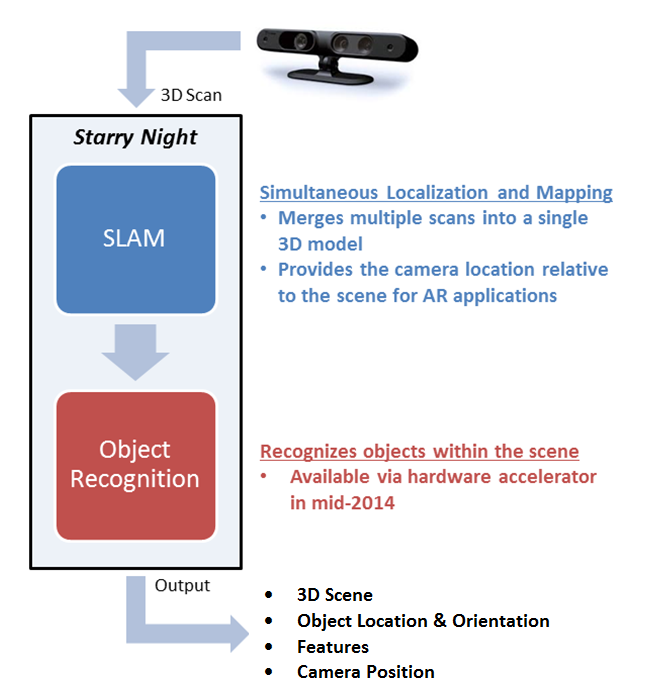
**Starry Night User Manual**

1. Description
   1. Starry Night is an easy to use Unity plugin that provides high-level 3D computer vision processing functions that allow applications to interact with the real world.
   2. The goal of Starry Night is for application builders to use very complex and sophisticated computer vision functions without the need to understand computer vision. As an example, simple function calls such as <where is the camera location> or <what is the object orientation> allow the application to recognize the nearby scene or objects.
   3. Starry Night can run on multiple platforms supported by Unity, although version 1.0 will focus on Android devices.
   4. Starry Night is also designed with *hardware acceleration* in mind. Future versions will support hardware acceleration using an FPGA, GPU, and/or DSPs.
   5. Version 1.0 will provide SLAM (Simultaneous Localization and Mapping) functions which can be used for 3D reconstruction, augmented reality, robot controls, and many other applications.
   6. Version 2.0, to be released soon, will provide object recognition capability for all types of 3D objects.
   7. Future versions will enhance both the SLAM and object recognition capabilities as well as add analysis tools for measurements and feature extraction.
   8. Starry Night will interface to any type of 3D sensor or stereo camera. However, version 1.0 will interface only to a PrimeSense Carmine sensor. Future versions will support other sensors and cameras.
   9. It’s best to physically mount (e.g., using Velcro) the sensor to the back of the Nexus tablet as shown in the picture below. A micro USB OTG-to-USB adapter cable is required to make the connection from the Carmine sensor to the Nexus tablet.



* 1. The ‘trial’ version of the plugin has a built-in watermark of VanGogh Imaging’s logo in the 3D model output. Also, the camera location only provides its coordinates and does not provide the rotation matrix and translation vector. If you wish to use Starry Night for commercial applications, please contact VanGogh Imaging to obtain a commercial license. VanGogh Imaging is interested in providing a commercial licenses for high-volume, commercially viable applications.

1. Starry Night Block Diagram



Note: Object Recognition will be provided in Starry Night version 2.0.

1. Output and Functions
   1. Output – The following information is provided in real-time by Starry Night
      1. Camera Location

When the plugin is first activated with the START function command, the location of the camera is set as the origin (0,0,0) in 3D space. As the camera moves around in 3D space, the plugin provides (in real-time via a function call) new coordinates of the camera that show the relative position of the camera with respect to the origin in the original 3D Cartesian coordinate system. This allows the application to know the current position of the camera and thus allows control of devices and display rendering based on this information.

* + 1. Point Clouds

Two point clouds, global point cloud and current point cloud, are displayed in real time once the application starts. The two point clouds provide visual information to help users monitor SLAM status.

* + 1. GlobalMesh.obj

The GlobalMesh.obj represents the final 3D model that’s constructed via the SLAM function and is generated only after SLAM is stopped. The mesh quality of this 3D model is limited by the resolution of the sensor being used.

* 1. Functions
     1. Start

This function starts the SLAM process which both constructs the global model and tracks the camera location with respect to the 3D model. This function also sets the origin of the camera at (0,0,0) and establishes the global Cartesian coordinate system.

* + 1. Get Camera Location

This function returns the current camera position relative to the global model. It outputs the current position that can be used to find the camera’s exact location. The camera position consists of its coordinates in the global 3D space as well as the translation vector and rotation matrix which indicates the direction the camera is facing. (Note: The translation and rotation vector is provided only in the commercial license version of the software.)

