

OpenIA

Version 3.0

User Manual

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 an 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/motionmatrix> (Figure1.1).

The latest release is always setup in **the OpenIA 3.0**. One can create a local copy of the entire repository 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.

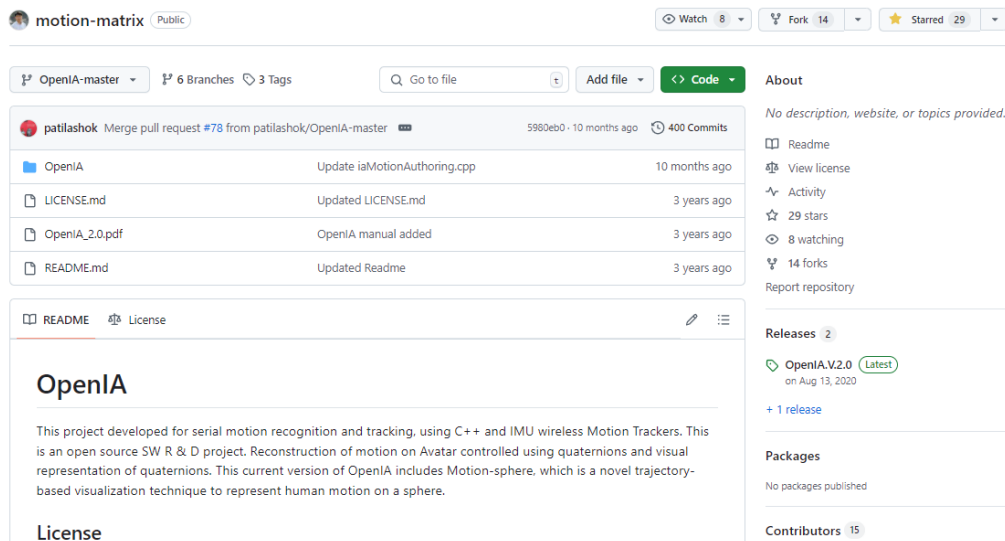
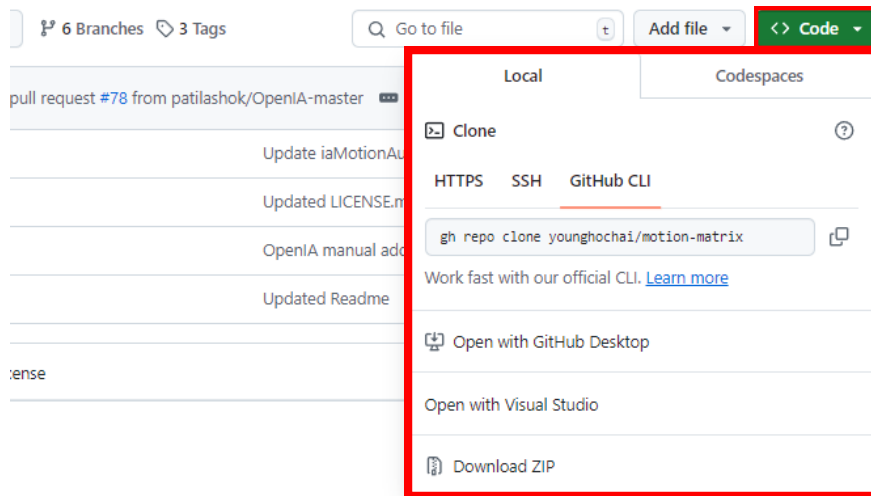


FIGURE 1.1: OpenIA project Github home page.

The GIT provides three options (Figure 1.2), directly 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.



oped for serial motion recognition and tracking, using C++ and IMU wireless Motion Trackers. This SW R & D project. Reconstruction of motion on Avatar controlled using quaternions and visual quaternions. This current version of OpenIA includes Motion-sphere, which is a novel trajectory- technique to represent human motion on a sphere.

