

## OPTITRACK AT THE DVIC: CHANGELOG

To get up to speed with Optitrack, you have 3 tutorials depending on your needs. Beginners who wish to use the setup unsupervised should look into **this Introduction to Optitrack**. Precision-dependant scenarios will make use of **the Optitrack changelog**. The changelog is where I compiled my list of issues with Optitrack. Finally, the long-term users might be interested in simplifying maintenance and capture processes with **the official changelog for arena maintenance**.

Last updated September 2020. Please contact me at [thomas.carstens@edu.devinci.fr](mailto:thomas.carstens@edu.devinci.fr) for any questions or system support.

### Part of a series

1. An introduction to Optitrack
2. DVIC Changelog
3. Arena maintenance

## Contents

.....	1
Introduction.....	2
System demo with robotic control.....	2
Sequential output.....	2
Taking a low-latency estimate and ramping up the precision .....	2
Section 1: Precision capture theory .....	3
Changelog .....	5
Testing different illumination levels.....	5
Interference affecting system precision.....	6
Camera: not receiving frame data.....	8
Data streaming into a Linux client.....	9

## Introduction

**Motion capture** is the discipline of tracking movements. It has usecases from film (to add CGI effects) to robotics (to find the position of robots). In my case, the Optitrack framework provided the means of detecting my drone's position where **onboard methods would be too imprecise**. After all, Optitrack boasts of their **submillimetre accuracy**, stating that an accuracy of less than a third of a millimetre should be "more than manageable".

This tutorial is meant to **upskill users** from Optitrack's setup and first use, to more precise uses of the system. The framework builds around **optical motion capture**, which has its own set of benefits and challenges, and Optitrack offers solutions to improving **capture precision** in various usecases. Finally, since the tracked data is multicasted, setting up a **multicast client** is possible both in Windows and Ubuntu, and a few **troubleshooting skills** can help connect your particular application (mine is using **ROS for drone control**). Using this framework on the **drone arena** might require **maintenance of the arena**, which is covered in a separate tutorial.

Let's understand what we are trying to do in the end.

## System demo with robotic control

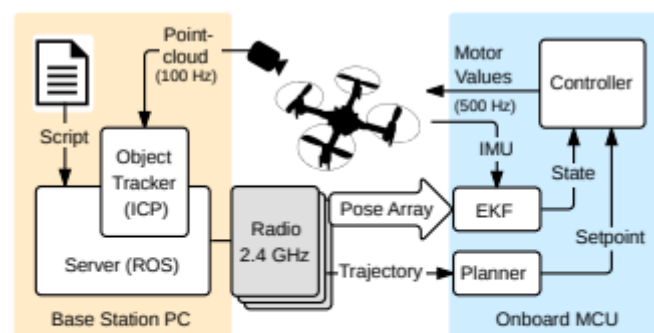


Fig. 2. Diagram of major system components. A point cloud of markers detected by a motion-capture system is used to track the quadcopters. All estimated poses are broadcasted using three radios. Planning, state estimation, and control run onboard each vehicle at 500 Hz.

## Sequential output

- Taking down cameras and refocusing on the ground
- Adding them with care for pully strings, and pulling net backwards
- Placing the drones (turned on) on the carpet
- Turning on ROS
- Sliding down the net carefully
- Executing Python code for trajectory.

In the interest of precision, here are some steps to have a more precise estimate.

## Taking a low-latency estimate and ramping up the precision

- Waiting for initial heat up
- Precision Mode
- Lens refocus
- Lower residual threshold
- Higher circularity vs number of rays per marker
- No need to touch on marker size

Taking two cameras and calibrating them according to their position

- Maximum ray length
- Position vs Ray Angle.

What does this mean?

## Section 1: Precision capture theory

If the cameras are too close to one another, they can reflect IR light on one another, or on objects that are too close. Inversely, if they are too far, the IR light does not reach the object. The camera might also be out of focus at certain distances, which makes it difficult to differentiate markers from extraneous reflections. All these problems contribute to identifying if a reflection is in fact a marker, or distinguishing markers from each other.