* + 1. Get Point Clouds

There are two functions to provide point cloud information. One function returns coordinates for the global point cloud and the other function returns coordinates for the current point cloud. Together, they can be displayed to show the visual representation of what the camera is currently seeing as well as the scene around the current camera view.

* + 1. Stop

This stops the SLAM process.

* + 1. Get GlobalMesh.obj

This function is available only after the Start function is invoked and then the Stop function is executed, i.e., after the SLAM processing is completed. The get GlobalMesh.obj function returns the global model which has been constructed via the SLAM function.

1. Unity3D Function Call with Plugin
   1. The block diagram below shows the interaction between the application and the plugin.

**Unity3D Function Call with Plugin**

**Starry Night Plugin**

**Unity3D Script**

Send Setting Params to Plugin

Initialize SLAM

Start SLAM and Initialize Settings

LOOP

Display Global Point Cloud (white)

Global Point Cloud

Update Mesh

Current Point Cloud

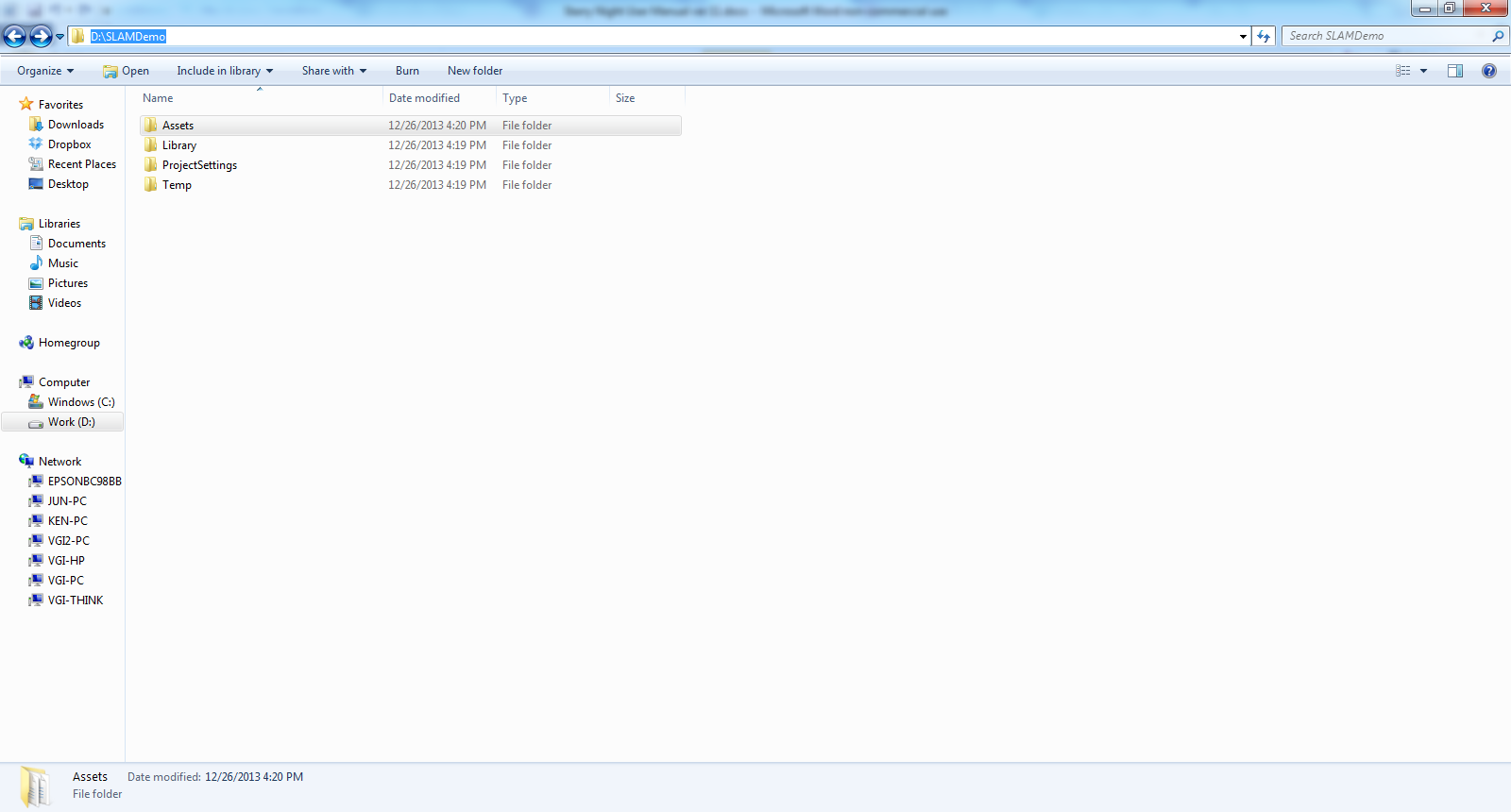
Display Current Point Cloud (green)