FIGURE 1.2: OpenIA Project clone window

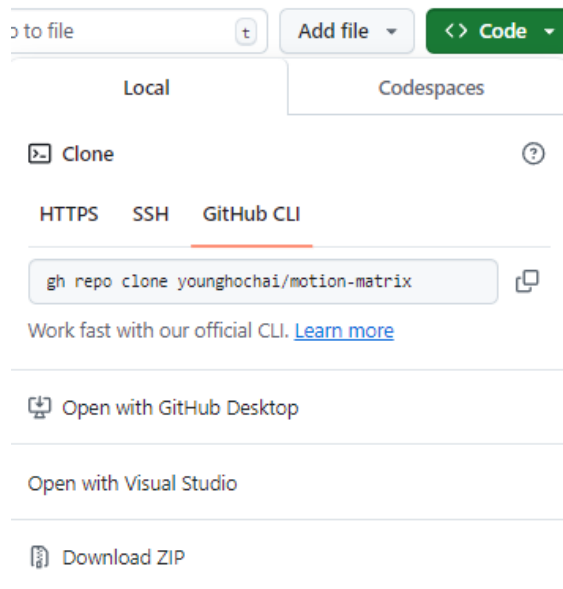


FIGURE 1.3: OpenIA Project clone options

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 implemented 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	CMake	To build the project solution
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	Visualization Tool Kit	Graphical Library for Motion reconstruction on 3D avatar
7	jsoncpp	Used for json file on the C++
8	PCAP	Used for packet capture
9	ImGui	Used for graphic user interface for motion editing functions.

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 precaptured motion data.

TABLE 1.2: Hardware requirements

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 requirement
3	A Computing environment	A system with decent computing capabilities and graphics card

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).

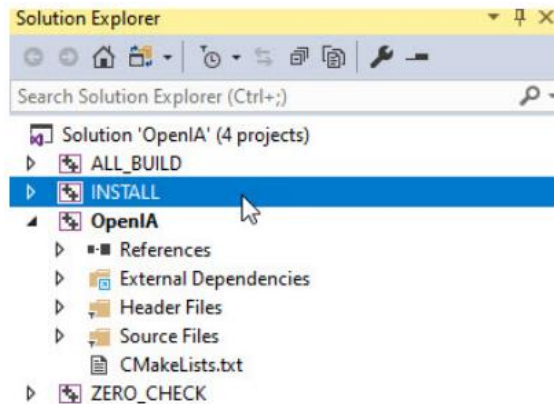


FIGURE 1.6: Building the Project from the Visual Studio.

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).

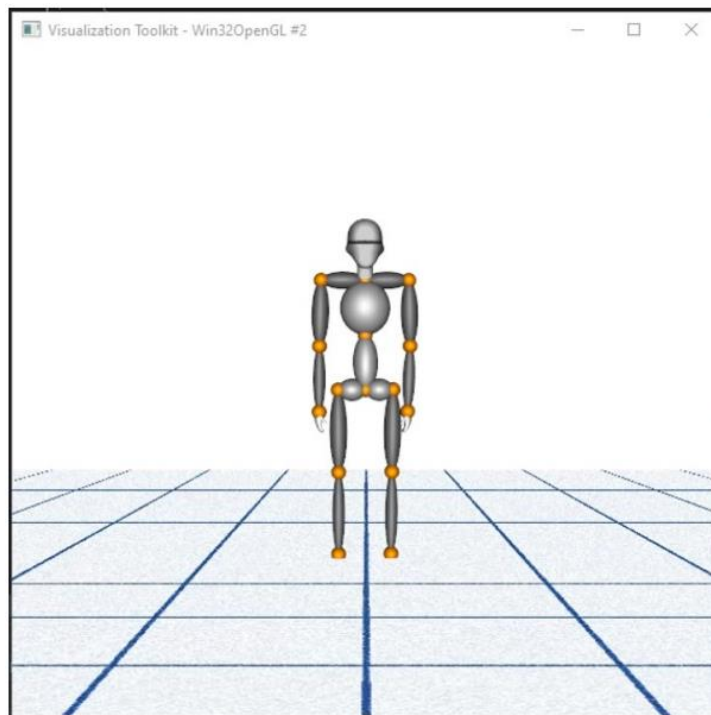


FIGURE 1.7: The first opening window of the OpenIA: 3D Avatar.