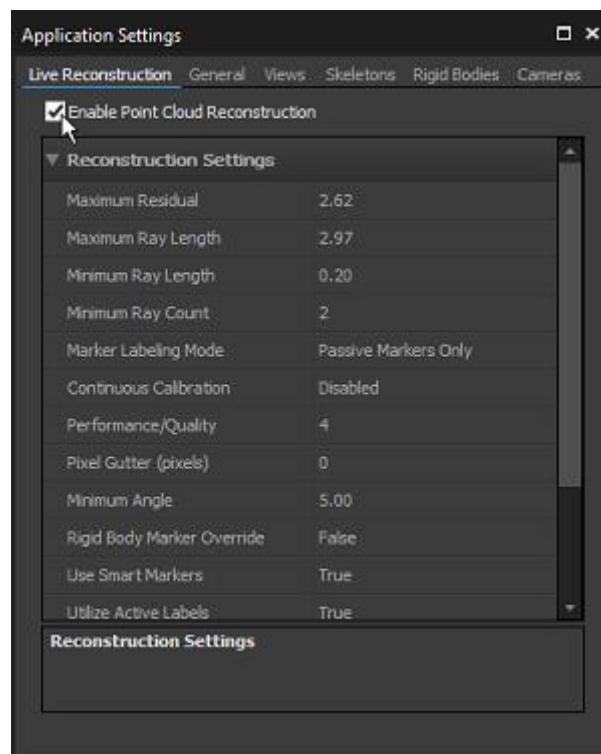
Various solutions exist. First, we wish to have a clear image on the camera. The lens can be refocused. The LED intensity can be varied, as well as exposure. After these hardware solutions, the camera itself processes images with 2D Filters: discarding reflections if they do not fall in a correct size range (size threshold) and if they do not have sufficient circularity (minimum circularity). The THR setting or threshold for brightness, also determines the pixels based on their illumination intensity. What is interesting is that these filters are applied frame by frame, and therefore cannot benefit of any reconstruction over time.

Reconstruction is the process of deriving 3D points from the 2D coordinates that are obtained from camera images. This process first obtains centroid locations from one camera's individual frames, and then use each camera's centroid to triangulate a 3D position in the capture area. Various means of precision are done, and they stem from a 3D vector extending from each camera origin to its detected centroid, otherwise called a marker ray.



The **average offset distance between the converging rays** gives us a precision rating, called the residual value or mean residual. A maximum residual can be set as a “reconstruction” software setting. The length of the 3D vector also has importance, as “middle-range” lengths are preferred

over “too-close” or “too far” (more likely to blur). Adding these two thresholds give two of the software filters post-reconstruction.