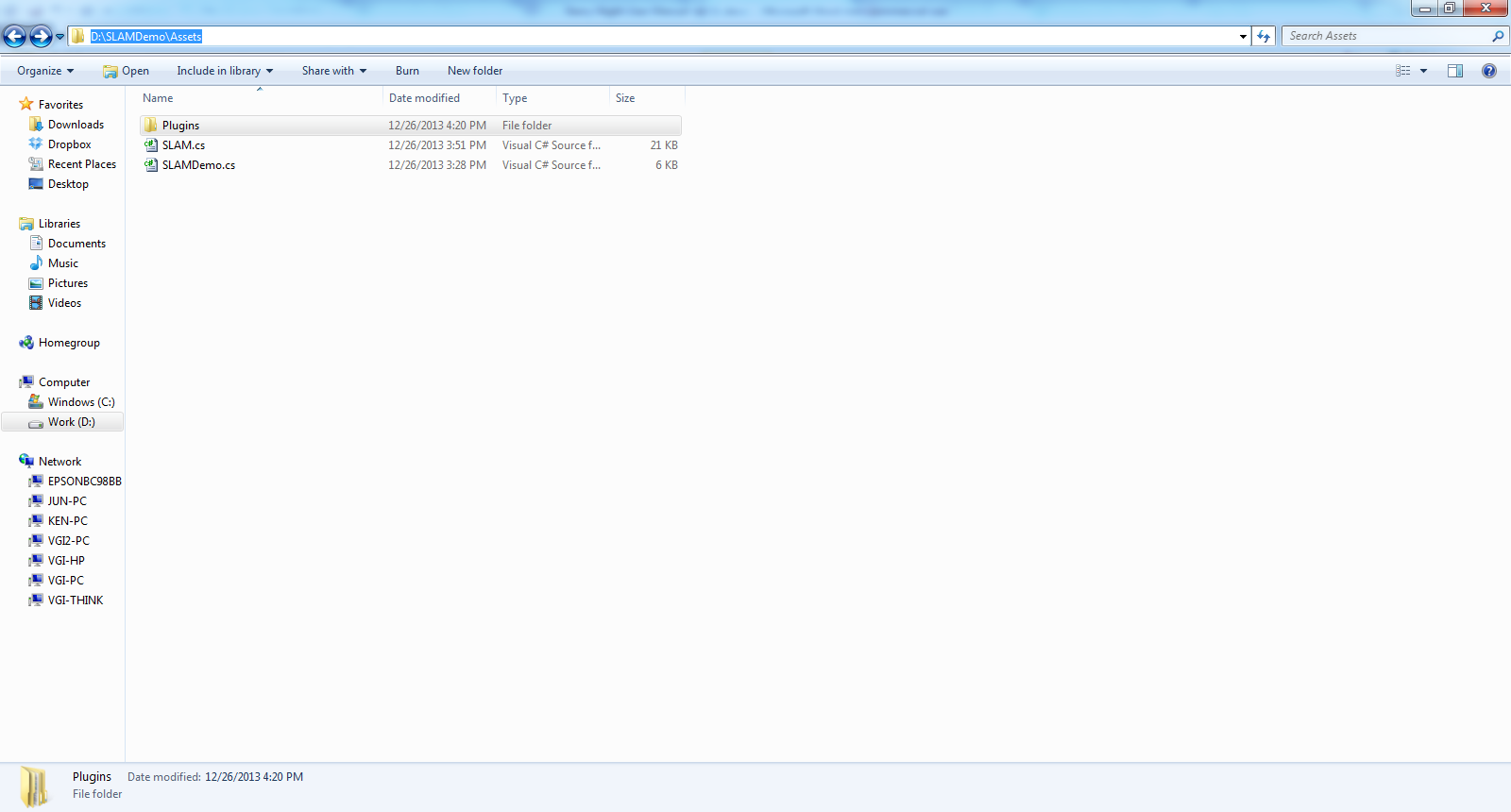
Stop SLAM

Stop SLAM

DESCRIPTION: Before starting the SLAM process, the Unity3D script will pass the setting parameters (e.g., camera resolution and the size of the area the application is interacting with) to the Starry Night plugin. The plugin will provide everything related to the mesh. In the Unity3d script, only the global point cloud and current point cloud are required from the plugin. Section 5.1 shows the file structure associated with using the Starry Night plugin (C# is used for Unity3D script).

1. Programming in the Unity Development Environment
   1. File Structure for Plugin
      1. The first step to using the Starry Night plugin is to put the plugin into your Unity project. Copy the whole folder “Plugins”, and paste it into the folder, “\...\Assets\” for your Unity project. An example is shown in the figure below.





* + 1. In the picture above, the files SLAM.cs and SLAMDemo.cs contain source code for providing a demonstration example. The source code in these files is available solely for the purpose of providing a demonstration example and is not to be used for any other purpose. See Section 6.0 for details on the demonstration example.
    2. All source code you develop for your specific application must be placed in the Assets folder as shown in the above picture.
  1. Overall Process

**Unity3D Script**

Display Global Point Cloud

“GlobalPTS ()”

Display Current Point Cloud

“SourcePTS ()”

Initialize SLAM

“InitializeSLAM()”

Stop SLAM

“StopSLAM()”

Start SLAM

“StartSLAM()”

Get Camera Position

“CamPos()”

**LOOP**

Note: The names within quotes are the functions within the Starry Night plugin which can used for any applications.

