LPVR-CAD
LPVR-DUO

Manual

Version 2.3

LIFE PERFORMANCE RESEARCH

CONTENTS

VERSION HISTORY

Version	Date	Changes
1.0	2018/4/2	Initial version
1.1	2018/6/20	Added VIVE Pro content. Added JSON configuration details.
1.2	2018/6/29	Inserted Optitrack rigid body setup tutorial (now removed again)
1.3	2018/7/2	Added preliminary pivot point calibration instructions
1.4	2019/4/19	Added hand controller assembly instructions
2.0	2019/9/29	- Updated description of configuration file - Added instructions for calibration - Added details about differential IMU operation
2.1	2019/11/5	- Updated parameter file description - Added more detailed explanation of ART rigid body adjustment - Added hand controller holder coordinate system alignment - Added description of dongle operation
2.2	2019/12/9	- Removed dongle status related information. Without installing Gemalto software status dialog can't be opened. - Added ART room coordinate system description - Corrected AxisPermutation settings
2.3	2019/12/11	Description on how to activate VIVE controller support added

OVERVIEW

Consumer virtual reality head mounted display (HMD) systems such as the HTC VIVE support room scale tracking. These systems can track head and controller motion of a user not only in a sitting or other stationary position, but support free, room-wide motions. The volume of this room scale tracking is limited to the capabilities of the specific system, covering for instance around 5m x 5m x 3m in case of the VIVE Lighthouse tracking system (version 1). Whereas for single user games and applications this space is usually sufficient, especially multi-user applications such as arcade-style game setups or enterprise applications require larger tracking volumes.

Multi-camera-based optical outside-in tracking systems offer tracking volumes of up to 15m x 15m x 3m or more. Although the positioning accuracy of optical tracking systems are in the sub-millimeter range, especially orientation measurement is often not sufficiently fast and accurate to provide an immersive experience to the user. Image processing and signal routing may introduce further latency. With LPVR we solve this problem by combining optical tracking information with inertial measurement data using specially tuned predictive algorithms.

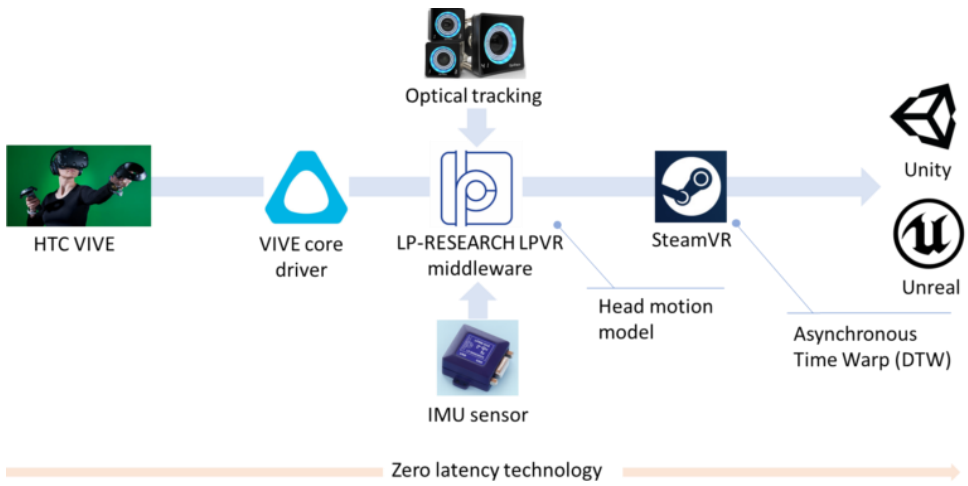

Figure 1 - Overview of the LPVR tracking system. Central to the system is the LPVR middleware that fuses data from IMU and optical tracking.

We started development of LPVR as a plug-in system for the Unity or Unreal game engine. Later on we extended our solution to provide a full OpenVR / SteamVR driver. This means that on the one hand any existing SteamVR application automatically supports the arbitrarily large tracking areas covered by the optical tracking system. On the other hand, no additional plugins for Unity, Unreal or your development platform of choice are required - support is automatic. Responsive behavior is guaranteed by using LP-RESEARCH's IMU technology in combination with low-latency VR technologies like asynchronous time warping, late latching etc. An overview of the functionality of the system is shown in Figure 1.

SYSTEM VERSIONS

We offer LPVR in two different versions:

Version name	Description
LPVR-CAD	LPVR-CAD is our solution for location-based VR, to enable large scale tracking volumes. It combines data from a headset IMU and an optical tracking system to output low-latency, accurate 6-DOF information. LPVR-CAD additionally supports VIVE handcontrollers.
LPVR-DUO	LPVR-DUO is our in-vehicle VR/AR tracking solution. It uses two IMUs, one attached to the headset and one attached to the vehicle itself.

NOTE: Both versions are based on the same driver infrastructure. Therefore, most parts of this manual, except where noted otherwise, apply to both system versions.

LPVR currently works with the following VR headsets:

Headset name	Support	Driver mode
HTC VIVE	LPVR-CAD, LPVR-DUO	SteamVR / OpenVR
HTC VIVE Pro	LPVR-CAD, LPVR-DUO	SteamVR / OpenVR
VARJO VR-1/2	LPVR-CAD, LPVR-DUO	VARJO driver plugin

LPVR MIDDLEWARE DETAILS

Building on the technology that we developed for our IMU sensors and large-scale VR tracking systems, we have created a full motion tracking and rendering pipeline for virtual reality (VR) and augmented reality (AR) applications. This middleware as shown in Figure 2 is a full solution for AR / VR that enables headset manufacturers to easily create a state-of-the-art visualization pipeline customized to their product. This middleware is the core of LPVR. Specifically, the middleware offers the following features:

- Flexible zero-latency tracking adaptable to any combination of IMU and optical tracking
- Rendering pipeline with motion prediction, late latching and asynchronous timewarp functionality
- Calibration algorithms for optical parameters (lens distortion, optical see-through calibration)
- Full integration in commonly used driver frameworks like OpenVR and OpenXR possible
- Specific algorithms and tools to enable VR / AR in vehicles (car, plane etc.) or motion simulators

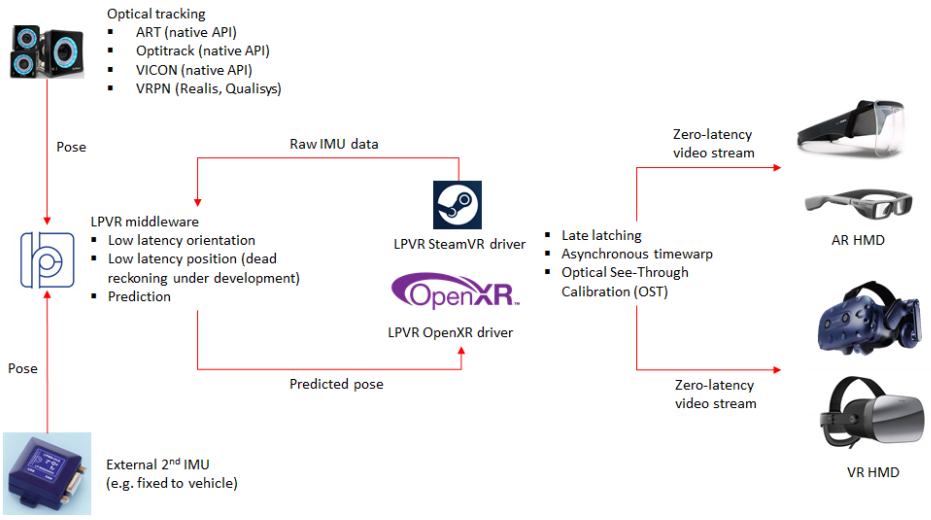

Figure 2 - Overview of complete LPVR middleware functionality.

APPLICATION OF LPVR MIDDLEWARE TO IN-CAR VR / AR (LPVR-DUO)

LPVR's tracking backend is especially advanced in the aspect that it allows the flexible combination of multiple optical systems and inertial measurement units (IMUs) for combined position and orientation tracking.

For in-vehicle VR/AR, the standard lighthouse tracking system of the VIVE is to be replaced by an alternative solution that can be operated inside the vehicle, specifically when it is in motion. The VIVE's built-in tracking mechanism works based on a fusion of information from an inertial measurement unit (IMU) mounted inside the headset and the lighthouse optical system. In a static environment this solution works well.

In case of a moving vehicle, data from the headset IMU contains a combination of vehicle orientation information and movement of the user's head. If those two types of movement cannot be separated before sending orientation data to the HMD's graphics rendering pipeline, as a result a wrong image will be displayed to the user.

Specifically, for this purpose we developed the LPVR middleware to enable the de-coupling of the head motion of a user and the motion of a vehicle the user might be riding in, such as a car or airplane. As shown in the illustration below, using a double-IMU or differential IMU calculation, the interior of a vehicle can be displayed as static relative to the user, while the scenery in the environment of the vehicle moves with vehicle motion.