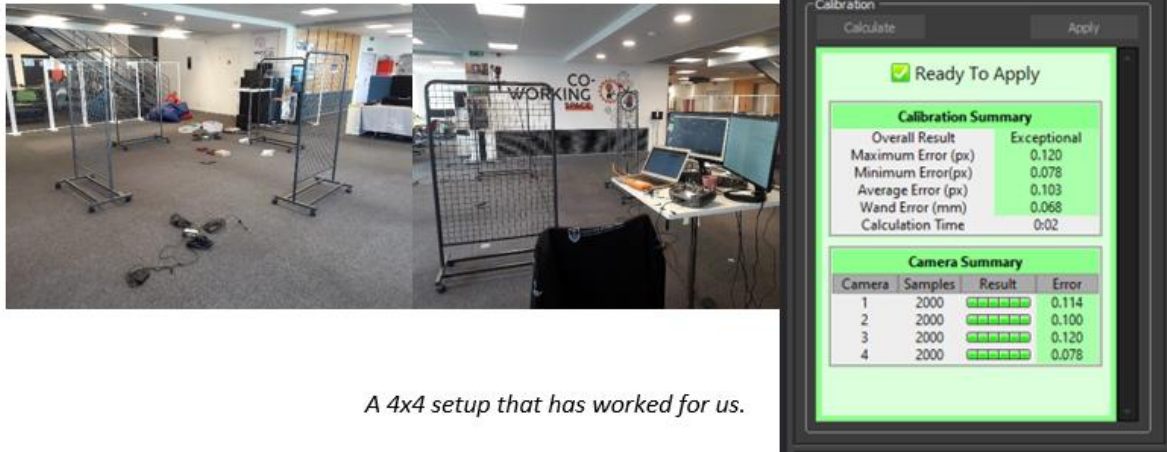
Having said this, there have been compromises that are made in the interest of reducing the amount of computation and thus risks to reduce system latency and system frame rate. The camera, by default, is in Object Mode, which means that 2D object metrics (the centroid location, the size of the markers and the roundness of the markers) are determined on-camera as the software filters mentioned earlier. This has the lowest processing latency as opposed to Precision Mode, where the marker reflections and centroid regions are sent to the host for more precise but more computationally expensive processing. This is acceptable in lower camera count environments.

The use of a camera in grayscale helps to understand what is going on, thus two Reference Modes are possible. MJPEG grayscale compresses images on-camera while raw grayscale remains uncompressed (to the risk of high streaming bandwidth).

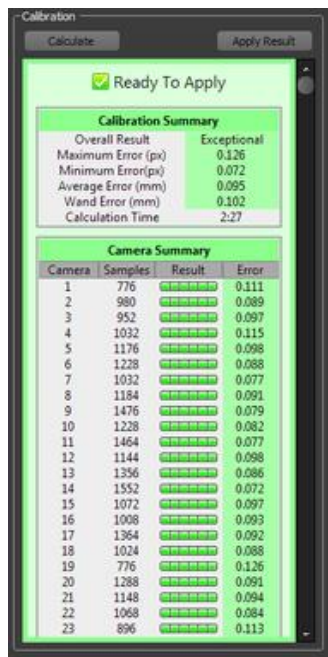
## Changelog

### Testing different illumination levels

It is worth noting that a 4x4 setup with full sunlight illumination has proven to work with results seen below (see calibration section below to understand the criteria). It is at your own discretion to check if markers are detected, **via Motive's grayscale option**. At the DVIC, we chose this second configuration as a more permanent option.



*A 4x4 setup that has worked for us.*



Calibration Result Report

Category	Description
Overall Reprojection	Displays the overall resulting 2D and 3D reprojection error values from the calibration.
Worst Camera	Displays the highest 2D and 3D reprojection error value from the calibration.
Triangulation	Triangulation section displays calibration results on <a href="#">residual offset</a> values. Smaller residual error means more precise reconstructions. <ul style="list-style-type: none"> <li>Recommended: Recommended maximum residual offset for point cloud reconstruction.</li> <li>Residual Mean Error: Average residual error from the calibration.</li> </ul>
Overall Wand Error	Displays a mean error value of the detected wand length throughout the wandering process.
Ray Length	Displays a suggested maximum tracking distance, or a ray length, for each camera.

