in real-time. Human motion tracking includes human detection, height and skeleton parameters estimation, position, and orientation estimation by fusing lidar-IMU sensor data. Finally, reconstructing the human motion on a virtual 3D avatar.The flow of the proposed system is shown in Figure 1.13.

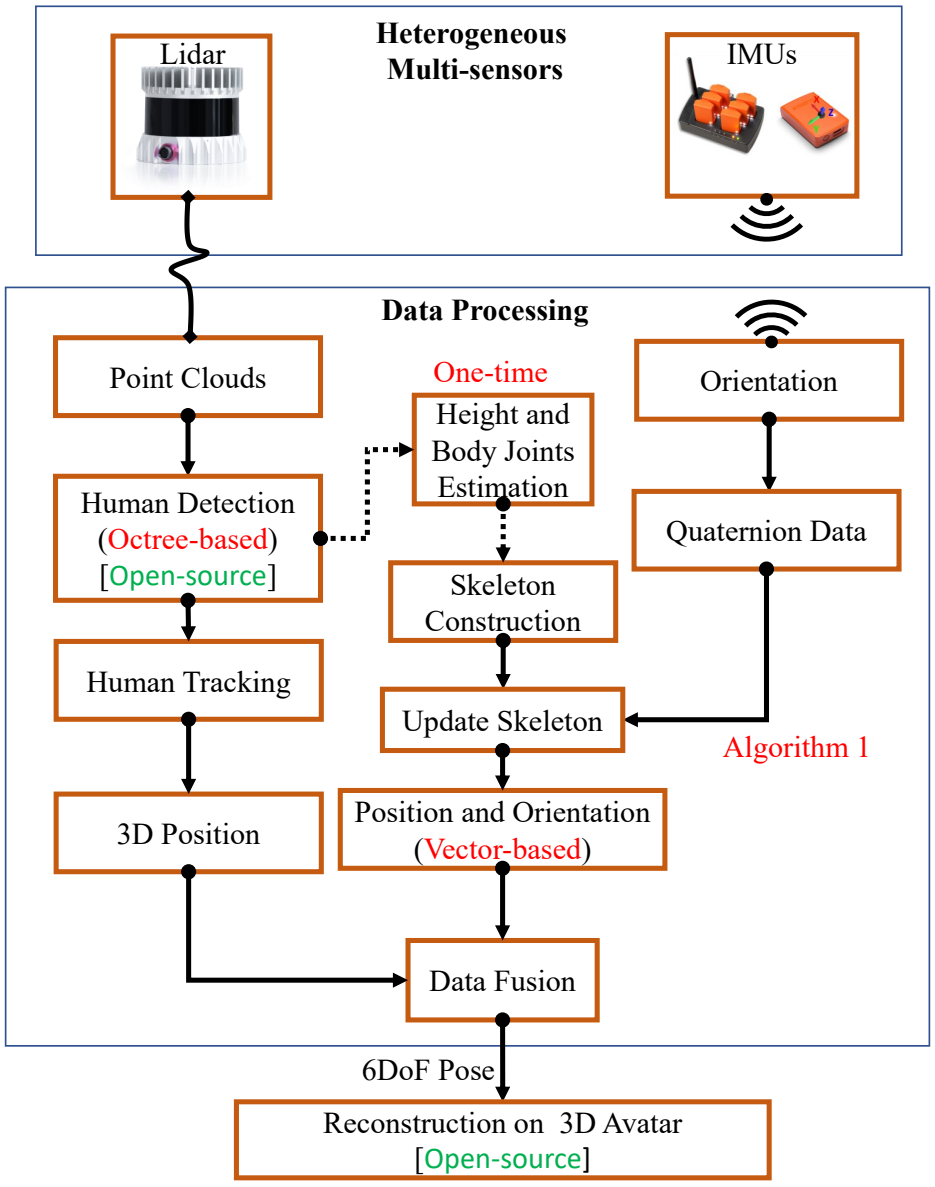

FIGURE 1.12: Heterogeneous sensing flowchart

Motion Sphere

Motion-sphere is a trajectory-based visualization technique to represent human motion on a unit sphere. Motion-sphere adopts a two-fold approach for human motion visualization, namely a 3D avatar to reconstruct the target motion and an interactive 3D unit sphere, that enables users to perceive subtle human motion as swing trajectories and color-coded miniature 3D models for twist.

Any human motion is a sequence of quaternion rotations, represented visually as a trajectory on a 3D unit sphere. Figure 1.13 shows an overview of the proposed Motion-sphere. A 3D LiDAR sensor is used to track the position of a moving human body. The 3D position of the pelvis is computed using the segmentation of raw point cloud data based on the user's height and IMU sensors are used to estimate the orientation of each bone joints during human body motion in a real environment. When a user performs a motion, orientation data from the IMU sensors are transformed into a 3D avatar's coordinate frame for reconstruction.

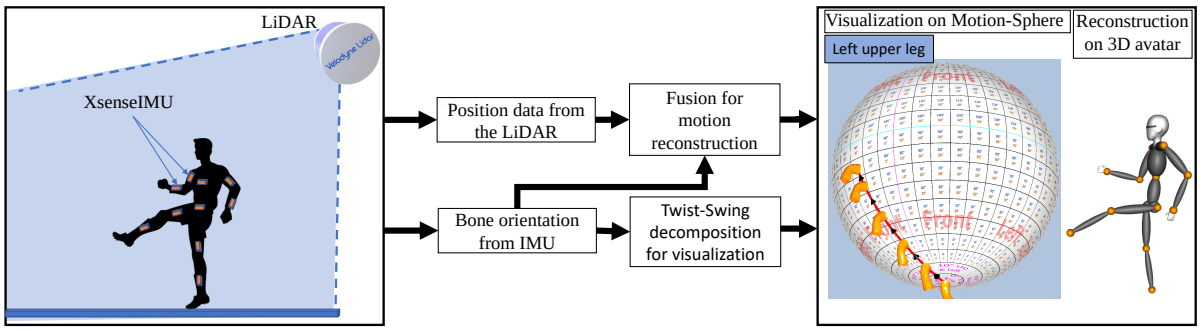

FIGURE 1.13: Overview of the Motion-sphere