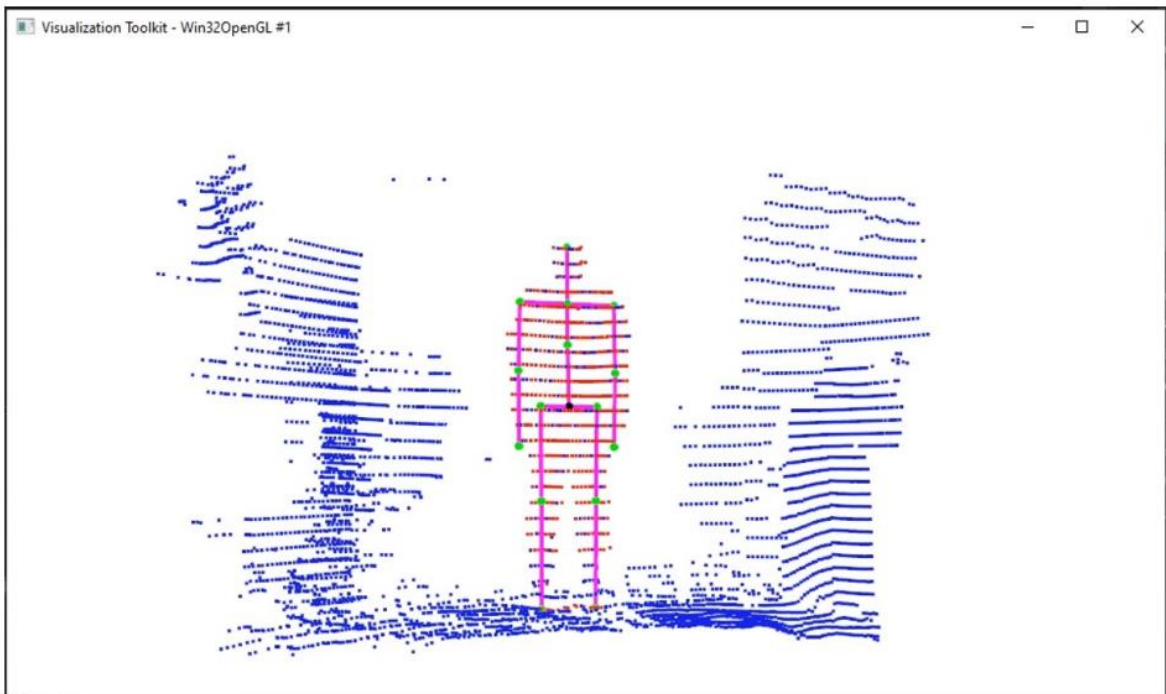


FIGURE 1.8: The second opening window of the OpenIA: Heterogeneous Sensing.