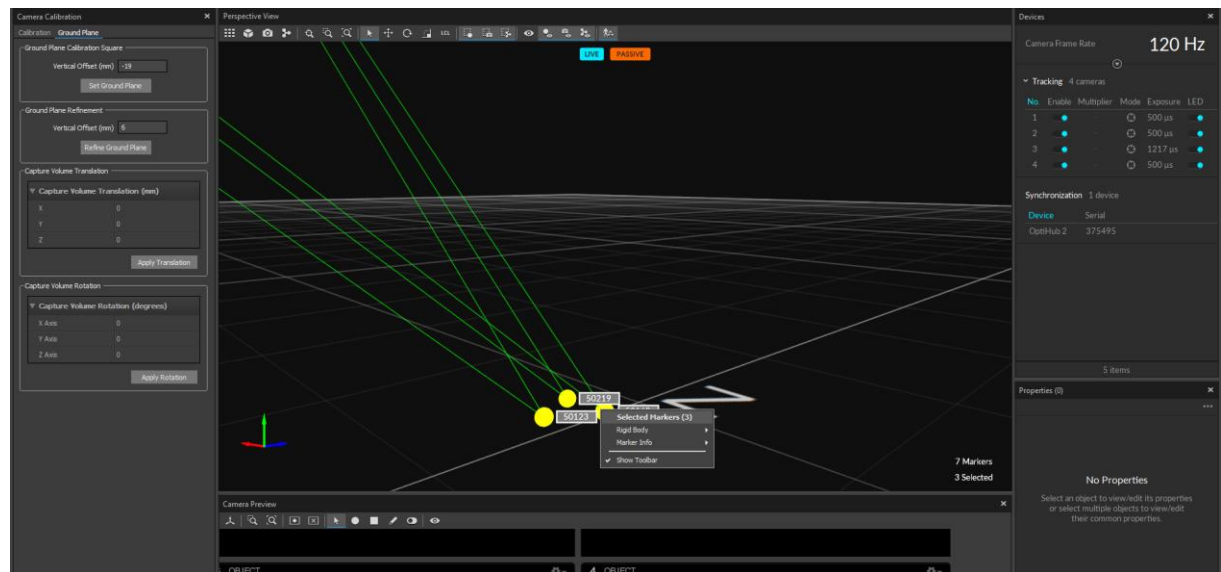
## Interference affecting system precision

Issues encountered: an excellent calibration loses in precision after some time. Not sure what point in the loop is affected by it.

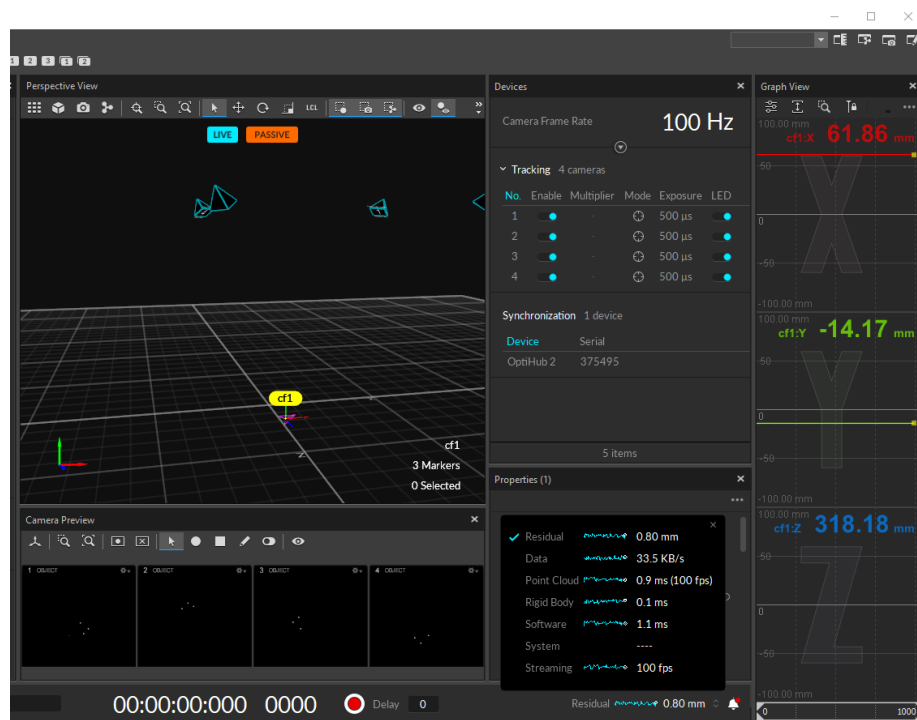
Approach: testing the cameras for position error.