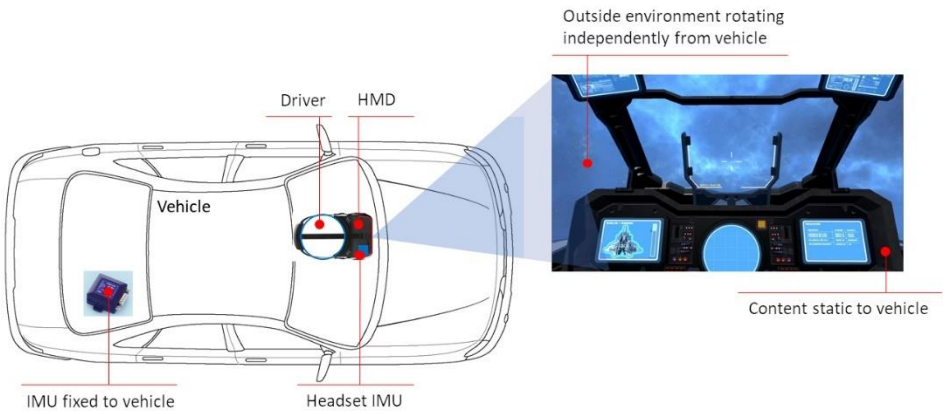

Figure 3 - Differential IMU operation allows the de-coupling of user head motion and vehicle rotations. In the following explanations the IMU attached to the headset will be referred to as headset IMU and the IMU fixed to the vehicle as vehicle-fixed or reference IMU.

One important focus for us when developing this system was to find a methodology to reduce latencies and mathematically enhance optical tracking data for the given application. Additionally, specially filtered inertial measurement data was added to the output signal to improve system reaction times without being negatively by the car's external motion.

DRIVER IMPLEMENTATION

STEAMVR DRIVER WRAPPER

The VIVE driver environment is based on the SteamVR infrastructure. The SteamVR driver for the VIVE implements the OpenVR API and is therefore compatible to a large variety of end-user applications. The first goal of our development was to create a replacement for the standard SteamVR VIVE hardware driver that allows modification of specific aspects of the system. The fundamental functionality of the driver and most of its mechanisms related to rendering graphics content are to be preserved. Specifically, we intend to modify the head tracking features of the driver as written below.

SteamVR is a hardware driver created by the company Valve Inc. to support the VIVE HMD hardware and other headsets. It exposes the OpenVR API and can therefore work with all applications supporting virtual reality setups that are OpenVR-compatible.

OpenVR is open-source and is hosted on Github: `https://github.com/ValveSoftware/openvr`

As shown in Figure 4 external software accesses the headset hardware using the OpenVR application API. SteamVR provides an interface for this access and translates commands to the headset hardware.

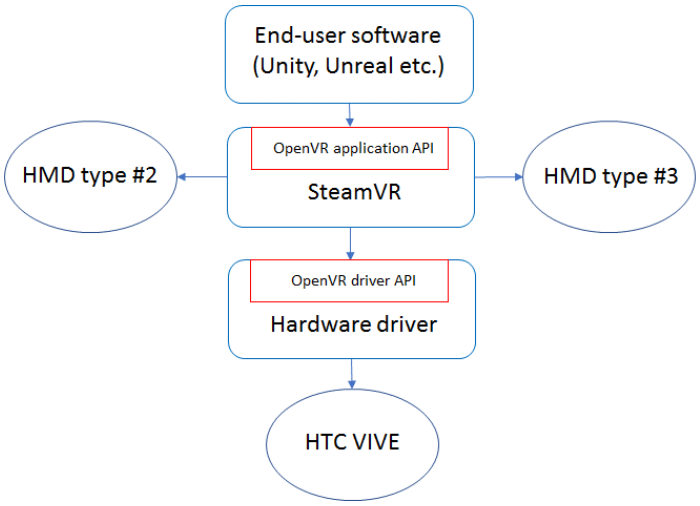

Figure 4 - The OpenVR system model has two API stages: the OpenVR application API and the driver API to talk to the headset hardware. Many types of headsets can be supported.

Besides the OpenVR application API, there exists an OpenVR driver API. This driver API is accessed by SteamVR to communicate with headset hardware. To work with an alternative HMD therefore a driver exposing this OpenVR driver API must be created and connected to SteamVR.

For the standalone driver, the functionality of the original VIVE hardware driver was recreated such that the hardware can be directly operated without using an intermediate third-party driver, while the replacement of the lighthouse position and orientation tracking system and implementation of a custom lens distortion method. The standalone driver supports both the orgiginal HTC Vive and the HTC Vive Pro.

TRACKING AND SENSOR FUSION

Key component of the LPVR tracking system is the pose combiner that merges absolute pose data from e.g. an optical tracking system with inertial measurement / IMU data. The diagram below shows an overview of this functionality.

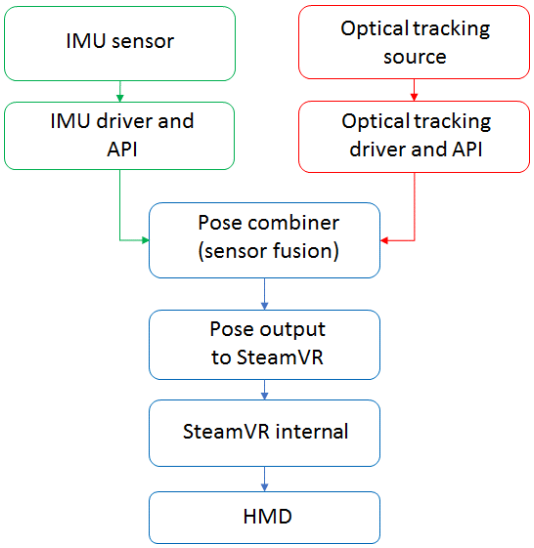

Figure 5 - The pose combiner fuses IMU sensor and optical tracking data.

SYSTEM SETUP

HARDWARE INSTALLATION

The system consists of the following hardware components:

- VR/AR headset with built-in or externally attached IMU
- Optical tracking system, preferredly an Advanced Realtime Motion System (ART), either installed in large room scale installation, seating buck (Sitzkiste) installation or in-vehicle installation.
- For in-vehicle installations (LPVR-DUO), an external IMU, preferredly LPMS-IG1P is required.

The optical tracking system needs to be installed as defined by the optical tracking system manufacturer. Please take care that the floor of the tracking space is calibrated vertically. A water balance built-into the clibration equipment should assist with that.

In case an external IMU is used for HMD tracking, this IMU needs to be installed inside or on the headset. It needs to be attached to the PC running LPVR via a USB connection.

In case of differential IMU tracking, we need to install a further IMU inside the vehicle, in a location visible to the optical tracking system. Alternatively, this IMU can also be installed in a location not visible to the optical tracking, but with exactly known orientation within the optical reference system.

We will provide further instructions on how to calibrate these components to work in the same coordinate system in the following chapters.

DRIVER SETUP

SYSTEM REQUIREMENTS

1. LPVR in its current version runs exclusively on Windows, from Windows 7 upwards.
2. SteamVR must be installed and operational. We are monitoring SteamVR version updates and try to follow with software updates as soon as we seen any API change by VALVE.

   NOTE: Once you have a running system, it makes sense to prevent automatic SteamVR updates by disconnecting your system from the internet or not starting the Steam client. SteamVR can run separately without the Steam client running.
3. Currently LPVR natively works with Advanced Realitime Tracking (ART), Optitrack NatNet and VICON. All other tracking systems need to support VRPN to be able to operate with our driver.
4. This driver plugs into SteamVR. You won't need any modifications to your Unity, Unreal or other SteamVR application code. Please use the standard SteamVR interfaces to support headset tracking. To connect a VRPN host, you need to pass the server Information and Sensor ID to the driver.

DRIVER INSTALLATION

1. Unpack the driver to a directory. It contains the SteamVR driver in the sub-directory *alighthouse* (named so for technical reasons). The directory where you unpacked the driver will be called *%path_to_alighthouse%* below.

You can copy and paste it to the command line easily by opening the folder in the Windows Explorer, and then selecting "Copy address as text" from the menu that appears after you right-click on the location bar.

2. We now need the `vrpathreg.exe` utility from your SteamVR installation. Typically, it's located in:

   C:/Program Files (x86)/Steam/steamapps/common/SteamVR/bin/win32/

   This directory is referred to as *%path_to_vrpathreg%* below. To setup the driver, open a command line and enter:

   "%path_to_vrpathreg%/vrpathreg.exe" adddriver "%path_to_alighthouse%/alighthouse"

   Depending on your system settings, you may need administrator rights for this step. If you run `vrpathreg` without arguments, it should now also print the path that you entered. Please be aware that the path is checked only at the time when SteamVR tries to load the driver.