* 1. Function Calls

The Starry Night plugin is specifically designed for the development of Android based applications using the Unity3D development platform. Before a user’s application calls the Starry Night plugin, make sure the generic instances of java.lang.Class and java.lang.Object are defined as the interface to the plugin.

AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");

AndroidJavaObject jo = jc.GetStatic<AndroidJavaObject>("currentActivity");

* + 1. Initialize Starry Night Plugin

jo.Call("InitializeSLAM");

* + 1. OBJ File Path

// get the obj file path in the Android device

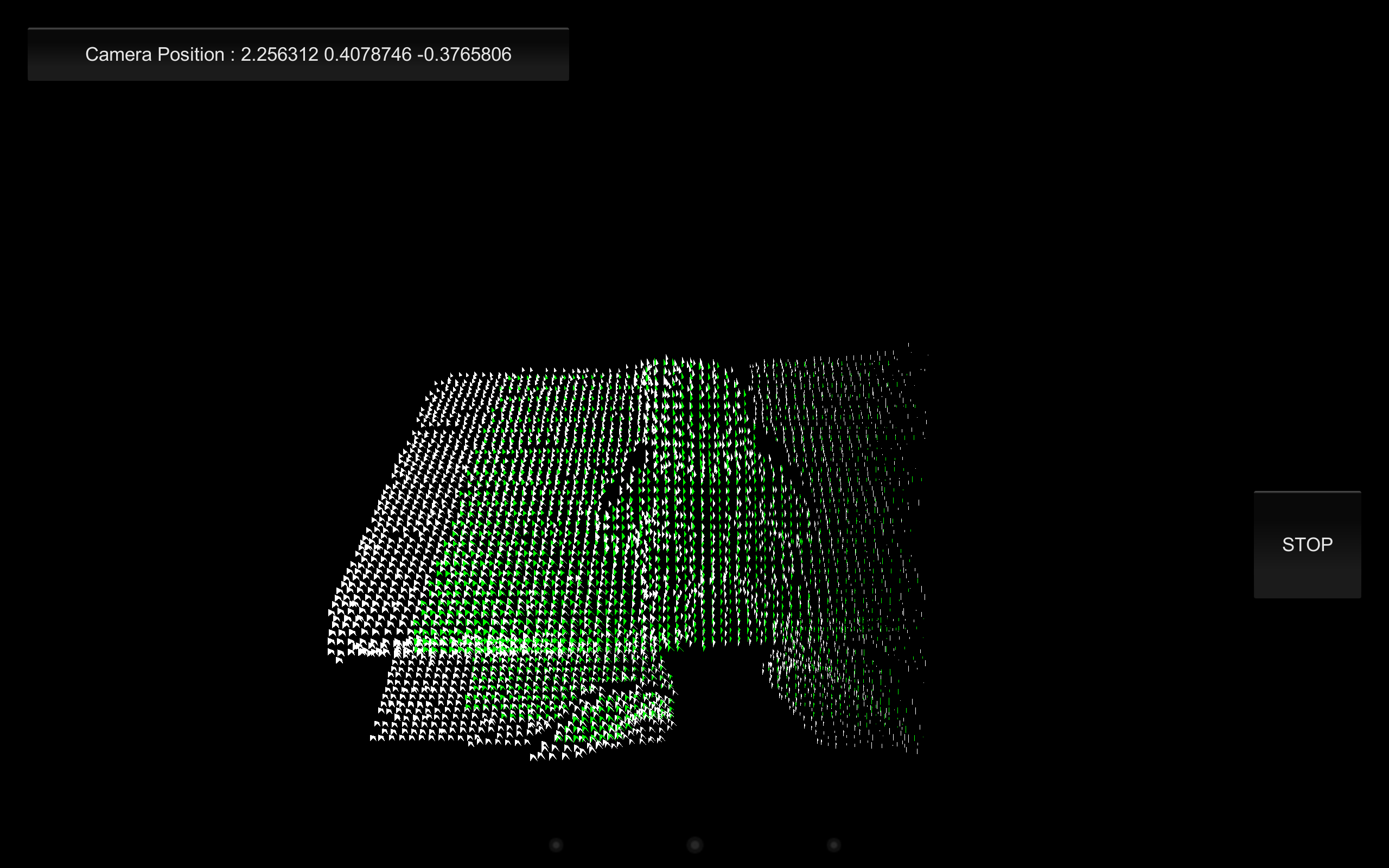
jo.Call<string>("MeshFilePath");

* + 1. Start SLAM and Return Health Status

// Start SLAM, return real-time status to check if SLAM result is good or not

bool bRet = jo.Call<bool>("StartSLAM");