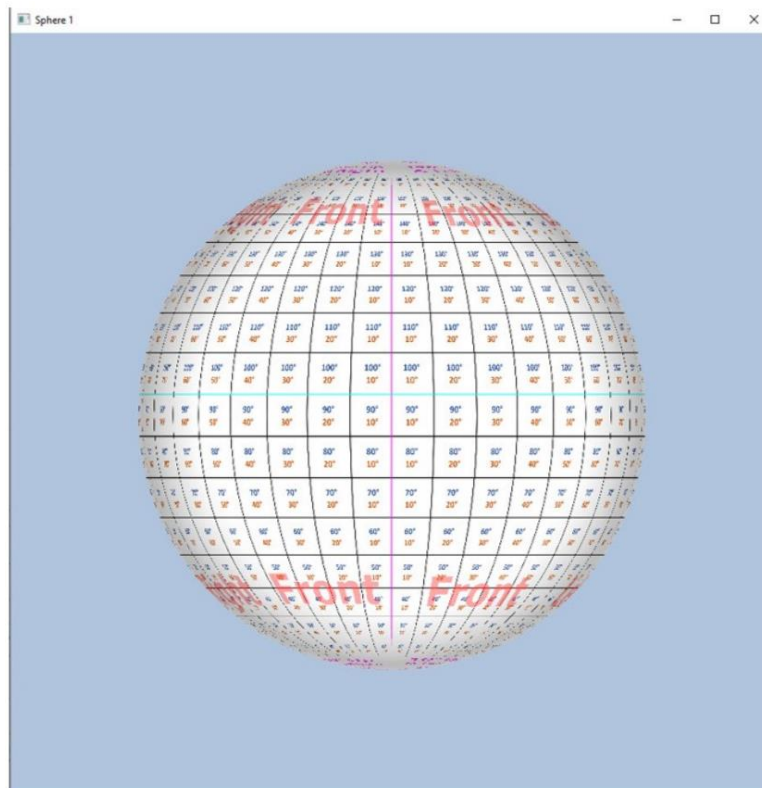


FIGURE 1.9: The third opening window of the OpenIA: Motion-Sphere.

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 b0–b9 in form of quaternions.

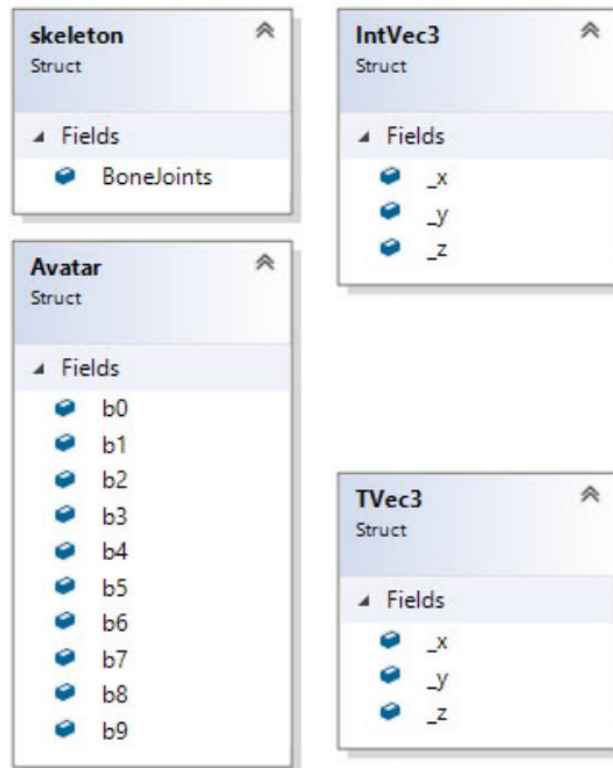


FIGURE 1.10: C++ Structures that are handy in OpenIA

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 subject's height. various point cloud transformations like the segmentation of point cloud, the estimation of pelvis position, saving the quaternions into files, updating the bone segment data in real time and so on are implemented in this class.

