Optical Sensor Calibration

SteamVR™ Tracking

Introduction

When developing a SteamVR™ tracked object, it is critical that the system knows exactly where the optical sensors are located on the object. Though the JSON file may be configured with the locations of the sensors as dictated the mechanical model, submillimeter anomalies introduced during the assembly process can have a negative impact on the tracking performance of the object. To counteract these anomalies, a brief calibration procedure should be performed on each object after assembly. The procedure produces a new JSON configuration that places the sensors closer to their real-world locations.

Overview

The calibration routine for tracked objects requires the same hardware used during normal operation. An ideal JSON configuration file is provided to the calibration tool, and the object is moved and rotated through the VR space. The calibration tool records sensor hits and attempts to fit a new set of sensor locations and orientations that best matches the data recorded. These best-fit locations and orientations are provided in a new JSON file that can be programmed into the object for improved performance.

Setup

To perform calibration, you will need the following:

1. **vrtrackingcalib.exe** - This command-line utility is located in the directory:
   ```
   C:\Program Files (x86)\Steam\steamapps\common\SteamVR Tracking HDK\tools\bin\win32
   ```
2. **Tracked object** - The fully assembled object, capable of rudimentary tracking. If your object is unable to boot into the SteamVR™ environment, you will not be able to calibrate it.
3. **JSON file** - This JSON file should be an ideal JSON file generated directly from the 3D model of the object. Note that the "device_serial_number" field must match the serial number of the object you are trying to calibrate. This is how the calibration tool identifies your object. Do not use a JSON file that was the result of a previous calibration routine. Using calibration results in further calibrations could cause small errors to stack up, similar to making a photocopy of a photocopy.
4. **Base Station** - A single base station. If you have a standard VR system with two base stations, power one of them off for the calibration procedure.
5. **Space** - The calibration routine works best when you are able to move the trackable object over a large space. The standard 2 x 1.5 meter VR space is ideal.

Running Calibration

There are many functions of the calibration tool that are outside the scope of this document. The following procedure describes standard body calibration.

vrtrackingcalib.exe is a command-line utility that must be called from the Windows command prompt and provided several runtime flags as shown here:

```
> vrtrackingcalib.exe /bodycal <your JSON file.json> 800 200
```

The runtime flags are as follows:

- /**bodycal** - Specifies that you want to perform a sensor body calibration. The following three arguments are the object JSON configuration file as well as two numbers that represent the total number of required sensor hits and the number of hits required per sensor. Larger numbers collect more data and produce better results, but they also take longer to finish calibration.

With the appropriate flags input into the command line, press enter to run the calibration routine. Vrtrackingcalib outputs a lot of information describing its setup procedure. Eventually, vrtrackingcalib is ready for calibration:

```
Ready to run capture position number 0.
Press <enter> key to begin:
```

If you do not see this prompt, verify that the object is turned on and in view of the base station. Also verify that the object boots into the SteamVR™ environment.

Press enter to begin the calibration process. The goal at this point is to introduce the object to the widest variety of translational and rotational positions possible within the base station's active area. A line of dots grows to represent the number of positions and orientations that have been recorded. If the object is left motionless, the dots and the calibration routine halt. You may need to use a large amount of physical volume to achieve the best results.

Eventually, vrtrackingcalib.exe outputs a line like the following:

```
Sensor IDs TBD:  22 (0-hits) 24 (0-hits) 20 (0-hits) 18 (0-hits) 14 (0-hits) 16
(0-hits) 12 (0-hits) 10 (0-hits)  8  6  4  2  0 (0-hits) 30 (0-hits) 29 31  1  3  5  7
9 15 11 13 17 19 21 23 25 27 28 26 (0-hits)
```

This lists the sensors that have not yet received their minimum number of hits as specified in the runtime argument. It also specifically calls out sensors that have received zero hits. The goal during calibration is to make sure that every sensor on your object has received the minimum number of hits necessary. If a sensor is not getting enough hits, rotate the object so that it can see the base station.

Eventually, the number of sensors listed drops:

```
Sensor IDs TBD:  16 10  8  6  4  0  1  3  5 11
```

When it drops to zero and the required total number of hits as specified in the runtime argument has been recorded, the calibration routine stops and displays the results.

Calibration Results

Most of the resulting data is irrelevant for basic body calibration, but there are a few important points.

```
Termination: CONVERGENCE
```

This specifies that the calibration routine was successful, and vrtrackingcalib.exe was able to fit a sensor map to the data collected. If you do not see this result, verify that your sensors are collecting data and that your JSON configuration file has them mapped as well as possible.

```
Corrected scale of 0.9901
```

This specifies how much larger or smaller the trackable object is when compared to its JSON configuration file. Closer to 1 is better.

```
Corrected normal shift of -0.0013
```

This number represents the shift in location of the actual photodiode to the location in the CAD model along a line normal to the sensor plane. It's usually a very small number which just accounts for the thickness of the lens covering the photodiode.

```
Corrected rotation of 0.0583 deg
```

This specifies how much the object is rotated compared to its JSON file. Lower is better. 0 is best.

```
Corrected translation of 0.1459 mm
```

This specifies how much the object has been shifted compared to its JSON file. Lower is better. 0 is best.

If any of your results are far away from their ideal values, examine the object and JSON file to see if there are any major errors.

In addition to this debug information, the tool generates two JSON files: a calib_observations_xxxxx.json file which contains mostly the same information as the command-line output and an auto_<serial number>.json file which contains the same information as the object's original serial number but with the modeNormals and modelPoints adjusted. As shown below, the new values are slightly different than the original values and have a higher level of precision. Upload the new JSON file onto the object to improve tracking performance.

JSON Pre-calibration:

```
"modelNormals" : [
[0.851,0.397,-0.342],
[0.397131, 0.85165074, -0.34202014],
[0.707107, 0, -0.70710678],...
```

JSON Post-calibration:

```
"modelNormals" : [
[ 0.85152214765548706, 0.39724358916282654, -0.34220984578132629 ],
[ 0.3971310555934906, 0.85165083408355713, -0.342020183801651 ],
[ 0.70710694789886475, 0, -0.70710670948028564 ],...
```