3. Insert the LPVR dongle into a free USB slot of your PC. If the dongle is not inserted, the driver will not start, and an error message will be displayed.

   NOTE: If you are using the LPVR trial version, you can use the application for a limited duration without having a dongle.

4. Restart SteamVR. Check the logfile *vrserver.txt* (available from SteamVR's System Report), there should be several messages sent by `alighthouse`. If the messages are sent by `lighthouse` instead, please check for an error from `alighthouse` near the top. Pointing your browser to *http://localhost:8998/console/index.html* (SteamVR versions from Nov 2019): *http://localhost:27062/console/index.html*) will show you the log output of the SteamVR server.

5. Once the driver has successfully loaded, you can try to connect to *http://localhost:7118/index.html* in your web browser. The driver will allow you to enter a configuration in JSON format there. Also, if you put on your headset it should track your head's orientation even if you didn't configure any position tracking.

DRIVER REMOVAL

1. Run *vrpathreg*, but use the *removedriver* command:

   "%path_to%/vrpathreg.exe" removedriver "%path_to%/alighthouse"

   Paths must be entered as above. The best thing to do is to call *vrpathreg* without arguments and copy and paste from there as the path must be literally identical for the driver to be removed.

DRIVER CONFIGURATION

CONFIGURATION INTERFACE

Once the driver is correctly installed it should be started automatically when launching SteamVR. With a valid driver configuration any application using SteamVR as well as any SteamVR content should run out-of-the-box.

Depending on your system configuration you should receive a default configuration file from us. To change the configuration manually, direct your browser to *http://localhost:7118/index.html* after lauching the driver.

NOTE: In some cases, the driver doesn't start up correctly and the configuration page won't be available. In this case, edit *%path_to_alighthouse%*\resources\configuration\settings.json directly. Pointing your browser to *http://localhost:8998/console/index.html* (new SteamVR versions: *http://localhost:27062/console/index.html*) will show you the log output of the SteamVR server. Use this information to debug the driver configuration. Enter *alighthouse* in the filter bar to see only log output related to the LPVR driver. After manually editing the configuration file restart SteamVR to activate the new configuration.

<table>
<tr><th>Button Name</th><th>Function</th></tr>
<tr><td>Driver status</td><td>Shows status of LPVR driver:<table><tr><td>Green</td><td>Driver OK</td></tr><tr><td>Yellow</td><td>Driver running, but some data sources not ready</td></tr><tr><td>Red</td><td>Driver not running</td></tr></table></td></tr>
<tr><td>Load a JSON file</td><td>Loads an external configuration file.</td></tr>
<tr><td>Save configuration to JSON file</td><td>Exports current LPVR configuration to JSON file.</td></tr>
<tr><td>Load currently active configuration</td><td>Loads the current LPVR configuration into the editor window.</td></tr>
<tr><td>Push current settings to driver</td><td>Loads current settings as shown in the editor window to driver. Push this button after making changes to the JSON code and to activate them in the driver.</td></tr>
</table>

The table above describes the functionality of the various buttons of the configuration interface. Figure 6 shows the configuration interface itself.

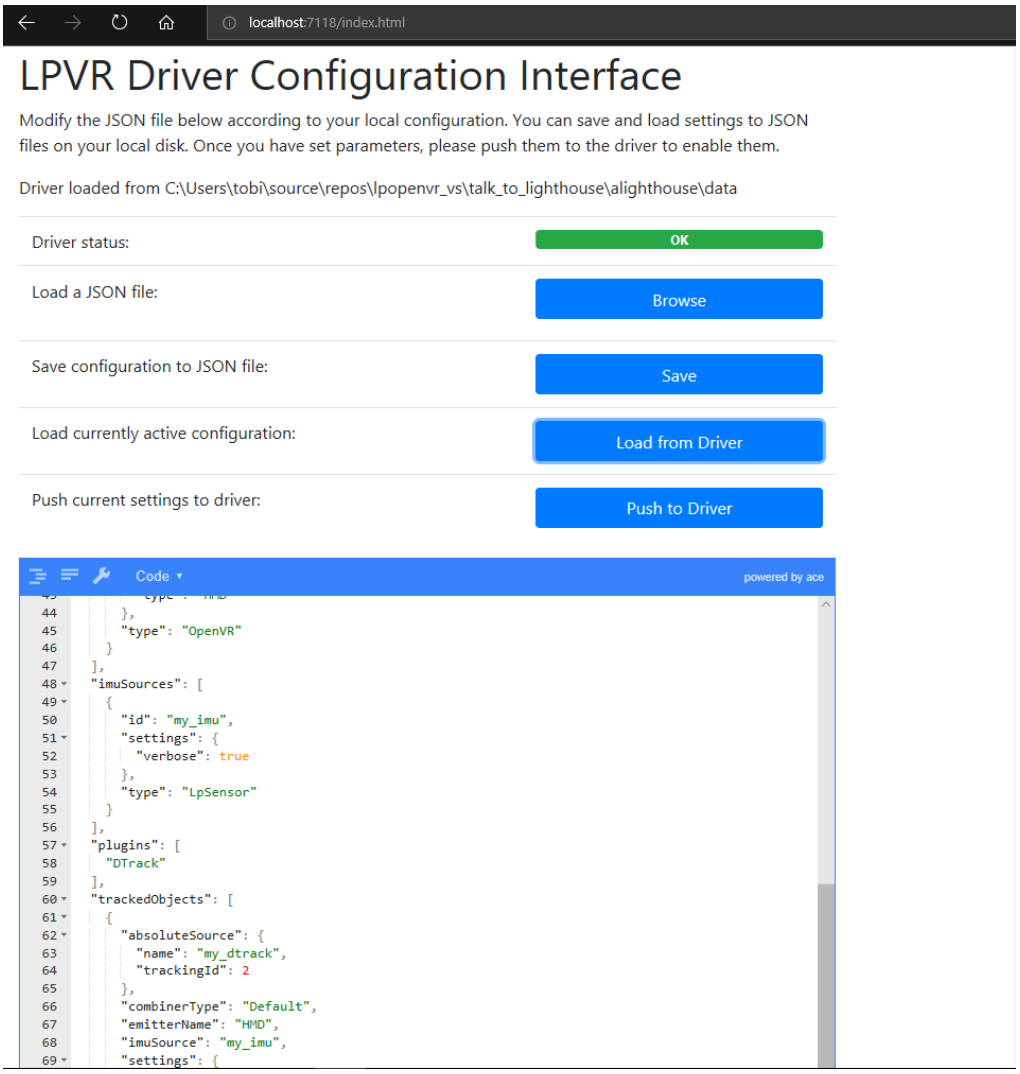

Figure 6 - Screenshot of the LPVR driver configuration interface.

In the following we will describe the possible contents and parameters of the configuration script. The script is in JSON format and requires the corresponding syntax.

JSON FILE STRUCTURE OVERVIEW

The JSON configuration file defines the input/output and processing components within LPVR. More specifically there are on overall header tag and four sub-module types:

	JSON Tag Name	**Function**
1	*PoseMachineConfig*	Header tag that preceeds the definition of all sub-modules
2	*absoluteSources*	Defines the absolute positioning (optical tracking) devices used to acquire global poses
3	*imuSources*	Defines sources for IMU orientation data

4	*trackedObjects*	Defines how objects are tracked i.e. their data sources and how the different data sources are combined
5	*emitters*	Defines the output target of the sensor fusion done for trackedObjects. This is usually an HMD.

The configuration file consists of these components to define the flow of input signals to output signals. The input to a typical system would consist of an absoluteSources structure for an optical tracking input, one or more imuSources and one or more emitters to output the information. All input data is combined in the trackedObjects structure that defines the parameters of the sensor fusion and signal routing. Figure 7 shows an overview of the structure of a configuration file that works for differential IMU headset tracking.

NOTE: The outer-most structure *"PoseMachineConfig" : { }* should only be added when editing *settings.json* directly. When editing the configuration in the integrated editor on the configuration page, needs to be left away.

Figure 7 - Overview of the structure of a typical LPVR JSON file for LPVR-DUO. This specific setup is based on ART for optical tracking, the VIVE internal IMU as headset IMU and an LPMS-CU2 as vehicle-fixed IMU. LPVR-CAD uses eactly the same file structure, just without the reference IMU part.

OPTICAL TRACKING SOURCES

The absoluteSources tag describes an absolute position and orientation source like an optical tracking system. Currently LPVR supports VICON, Optitrack and ART tracking system natively, as well as the common VR communication protocol VRPN.

Optical Tracking System	Example Code Block	Explanation	

ART	`"absoluteSources": [`
	`{`
	`"name": "my_dtrack",`
		name	Defines the name of the source. Any name is good.