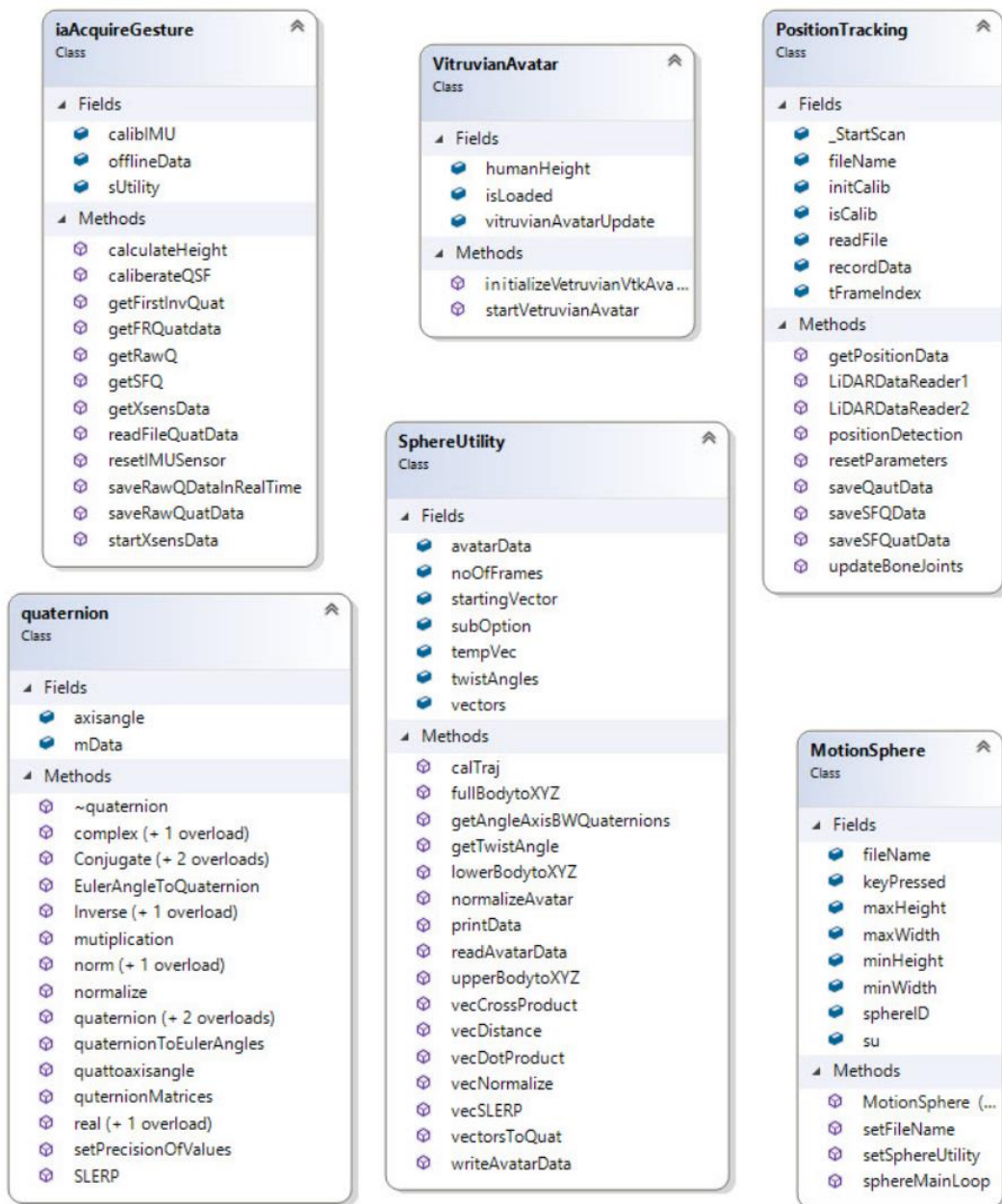


FIGURE 1.11: Class diagram of all the modules in the OpenIA

iaAquireGesture

The iaAquireGesture mainly implements the motion capture functionality. The class starts the Xsense IMU sensors (startXsenseData()), gets raw quaternions from the sensors (getRawQ), calibrates the 3dAvatar (calibrateSFQ()), 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) implementation of the 3D avatar. The avatar exposes an 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 iaAquireGesture 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 an 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 implemented using OpenGL.

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.