1. Placing an object to be tracked as a rigidbody.

This can be defined directly from the Perspective View as such.



2. Position and orientation information can be graphed as such.



Let's attempt the whole process: bringing the nets down.

And now, examining the position curves!

### 1. Oscillatory behaviour

In my experience, this has been associated with cameras that have been moved, but are now stable. If this behaviour appears on the **x** axis, ask yourself: which cameras is that associated to? Also examine the camera pane for **wobbling** points.

## 2. Replicate the procedure, this time looking closely for wobbling.

Solution: I detected a pully in contact with Camera 3.


**Similar** diagnostic: marker wobble **and** marker too small **and** loses Solved component

- Exposure imprecision
- Increase THR setting

## Camera: not receiving frame data

Issues encountered: as Motive starts up, the cameras appear off. They seem to be bugging as they appear to be enabled.

1. Disconnect and reconnect the cameras.
2. Then turn on Motive. Motive has a log that outputs:

	CAM Camera #: Not Receiving Frame Data.	Indicates that the Camera (#) is not receiving frame data. This could be just because the cameras are still waiting to be initialized. If this status persists, it is like due to a hardware problem.
---	---	---

Other such errors can be found here: [https://v22.wiki.optitrack.com/index.php?title=Status\\_Log](https://v22.wiki.optitrack.com/index.php?title=Status_Log)

Solution: first see if cameras can be detected, using the Camera SDK.

1. Close Motive and download the camera SDK. Run the cameras in grayscale mode.
2. Reboot Motive.

This seems to have worked. It seems like an initialization issue, and it has occurred when the camera power is connected **after** Motive has enabled the cameras. Just another subtle reminder to keep the cameras off until needed.

## General introduction to Flex 13 cameras

Introductory video: <https://www.youtube.com/watch?v=4RMz1KVDsbA>



## TECHNICAL SPECIFICATIONS

### CAMERA BODY

- Width: 2.12 inches (53.8 mm)
- Height: 3.19 inches (81 mm)
- Depth: 1.67 inches (42.4 mm)
- Weight: 6.60 ounces (187 g)
- Mounting: 1/4"-20 tripod thread
- Status Indicators:
  - 2 digit numeric LEDs
  - 1 bicolor status LED

### IMAGE SENSOR

- Pixel Size : 4.8  $\mu\text{m}$   $\times$  4.8  $\mu\text{m}$
- Imager Size : 6.144 mm  $\times$  4.9152 mm
- Imager Resolution : 1280  $\times$  1024 (1.3 MP)
- Frame Rate: 30-120 FPS (adjustable)
- Accuracy: Sub-millimeter
- Latency: 8.3 ms
- Shutter Type: Global
- Shutter Speed:
  - Default: 500  $\mu\text{s}$
  - Minimum: 20  $\mu\text{s}$
  - Maximum: 7.5 ms (at 120 FPS)

### IMAGE PROCESSING TYPES

- Object (Centroids)
- Precision (Grayscale)
- Segment (Thresholded)
- MJPEG Grayscale
- Raw Grayscale

### LENS & FILTER

- Default Lens: 5.5mm F#1.8
  - Horizontal FOV: 56°
  - Vertical FOV: 46°
- Optional Lens: 8 mm F#1.8
  - Horizontal FOV: 42°
  - Vertical FOV: 34°
- M12 Lens Mount
- Adjustable focus w/ spring assist
- 800 nm IR pass filter
- Optional: 800nm IR pass filter w/ Filter Switcher

### LED RING

- 28 LEDs
- 850 nm IR
- Adjustable brightness
- Strobe or Continuous Illumination
- Removable

### INPUT/OUTPUT & POWER

- Data: USB 2.0
- Camera Sync: USB 2.0 (via OptiSync)
- Power: USB 2.0 @ 1A

### SYSTEM REQUIREMENTS

- Windows XP/Vista/7
- 1GHz processor
- 1GB of RAM
- 50MB of available disk space
- USB 2.0 Hi-speed port
- OptiHub 2 (required for IR LED power)

### IN THE BOX

- 1 Flex 13 camera (part number: FL-13)
- 1 Quick Start guide

## Data streaming into a Linux client

### Issues encountered: no incoming data?

Solution: first see if data does in fact stream.