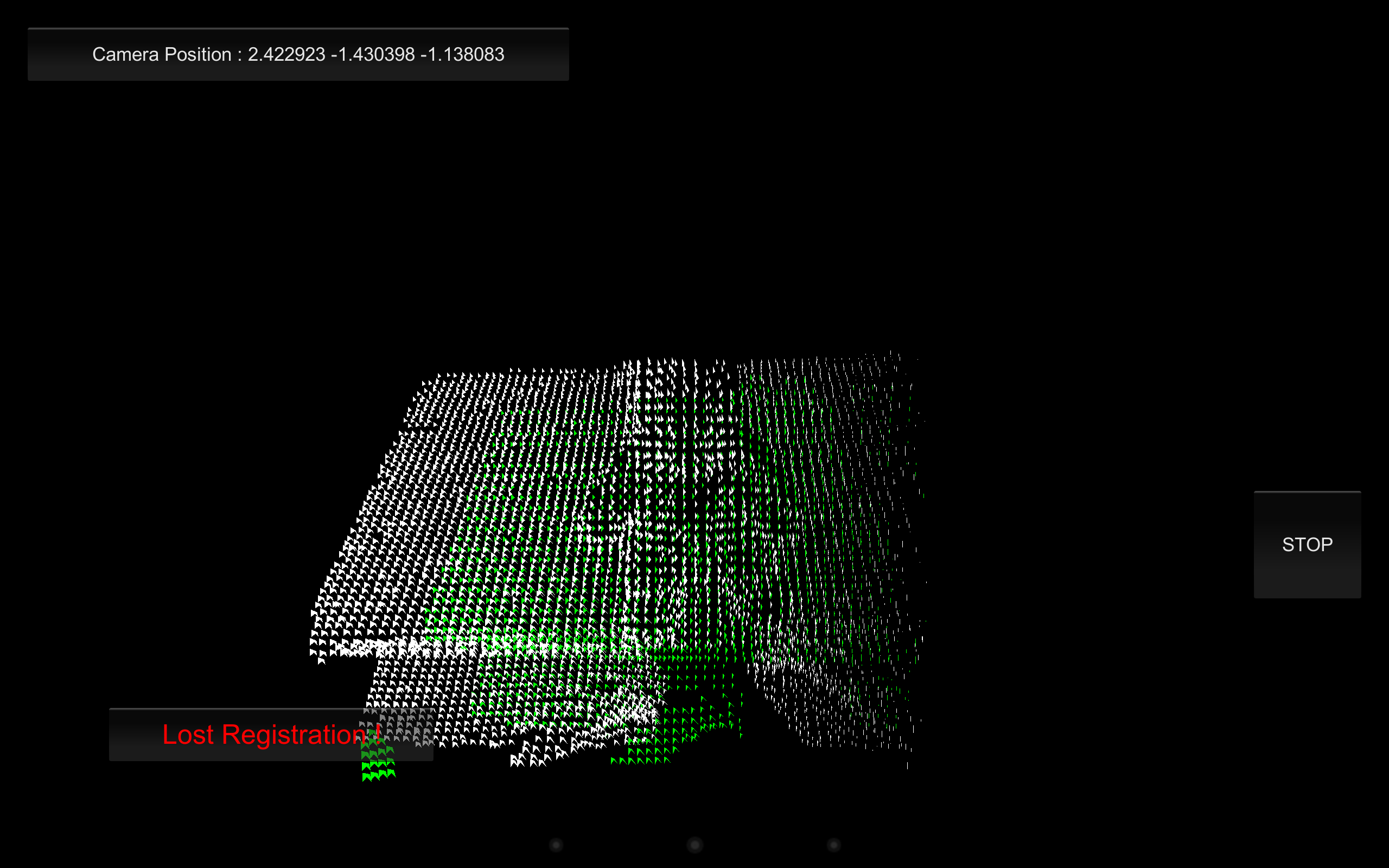
// if bRet = true ---> SLAM is running very well which means the current //point cloud (in green) overlaps the global point cloud (in white). See image below from the demonstration apk provided as a reference.



Current Point Cloud

Global Point Cloud

// if bRet = false ---> the current scan fails and error message will show at the //bottom-left corner, “Lost Registration!”. (Move camera back to the previous //position and make sure the green point cloud overlaps the white point cloud)