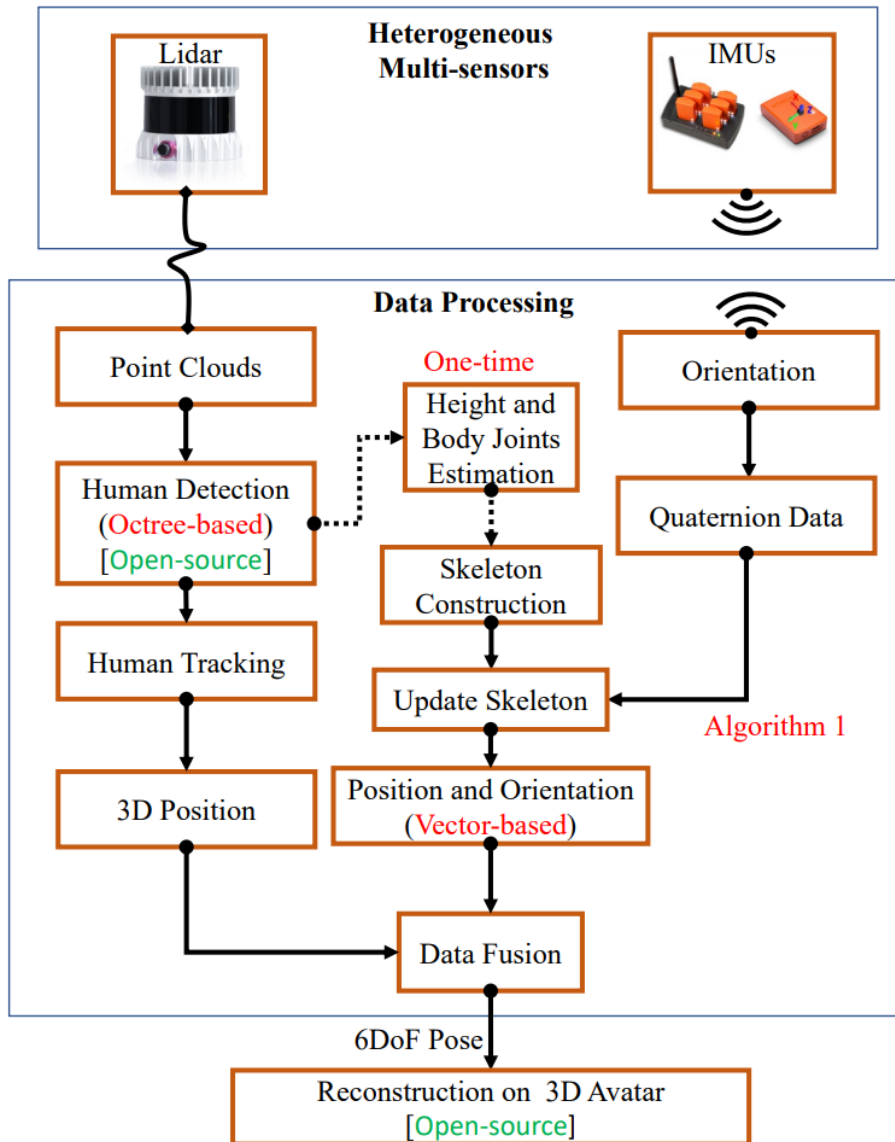


FIGURE 1.12: Heterogeneous sensing flowchart

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.

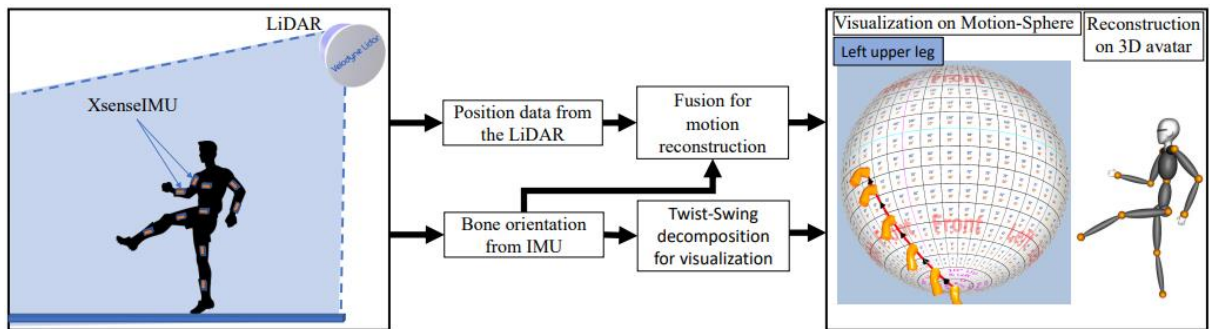


FIGURE 1.13: Overview of the Motion-sphere

Thread

To capture human body motion and save the angle data successful, there are some threads in main.cpp.