	<pre>"settings": {
	"axisPermutation": "xyz",
	"host": "192.168.1.38",
	"port": 5000
	},
	"type": "DTrack"
	}
	]</pre>
		settings: axisPermutation
		
		settings: host
		
		settings: port
		
		type
			Optional axis permutation setting to adjust coordinate system
			
			Address of the host PC
			
			Port number of host PC
			
			Must be **DTrack**

VICON	<pre>"absoluteSources": [
	{
	"name": "my_vicon",
	"settings": {
	"host": "192.168.1.6:801",
	"segmentName": "HMD",
	"subjectName": "HMD"
	},
	"type": "Vicon"
	}
	]</pre>
		name
		
		settings: host
		
		settings: segmentName
		
		settings: subjectName
		
		type
			Defines the name of the source. Any name is good. Will be referenced in trackedObjects further down.
			
			IP address of the VICON host computer running VICON Tracker, Blade etc.
			
			Name of the rigid body in VICON software.
			
			Should be the same as the segmentName
			
			Must be **Vicon**

Optitrack	<pre>"absoluteSources": [
	{
	"name": "my_optitrack",
	"settings": {
	"connectionType": "Multicast",
	"localAddress": "127.0.0.1",
	"remoteAddress": "127.0.0.1",
	"serverCommandPort": 1510,
	"serverDataPort": 1511
	},
	"type": "OptiTrack"
	}
	]</pre>
		name
		
		settings: connectionType
		
		settings: localAddress
		
		settings: remoteAddress
		
		settings: serverCommandPort
		
		settings: serverDataPort
		
		type
			Defines the name of the source. Any name is good. Will be referenced in trackedObjects further down.
			
			Must be **Multicast**
			
			Local address of the Optitrack client
			
			Address of the Optitrack server
			
			Must be **1510**
			
			Must be **1511**
			
			Must be **OptiTrack**

VRPN	<pre>"absoluteSources": [
	{
	"name": "my_vrpn",
	"settings": {
	"tracker": "DTrack@127.0.0.1"
	},
	"type": "VRPN"
	}
	]</pre>
		name
		
		settings: tracker
		
		type
			Defines the name of the source. Any name is good. Will be referenced in trackedObjects further down.
			
			Name and address of VRPN server
			
			Must be **VRPN**

IMU SOURCES

The imuSources tag describes the IMU attached to the headset. At the moment only the LP-REASEARCH LPMS IMU is supported.

Example Code Block	Explanation	

<pre>"imuSources": [
  {
  "id": "my_imu",
  "type": "ViveHeadset"
  },
  {
  "id": "reference_imu",
  "settings": {
  "name": "lpmscu2000327"
  },
  "type": "OpenZen"
  },
  {
  "id": "no_imu",
  "type": "None"
  }
]</pre>
	id	Defines the name of the source. Any name is good. Will be referenced in trackedObjects further down.

	type	Can either be - **OpenZen** to use OpenZen library - **ViveHeadset** to use internal Vive IMU - **LpSensor** to use LpSensor library (deprecated) - **None** for a dummy IMU that doesn't emit any data
	settings: name	Specifies the ID of the connected IMU (not required for ViveHeadset)

OUTPUT OF POSE CALCULATION

The orientation and position calculated by sensor fusion of IMU and optical tracking is output to a headset by an emitter. The emitter allows setting parameters determining the position and orientation offset between IMU coordinate system and the optical coordinate system of the headset.

Example Code Block	Explanation	

<pre>"emitters": [
  {
  "name": "HMD",
  "settings": {
  "imuToEyeQuat": {
  "w": 1,
  "x": 0,
  "y": 0,
  "z": 0
  },
  "imuToEyeVect": {
  "x": 0,
  "y": 0,
  "z": 0
  },
  "type": "HMD"
  },
  "type": "OpenVR"
  },</pre>
	name	Defines the name of the output device. Any name is good. Will be referenced in trackedObjects tag.

	HMD Emitter	Emits orientation to HMD
	settings: imuToEyeQuat	Rotation from IMU frame to the eye frame in which the graphics are rendered
	settings: imuToEyeVect	Translation from IMU frame to the eye frame in which the graphics are rendered
	settings: type	Must be HMD
	type	Must be OpenVR
	Console Emitter	Displays orientation as SteamVR console output

<pre>{
  "name": "console",
  "settings": {
  "interval": 10
  },
  "type": "Console"
}
]</pre>
	settings: interval Interval between log outputs in ms
	
	name Must be Console

POSE CALCULATION

The actual pose of an object is calculated by combining IMU data and optical tracking information. This tag combines the modules we defined above. The result pose is forwarded to the emitter block.

Example Code Block	Explanation

<pre>"trackedObjects": [
  {
  "absoluteSource": {
  "name": "my_vicon"
  "trackingId": 0
  },
  "combinerType": "DifferentialImu",
  "emitterName": "HMD",
  "imuSource": "my_imu",
  "settings": {
  "absoluteFromImuFrameQuat": {
  "w": 1,
  "x": 0,
  "y": 0,
  "z": 0
  },
  "absoluteFromImuFrameVect": {
  "x": 0,
  "y": 0,
  "z": 0
  }
  "ignoreGravity": true,
  "opticalWeight": 0.005,
  "referenceImu": "reference_imu",
  "referenceOrientationQuat": {
  "w": 1,
  "x": 0,
  "y": 0,
  "z": 0
  },
  "referenceToOpticalQuat": {
  "w": 1,
  "x": 0,
  "y": 0,
  "z": 0</pre>
	absoluteSource: name Name of the previously defined absolute source (VICON etc.).
	
	absoluteSource: trackingId ID of object tracked by optical system.
	
	combinerType Type of sensor fusion used.
	- **Default**: uses single-IMU fusion
	- **DifferentialImu**: uses differntial dual-IMU operation (LPVR-DUO)
	
	emitterName Name of emitter to output data to
	
	imuSource Name of IMU source declared above (headset IMU)
	
	settings:absoluteFromImuFrameQuat Orientation of the tracked body frame relative to the IMU frame
	
	settings:absoluteFromImuFrameVect Translation of the tracked body frame relative to the IMU frame
	
	settings: ignoreGravity If **true**, acceleromter data of the headset IMU is not used to correct HMD pitch and roll orientation.
	This should be true for in vehicle applications and false for stationary installations.
	
	settings: opticalWeight Impact of the optical orientation tracking on oprientation

<pre> } } }]</pre>	measurements. Default = 0.005 **settings: reference_imu** Name of the IMU to be used as reference (fixed to vehicle) IMU **settings: referenceOrientationQuat** Orientation of the reference IMU body inside the optical tracking space **settings: referenceToOptiCalQuat** Rotation to translate from reference IMU internal coordinate system to optical tracking coordinate system

SYSTEM CALIBRATION

ADJUSTMENT OF HMD RIGID BODY

Perform the following steps to adjust and calibrate the HMD rigid body and its attached IMU:

1. Make sure that your ART room calibration is configured to “Power wall” in DTrack as shown in Figure 8. In this case the `"axisPermutation"` should be set to `"xyz"` in the `"absoluteSources"` block of the LPVR configuration file.

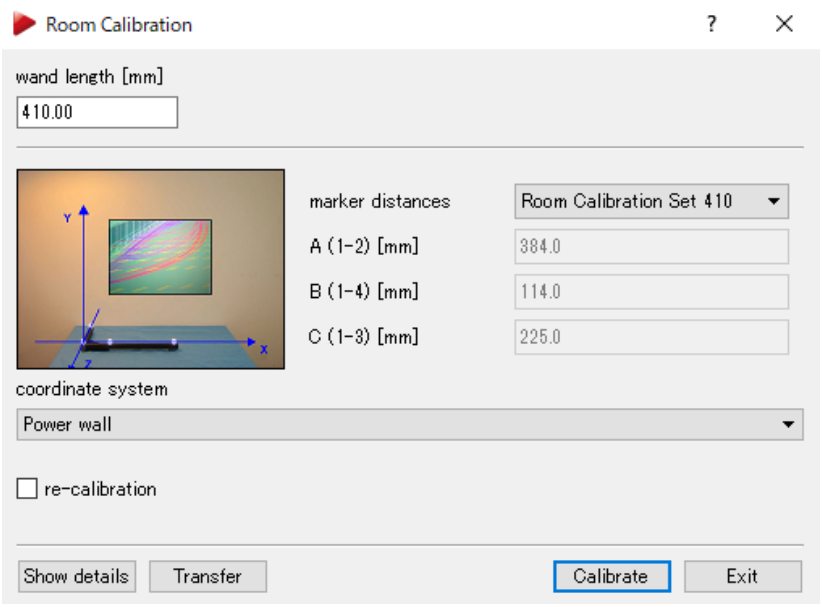

Figure 8 – Dtrack’s room calibration offers two coordinate system options. For our application Power wall is the correct configuration.

2. The rigid body in the optical tracking system software should be aligned to fit with the OpenVR alignment standard as shown below (Figure 9, Figure 10, Figure 11, Figure 12, Figure 13, Figure 14, Figure 15). The center (the position source) of the rigid body needs to be adjusted, so that it lies closely on the upper part of the nose, in the middle between the eyes

of the user, while wearing the headset. This position needs to be manually adjusted to be accurate. Direction of the user's gaze should be the Z-axis of the rigid body.