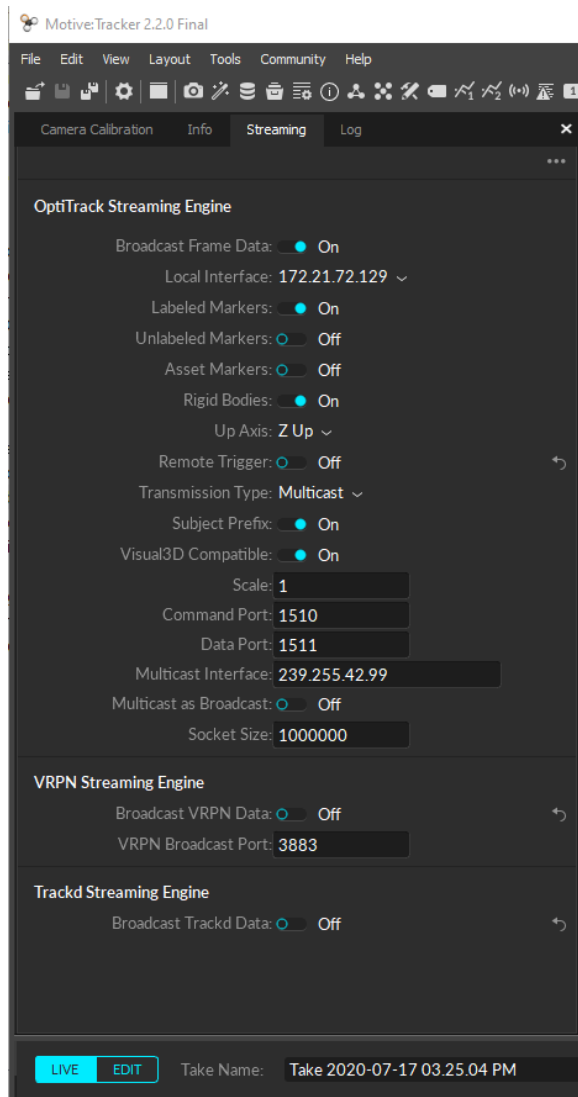
1. For some reason yet unknown, it takes either a ping to the Windows host and another back in order to establish a secure connection.
2. Data streaming into a hexdump.

Now the data does stream. Ubuntu needs to be configured for two things: enabling multicasts, and opening up ports on the firewall for streamed data.

1. Multicasting options need to be configured on Ubuntu: <https://doc.ubuntu-fr.org/multicast>
2. Firewall ports must be opened.

## General introduction to data streaming

Official Optitrack documentation: [https://v22.wiki.optitrack.com/index.php?title=Data\\_Streaming](https://v22.wiki.optitrack.com/index.php?title=Data_Streaming)



From the documentation:

*Motive offers multiple options to stream tracking data onto external applications in real-time. Streaming plugins are available for Autodesk Motion Builder, The MotionMonitor, Visual3D, Unreal Engine 4, 3ds Max, Maya (VCS), VRPN, and trackd, and they can be downloaded from the OptiTrack website. For other streaming options, the NatNet SDK enables users to build custom clients to receive capture data. All of the listed streaming options do not require separate licenses to use. Common motion capture applications rely on real-time tracking, and the OptiTrack system is designed to deliver data at an extremely low latency even when streaming to third-party pipelines. This page covers configuring Motive to broadcast frame data over a selected server network. Detailed instructions on specific [streaming protocols](#) are included in the PDF documentation that ships with the respective plugins or SDK's.*