XSensDataReader

Thread about IMU Xsens. It searches the signal about IMU sensors and notices users to ready to record the motion capture data.

startAvatar

Thread about VitruvianAvatar. It renders avatar based on VTK and shows how actually the human avatar moves.

getOSLiDARdata

Thread about LiDAR. Previously, we used VelodyneCapture LiDAR. (This repository contains it also) Now we uses Ouster LiDAR. It searches the signal about LiDAR and renders the point clouds.

poseTracking

Thread about the point clouds window. LiDAR computes the point clouds, and this thread shows them.

motionSphere

Thread about motion sphere. It shows spheres about each joints, reads the text file which contains motion captured quaternion angle, and renders the points about that frames. Also, It passes the data to VitruvianAvatar so that the avatar can move based on the data.

saveIMUData

After recording all the human motion, this thread saves the quaternion data on a text file.

```
thread t1(XSensDataReader);
thread t2(startAvatar);
thread t3(getOSLiDARdata);
//thread t3(getLiDARdata1);
//thread t4(getLiDARdata2);
thread t5(poseTracking);
thread t6(motionSphere);
thread t7(saveIMUData);
```

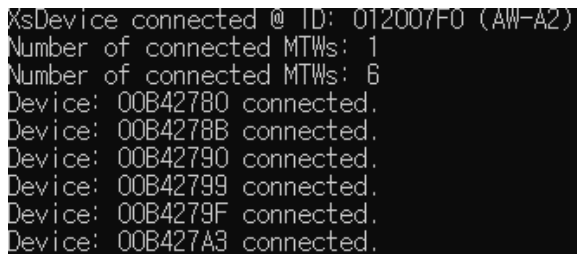
FIGURE 1.14: Thread of the Motion-sphere

How to use

Prerequisite

Before you control following keyboards, you should check these status.

- All the windows should open. (Avatar, Motion sphere, and LiDAR)
- Point cloud should render.
- Xsens IMU sensors should connect.



```
XsDevice connected @ ID: 012007F0 (AW-A2)
Number of connected MTWs: 1
Number of connected MTWs: 6
Device: 00B42780 connected.
Device: 00B4278B connected.
Device: 00B42790 connected.
Device: 00B42799 connected.
Device: 00B4279F connected.
Device: 00B427A3 connected.
```

FIGURE 1.15: Signal from the Xsens IMU sensors

FIGURE 1.16: Point clouds from the LiDAR sensor

Motion Capture

The motion sphere motion capture system works in the following order :

- i : LiDAR captures the spatial information and saves it. You should press this button without users.
- s : Save and initialize the information about the LiDAR and IMU sensors.
- b : Heading reset and pose calibration of the Xsens IMU sensors. Before you press this button, you should check whether the IMU sensors are paired.
- v : Pose calibration again. Before you press this button, users should attach the Xsens IMU sensors on their body parts.
- d : Extract the skeleton on the point clouds. Before you press this button, users should stand at the center of the space.
- a : Make the avatar move.
- n : Start and end recording. Users can move body parts ahead. If you press this button again, all the frames are saved on the text file.

Read text file

- 1 : You can read captured quaternion angle data from text file. If it doesn't work, please check the path of the text file.

Real-Time render

- 2 : You can move avatar and draw the trajectory on motion sphere with real-time rendering.

OpenIA 3.0 release

iaMotionSphere

Real Time Pose Tracking

Additional function

User Interface

There is a frame editing function which is utilized by keyboard and mouse interaction. (Up and down, right and left, and scroll) This release contains graphic user interface include these editing function so that we can observe the value changes.

We adopt the user interface library for C++ named ImGui.

- Edit Mode (Spherical / Quaternion)
 - Spherical : Edit function on spherical coordinate. You can modify the longitude and latitude of each points.
 - Quaternion : Edit function about the quaternion of each points. You can modify x, y, z value.
- Magnitude : You can choose how many distance the points move. The distance decides acceleration of the motion.
- Previous Index / Next Index
- Save
- Undo
- Reset

iaQuaternion

Additional operators about addition and subtraction of quaternion.

- addition
- subtraction