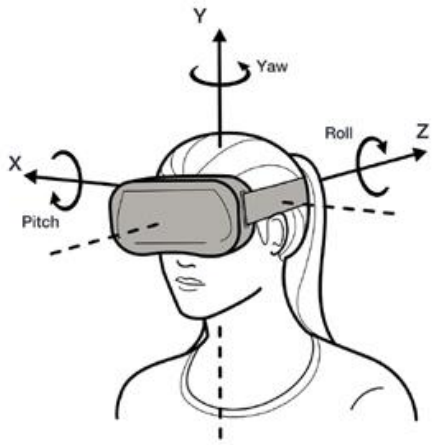

Figure 9 - OpenVR coordinate system alignment

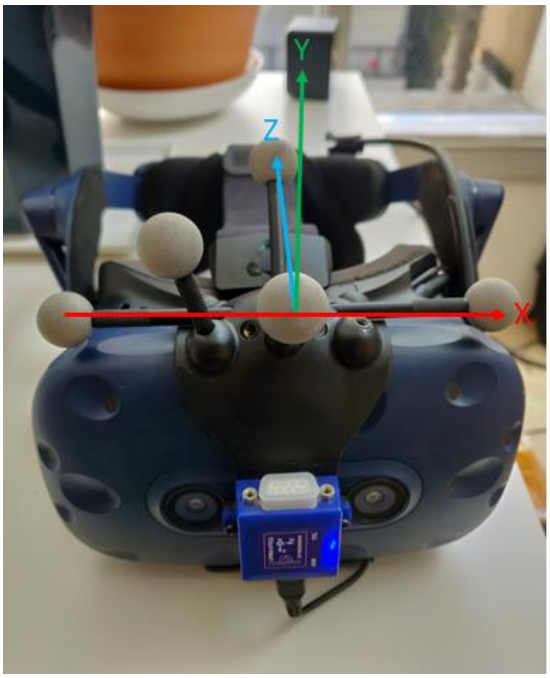

Figure 10 - Example marker arrangement, front view. Two markers (for each axis) defining a horizontal line parallel to the X and Y-axis of the HMD will make it easier to adjust the rigid body coordinate system in the optical tracking software.

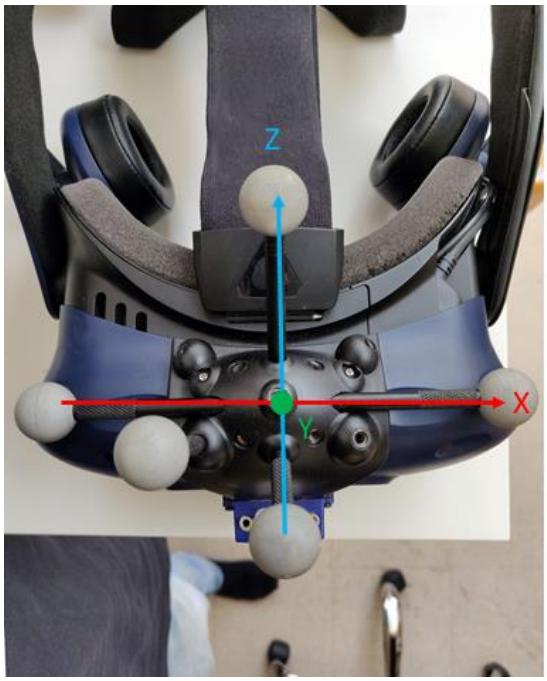

Figure 11 - Example marker arrangement, top view.

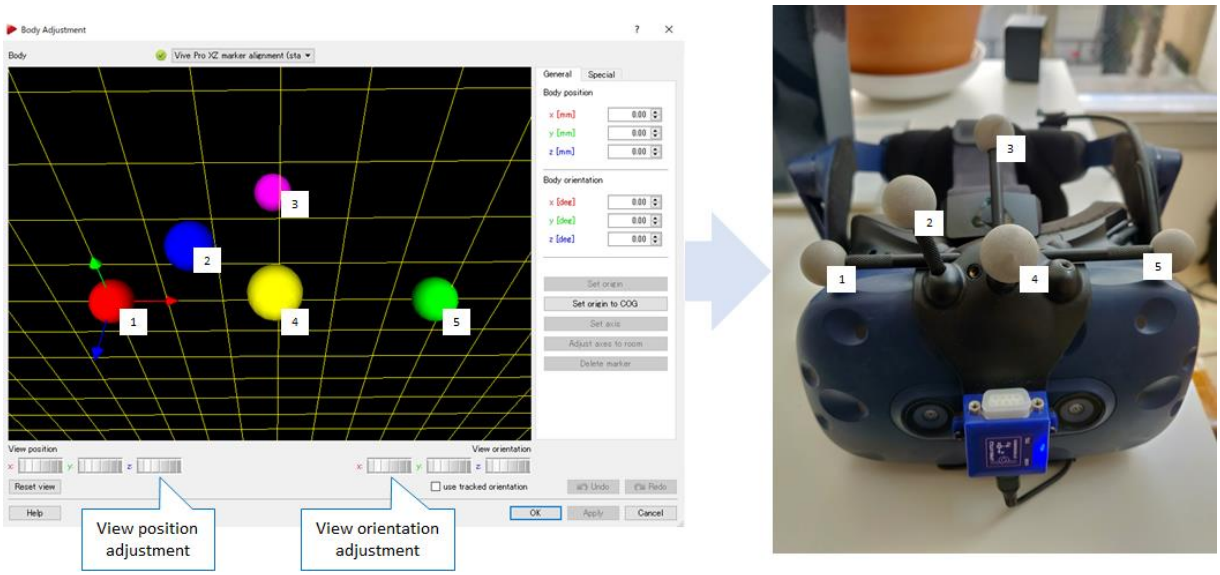

Figure 12 - As a first step in aligning the rigid body in ART, adjust the view so that you can visually assign marker balls on the HMD to markers detected by the ART body calibration.

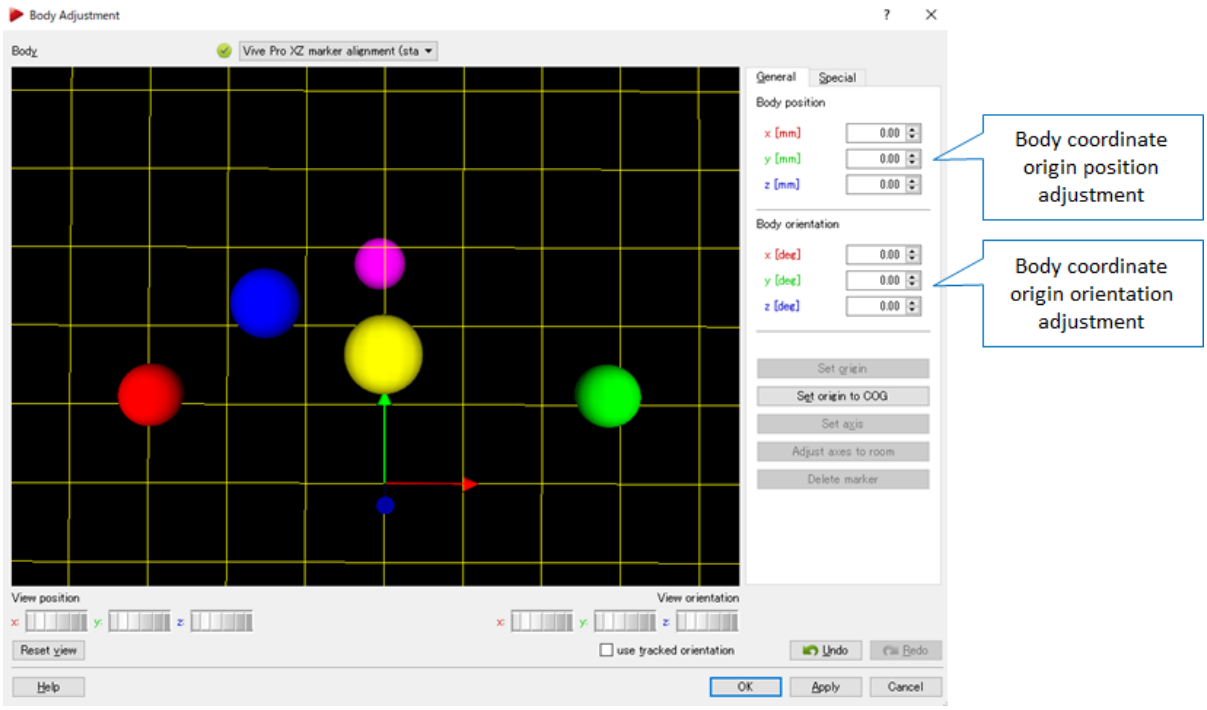

Figure 13 – In the second step, use the Body position and Body orientation fields to correct the orientation of the rigid body's coordinate system. The figure shows a correctly aligned body origin, front view along Z-axis. Note that the relative position of the coordinate origin should be ~3cm below the origin of the marker holder. It is assumed to lie in the center between the user's eyes, on the nose bridge.