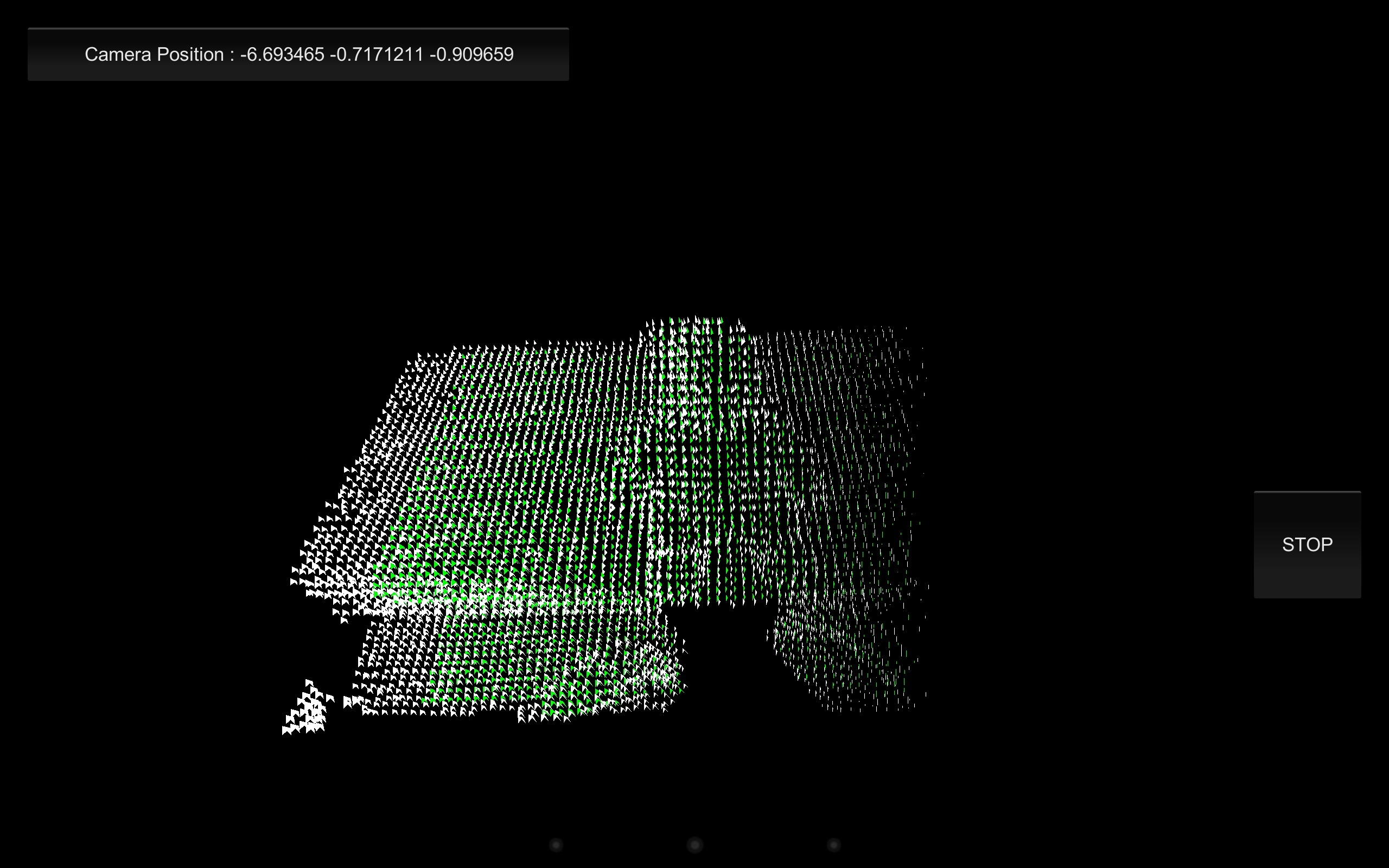


Error Message

Current Point Cloud

Global Point Cloud

Bad result (Error message)



Good result (No error message)

* + 1. Global Point Cloud

// get global point cloud data with the format (p0\_x, p0\_y, p0\_z, p1\_x, p1\_y, //p1\_z……)

jo.Call<float[]>("GlobalPTS");