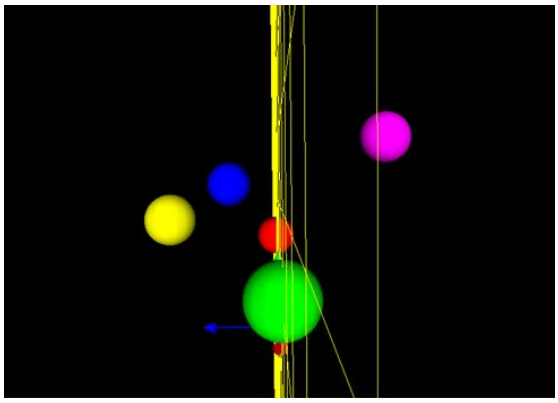

Figure 14 - Correctly aligned body origin, side view along X-axis

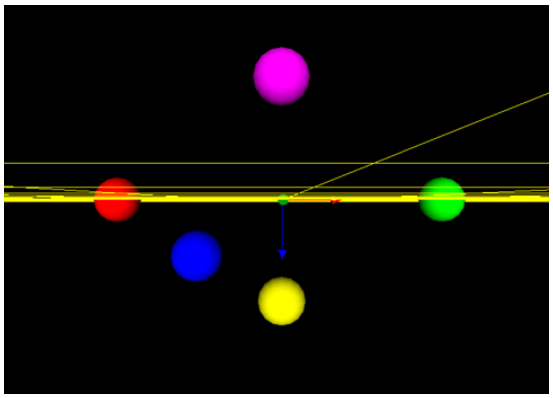

Figure 15 - Correctly aligned body origin, top view along Y-axis

2. After doing the alignment adjustment using the rigid body editor of your optical tracking system, activate the automatic pivot point calibration by selecting your optical tracking system in the `absolute source` field with rigid body id `tracking id` and the headset IMU in the *imu source* field. Clicking `Find Rotation` will start the calibration process.

   To activate the calibration features of LPVR, click on `Show Additional Features` on the LPVR configuration page as displayed in Figure 16.

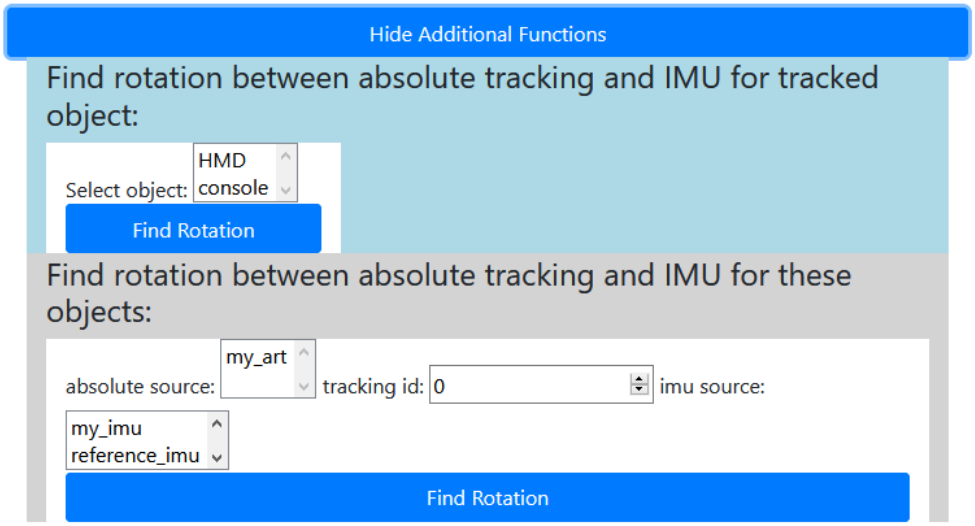

Figure 16 - LPVR calibration dialog

3. During the calibration process, slowly rotate the headset within view of the optical tracking system. Pause the rotation from time-to-time to allow for some internal processing.

4. After the calibration process is finished the result will be output below the calibration feature dialog. The output qutaernion needs to be manually insterted into `absoluteFromImuFrameQuat`.

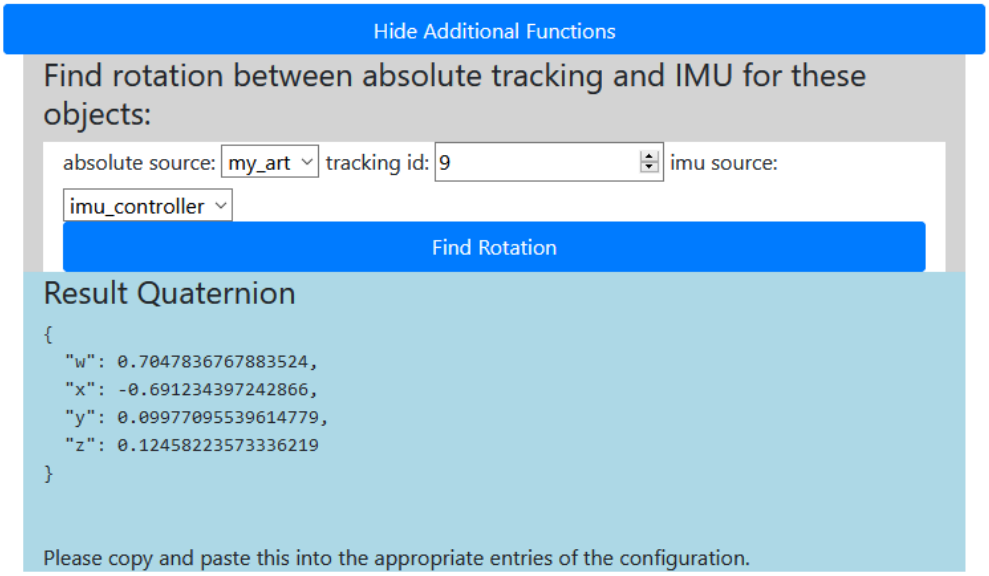

Figure 17 - Output of calibration showing orientation quaternion

5. `imuToEyeQuat` needs to be adjusted to one of the following quaternions depending on the alignment of the IMU:

IMU Attachment	**Quaternion (w, x, y, z)**
LPVR VIVE Pro holder (sensor USB plug pointing up)	`{ 0, -1, 1, 0 }`
LPVR VIVE holder (sensor USB plug pointing down)	`{ 0, 1, 1, 0 }`
VIVE Pro internal IMU	`{ 0, 0, 0, 1 }`

6. The system should work now for tracking the HMD within the optical tracking volume. In case you notice a slight misalignment of your head orientation and the orientation of the floor, or the floor is not horizontal, please make some small adjustments to the alignment of your HMD rigid body in the body alignment editor of the optical tracking system. In our experience, sometimes a few degrees adjustment of the Z-axis helps to make the orientation result feel perfectly natural.

   Please note that in case you notice a very strong misalignment of your movement and the rotation of the image, probably either `imuToEyeQuat` is not correctly adjusted (should always be one of the options shown above), the optical rigid body axis alignment is not correct or you need to redo the `absoluteFromImuFrameQuat` calibration.

   If alignment problems persist, please take the following steps to isolate the source of the problem:

   A. Track headset with IMU-only (no optical):

      - Stop ART tracking in DTrack
      - Set `"ignoreGravity": false` to turn on the vertical gravity reference
      - Set `"combinerType": "Default"` to turn off relative orientation tracking with the reference IMU

      With these settings the headset will be tracked by the HMD IMU only. When wearing the headset, if the IMU alignment settings are correct, orientation tracking should feel natural. Please note that without optical system there is no position tracking. If orientation

tracking is not working well, the alignment between IMU and the HMD optics (display, lenses) is not good. Go back to step 5 and adjust *imuToEyeQuat*.

B. Track headset with optical system-only (no IMU):

- Make sure ART tracking is running
- If it doesn't exist yet, in *imuSources* create

```
{ "id": "no_imu", "type": "None" }
```

- Set *"imuSource": "my_imu"* in *trackedObjects*

Within the range of the optical system, the HMD should now be orientation and position tracked. The tracking will feel a lot less smooth when only using the optical system. However, this mode is a good way to assess the quality of the rigid body adjustment. If orientation tracking completely off, it is better to readjust the rigid body.

USING THE VIVE HAND CONTROLLER

To activate a VIVE hand controller to work with LPVR, add the following code blocks to settings.json. Note that in the code snippets below "…" stands for previously existing code in the respective block. Make sure to enter the correct controller serial number and the ART rigid body ID (zero-indexed) for the optical target attached to the controller.

Make sure the rigid body of the controller is axis aligned as shown in Figure 18.

NOTE: In the current version of LPVR only one controller is supported. The code works with LPVR-CAD and LPVR-DUO.

```
"imuSources": [
  …
  {
    "id": "controller_imu",
    "type": "OpenVR",
    "settings": {
      "serialNumber": "LHR-FCXDDF"
    }
  }
]
```

```
"emitters": [
  …
  {
    "name": "Controller",
    "settings": {
      "type": "Controller"
    },
    "type": "OpenVR"
  }
]
```

```
"trackedObjects": [
  …
  {
    "absoluteSource": {
      "name": "my_art",
      "trackingId": 4
    },
    "combinerType": "Default",
    "emitterName": "Controller",
    "imuSource": "no_imu"
  }
]
```

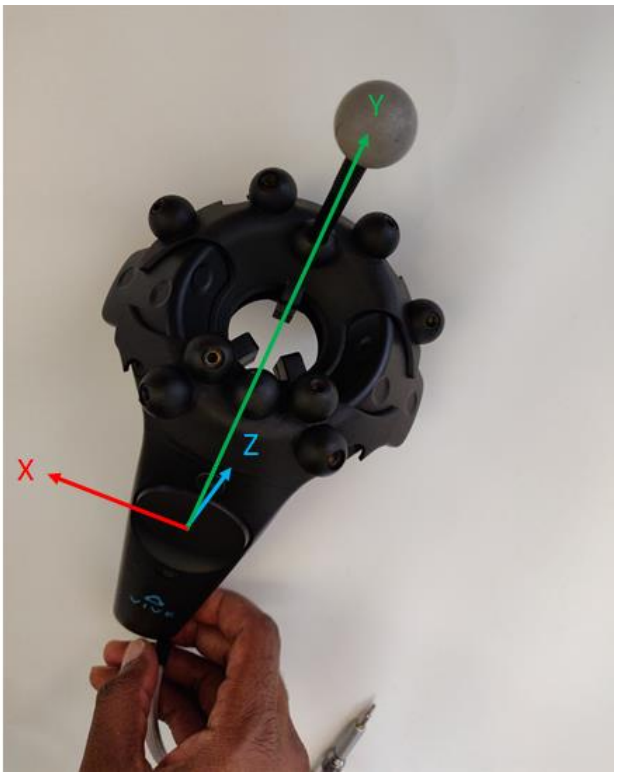

Figure 18 - Alignment of the controller coordinate system (defined in OpenVR): X-axis pointing left, Y pointing upwards along the handle and Z pointing perpendicular into the touchpad. Note that in this image just one marker post is attached, but at least 3 markers are required for proper optical tracking.

ADJUSTMENT OF VEHICLE-FIXED IMU (LPVR-DUO ONLY)

For a LPVR-DUO installation, perform the following steps to adjust and calibrate the orientation and coordinate system of the reference IMU:

1. In a first step the orientation of the internal coordinate system of the vehicle fixed IMU and the optical tracking system needs to be defined. For this purpose, attach an optical marker to the enclosure of the IMU and create a new rigid body in the optical tracking software.

2. Activate the automatic pivot point calibration by selecting your optical tracking system in the *absolute source* field with rigid body id *tracking id* and the vehicle fixed IMU in the *imu source* field. Clicking *Find Rotation* will start the calibration process.

3. During the calibration process, slowly rotate the IMU with attached optical marker within the view of the optical tracking system. Pause the rotation from time-to-time to allow for some internal processing.

4. After the calibration process is finished the result will be output below the calibration feature dialog. The output quaternion needs to be manually insterted into *referenceToOpticalQuat*.

5. Fix the reference IMU to the vehicle with the optical marker still attached. Check the SteamVR console output for the orientation quaternion of the reference IMU optical marker. This value needs to be applied to the field `referenceOrientationQuat` in the configuration script.

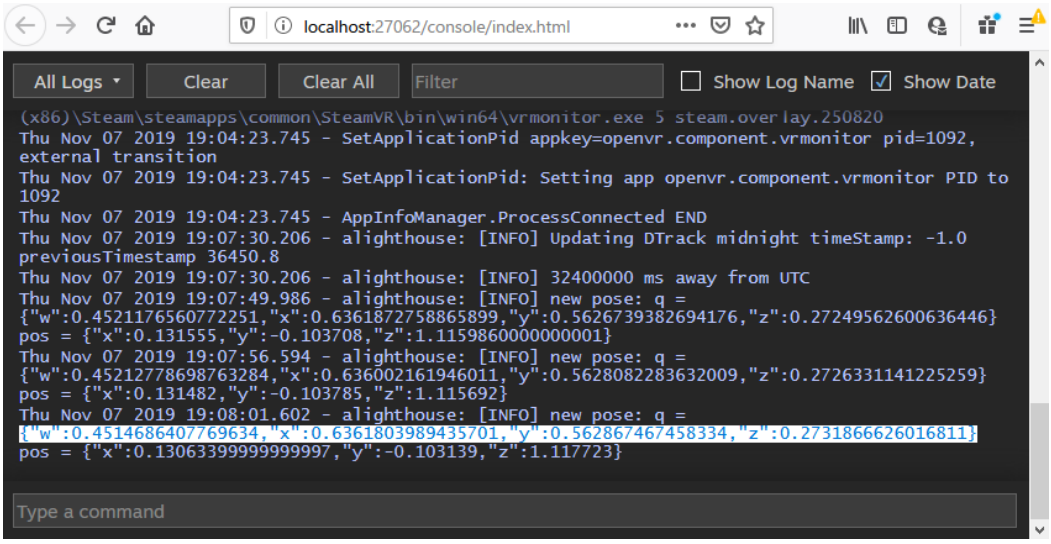

Figure 19 - Output for reference orientation quaternion in SteamVR log output

6. After the above procedure the optical marker can be detached from the reference IMU. NOTE: The reference IMU should now not be moved anymore.

VERIFICATION (LPVR-DUO ONLY)

Note: This is not required for normal operation, but good to know, if you would like to get more accustomed to our system.

In the lab, we can't move the reference IMU along with the optical tracking system, but we can verify that everything moves the correct way. Assume the reference IMU is oriented +y up (in the coordinates on the case). Setting both `referenceToOpticalQuat` and `referenceOrientationQuat` to `w, x, y, z = 1, 0, 0, 0`, the rotations of the reference IMU should move the headset inversely, considering that +z corresponds to the backwards direction, and +y to the up direction.

E.g. leaning forward (negative rotation around x) should make the headset lean backward (i.e. look up). The direction of a rotation of the reference IMU from the nominal position should apply fixed in space. E.g. if a rotation of the reference IMU leads to the field of view moving upward, then turning the headset 90 degrees to the left, the same rotation of the reference IMU should look as though the headset is tilting left.

Another item to verify is that `absoluteFromImuQuat` and `eyeFromImuQuat` cancel each other (i.e. the orientation is the same no matter what values they are set to as long as their values are the same) and the IMU data is correctly translated to the headset frame (i.e. it moves the right way w/o optical tracking.

APPENDIX

COMMAND LINE-BASED PIVOT POINT CALIBRATION (OBSOLETE)

NOTE: The pivot point calibration using the command line-based calibration tool is now obsolete. The pivot point calibration can now be launched from inside the LPVR configuration screen.

CALIBRATION PROCEDURE

To use the PivotPoint tool, please follow the steps below:

1. Start the tool from the command line with *PivotPoint pivot.json*, where pivot.json is a configuration file that defines the sources for the pivot point calculation.
2. Rotate the headset very slowly within range of the optical tracking system. It is best to pause the rotating motion briefly from time to time.
3. The pivot point tool will count the number of samples in the command line window. If the calibration motion is performed well, the tool will finish after counting to 50 and output a result quaternion *absoluteFromImuQuat*.
4. Insert the new *absoluteFromImuQuat* into the LPVR driver's *settings.json*.

PIVOTPOINT TOOL CONFIGURATION FILE

The table below describes the entries to the PivotPoint configuration file that is to be passed to the PivotPoint tool:

Optical Tracking System	Example Code Block	Explanation	

Optitrack	<pre>{
	  "absoluteSource": {
	  "name": "my_optitrack",
	  "type": "OptiTrack",
	  "settings": {
	  "connectionType": "Multicast",
	  "localAddress": "127.0.0.1",
	  "remoteAddress": "127.0.0.1",
	  "serverCommandPort": 1510,
	  "serverDataPort": 1511
	  }
	  },
	  "trackingId": 0,
	  "imuName": "00:04:3e:9b:a3:5a"
	}</pre>
		name	Defines the name of the source. Any name is good.

		type	Must be "Optitrack"
		connectionType	Must be "Multicast"
		localAddress	Local address of the Optitrack client.
		remoteAddress	Address of the Optitrack server
		serverCommandPort	Must be 1510
		serverDataPort	Must be 1511
		trackingId	ID of object tracked by optical system.
		imuName	MAC address of LPMS IMU sensor (enter only if you are using an LPMS-B2 with your setup!)