* + 1. Current Point Cloud

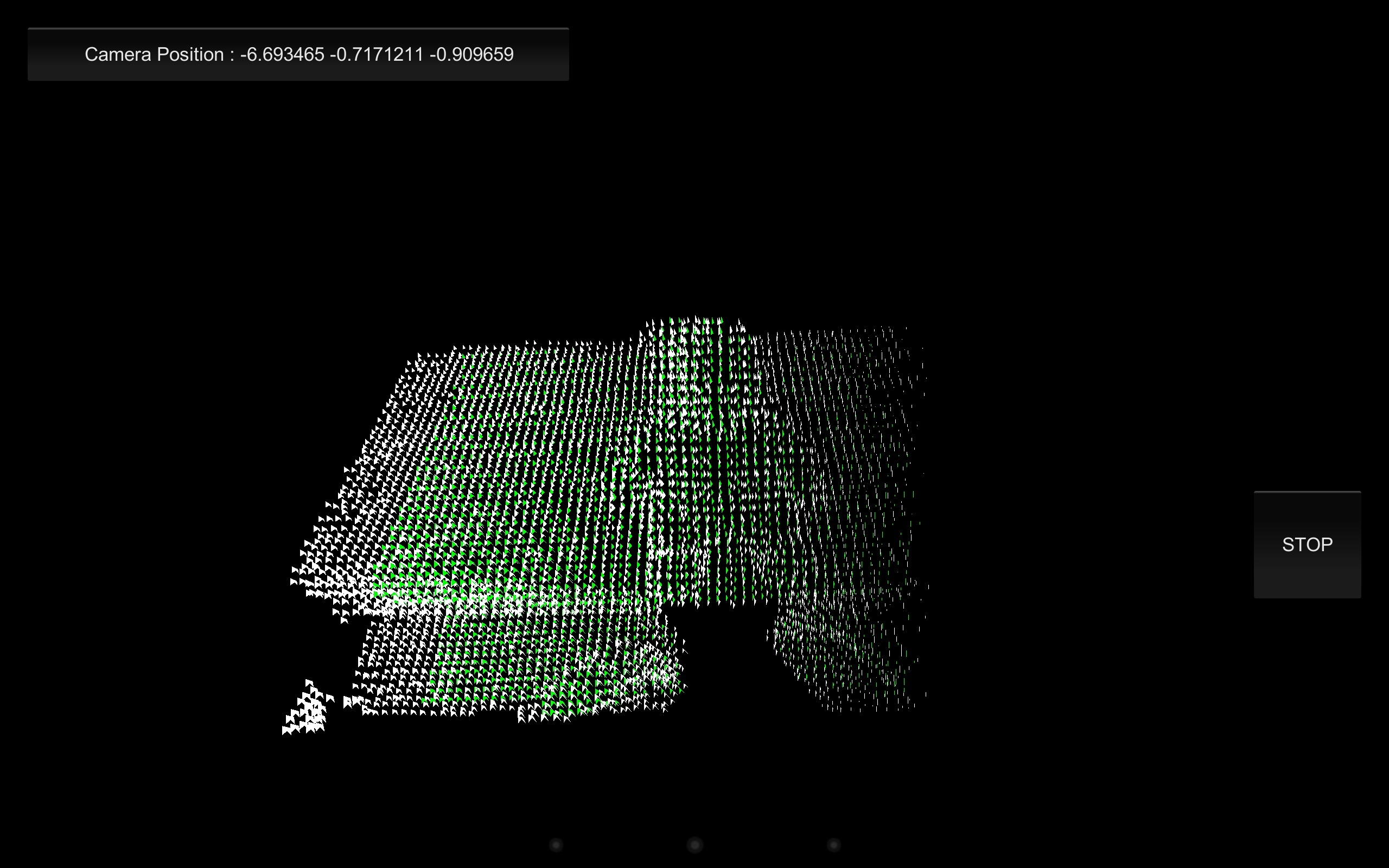
// get current point cloud data with the same format (p0\_x, p0\_y, p0\_z, p1\_x, //p1\_y,p1\_z……)

jo.Call<float[]>("SourcePTS");

* + 1. Current Camera Location

// get the current camera location with the format (x,y,z)

jo.Call<float[]>("CamPos");



Current Camera Position

* + 1. Stop SLAM

// stop SLAM

jo.Call(“StopSLAM”);

1. Demonstration Example

The demonstration software, SLAMDemo, is included both as an executable and as the source code to act as a reference guide for using the Starry Night plugin. The apk file, SLAMDemo.apk, can be directly installed in the Android device which is running Android OS 3.1 or higher. The demo shows how SLAM is used to provide the user with the ability to scan a scene while providing a point cloud display in real time. A global mesh is also created after the scene is captured.

* 1. Start Demo

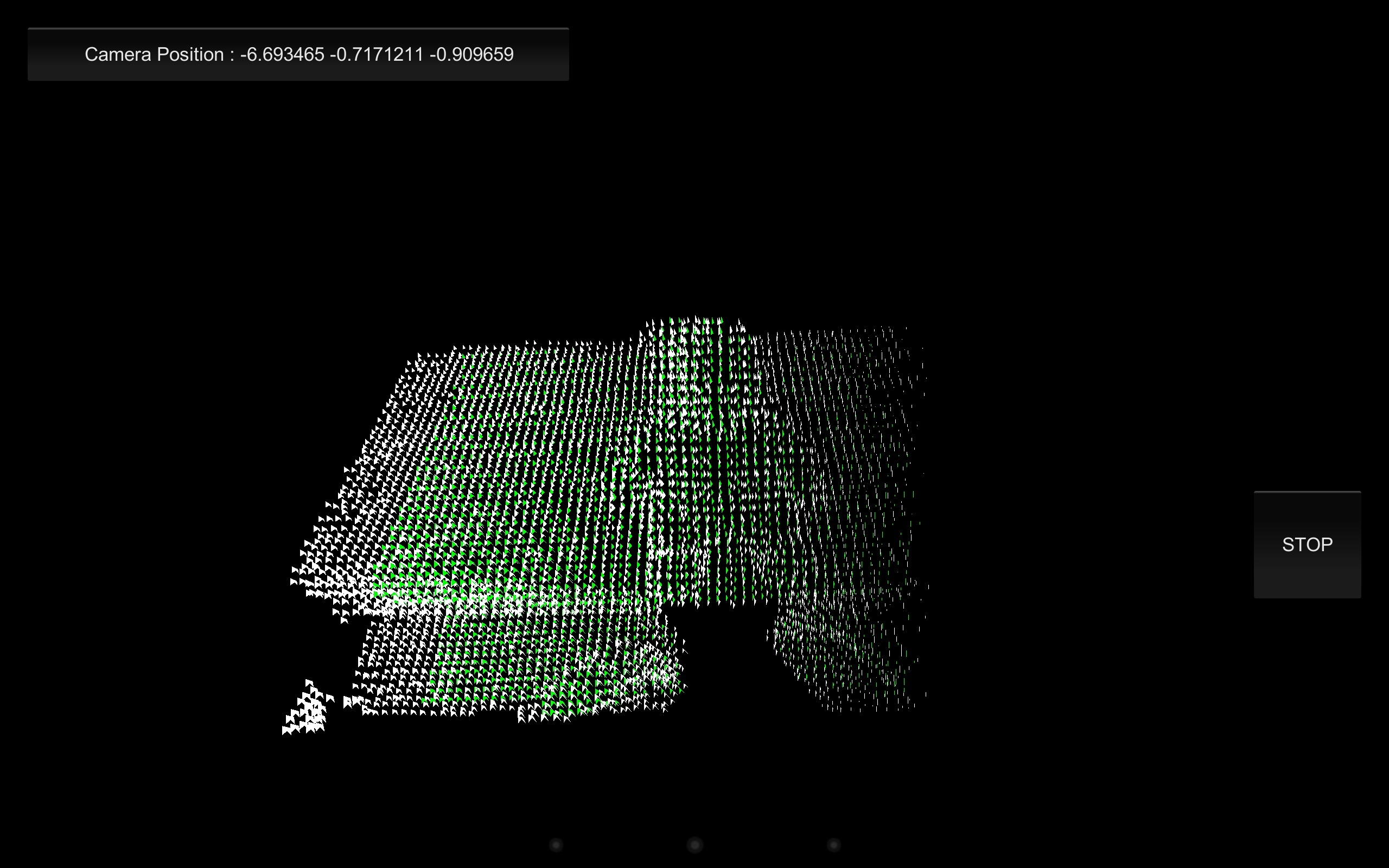
To start the demo software, point the sensor in the direction of the scene of interest, hit the “START” button (as shown in the screenshot below), and then slowly scan the scene. For the purpose of the demo software, the distance between the sensor and the scene of interest should be between 50 and 130 cm. Also, after starting the demo software the sensor should be moved very slowly to avoid a loss of registration. If registration is lost the display will show “LOST REGISTRATION!” as shown in the second figure in Section 5.3.3 above. Registration can be reestablished by slowly “backtracking” within the scene.



Hit “START” button

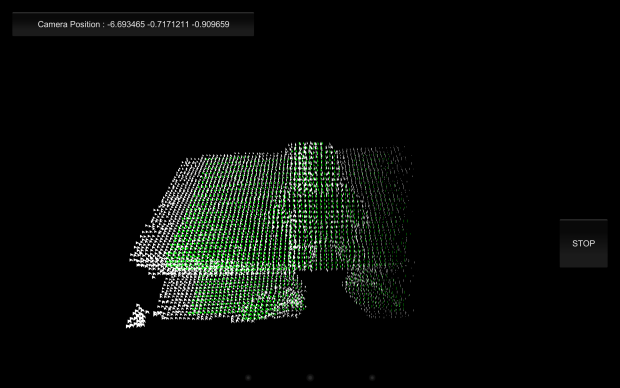
* 1. Scan Scene

While scanning the scene, the real-time scan is displayed in green while the global model is displayed in white.



* 1. Stop Demo

Scanning is stopped by hitting the STOP button as shown in the first screenshot below. After scanning is halted, the global point cloud is displayed as shown in the second screenshot below.

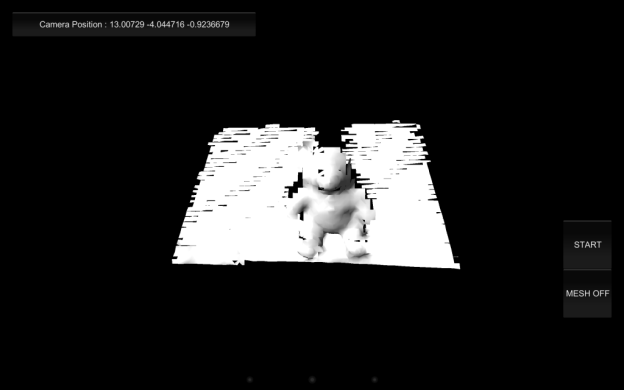
 

Hit “STOP” button

Global Point Cloud

* 1. Display Global Mesh

After the global point cloud is displayed as shown in the screenshot above, the user can hit the “MESH ON” button to display the global mesh as shown in the screenshots below.

Hit “MESH ON” button