PIVOTPOINT PARAMETERS FOR LPVR DRIVER

There are two transformations that affect the relationship between the IMU, the rigid body of the optical tracking system and the user's eyes:

1. *absoluteFromImuQuat* and *absoluteFromImuVect* gives the orientation and displacement of the tracked body in the IMU body frame
2. *eyeFromImuQuat* and *eyeFromImuVect* take us from the IMU to our eyes (more precisely the bridge of the nose) inside the IMU body frame

Technical note: The pose prediction by SteamVR is done in the body frame of the IMU, which is why we need to keep these transformations separate instead of transforming everything to the eye right away.

This was designed for the Lighthouse which updates rather infrequently (25 Hz along each axis). With a professional optical tracking system like ART on the other hand we have much quicker updates, rendering the evaluation of the navigation equations much less important, resulting in smaller terms that can be eliminated accordingly.

Therefore, it's usually not perceivable, if the prediction is done on the nose bridge right away. I.e. it is a valid simplification to keep `eyeFromImuVect = { "x":0, "y":0, "z":0 }`, and, if your optical tracking rigid body is aligned with the nose bridge, also to keep `absoluteFromImuVect = {"x":0, "y":0, "z":0 }`.

Otherwise you would enter the vector (in meters) taking you from the rigid body to the eyes.

LPVR HOLDER (VIVE VERSION) ASSEMBLY

HOLDER AND IMU ASSEMBLY

1. Make sure you set the VIVE's eye distance to a medium value to ensure fit of the strap holders. You can re-adjust the eye distance after assembly.
2. Remove cable cover to be able to access the USB port.
3. Place the headset in front of you, facing your way, cables up.
4. Slide marker holder pieces over VIVE as indicated in Figure 20.
5. Follow the further steps as shown in Figure 21, Figure 22.
6. Insert the USB cable into the IMU, thread it through the cable holder and into the open USB port in the VIVE, reinsert the cable cover.

OPTICAL MARKER ASSEMBLY

1. Attach optical tracking markers as shown in Figure 23.
2. One marker should be inserted in one of the four pods in the top corners using a 50mm rod. This ensures visibility from behind. All other markers can use shorter screws.
3. One marker each should be attached on the far ends on each side to define the horizontal axis of the solid body.
4. The remaining two markers are distributed over the attachment pods on the front to ensure different combinations for different headsets.
5. While screwing the screws into the pods, make sure that you stay in the correct direction to ensure maximal stability post-assembly. Figure 24 shows the complete assembled holder.

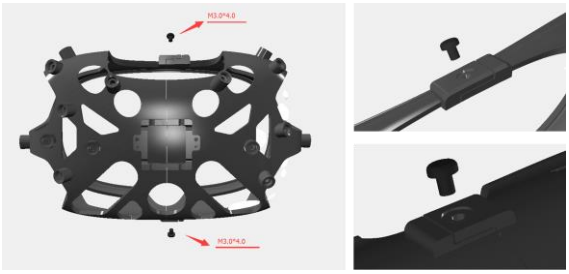

Figure 20 - Connect the two halves using the supplied (short) screws.

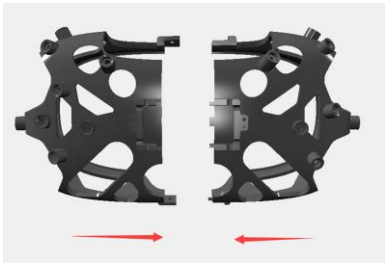

Figure 21 - The VIVE holder consists of two pieces that slide over the HMD from the side.

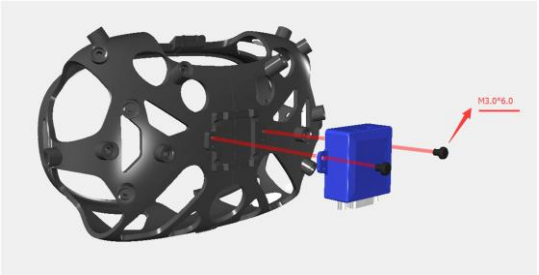

Figure 22 - Attach the IMU with the USB port facing upwards using the supplied (long) screws.

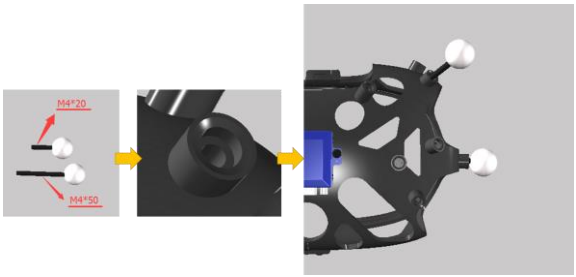

Figure 23 – Steps to attach the optical tracking marker.

Figure 24 - The fully assembled VIVE LPVR holder.

LPVR HOLDER (VIVE PRO VERSION) ASSEMBLY

HOLDER

1. Remove the small hex-key screws at the top and bottom of the VIVE Pro
2. Slide the holder over the headset with the protruding pieces on the inside of the holder locking into the now empty screw holes of the VIVE Pro.
3. Use the small hex key screws delivered with the holder to fix the holder to the headset.
4. Attach the LPMS-CU2 IMU to the holder using the supplied M3 screws.
5. Plug the micro USB cable into the IMU, fix it to the cable guide on the holder and plug the other side into the USB-C port of the VIVE.

OPTICAL TRACKING MARKER ATTACHMENT

- One marker should be inserted in one of the four pods in the top corners using a 50mm rod. This ensures visibility from behind. All other markers can use shorter screws.
- One marker each should be attached on the far ends on each side to define the horizontal axis of the solid body.
- The remaining two markers are distributed over the attachment pods on the front to ensure different combinations for different headsets.
- While screwing the screws into the pods, make sure that you stay in the correct direction to ensure maximal stability post-assembly. Fig. 6 shows the complete assembled holder.

Figure 25 - Fully assembled VIVE Pro LPVR holder

LPVR HAND CONTROLLER ASSEMBLY

Figure 26 - The upper and lower part of the maker holder needs to be assembled around the VIVE / VIVE Pro hand controller.

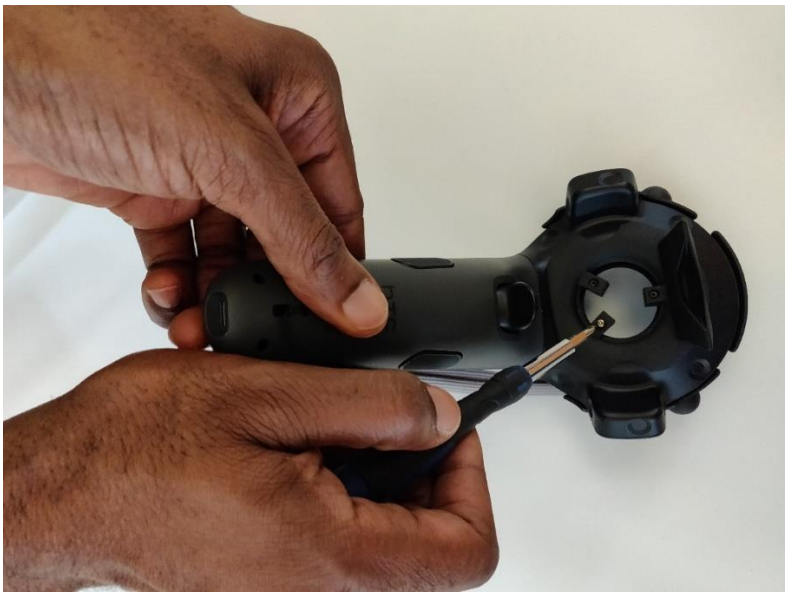

Figure 27 - Align the holder components with the hand controller as shown in the image and connect them using the screws delivered with the set.

Figure 28 - Tracking markers are attached to the holder using a standard M3 distance rod.

OPEN SOURCE LICENSES

- These binaries use the Eigen library version 3.3.4 which is licensed under the Mozilla Public License 2.0, and which is available for download here:

  `http://bitbucket.org/eigen/eigen/get/3.3.4.zip`

- We also make use of nlohmann::json library which carries the following copyright notice:

  The MIT License (MIT)

  Copyright c 2013-2017 Niels Lohmann

  Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the Software), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

  The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

  THE SOFTWARE IS PROVIDED AS IS, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- We also make use of the cpprestsdk library which carries the following copyright notice:

  C++ REST SDK

ABOUT US

This manual was created by LP-RESEARCH Inc. If you have any feedback or questions, please contact us.

Email:	info@lp-research.com
Tel:	+81-3-6804-1610
Address:	#303 Y-Flat, 1-11-15, Nishiazabu, Minato-ku, Tokyo, 106-0031 Japan