Spatial tracking for VR

Part 1: IMUs, Valve Lighthouse v1.0 / v2.0 and Oculus cameras

Karol Gasinski, 9 July 2020

Agenda

Spatial tracking for VR (Part 1 - HW)

- Basics:
  - Tracking overview
  - Tracking types by DOF (Degrees of Freedom)
  - Pose reconstruction
- HW:
  - IMUs
  - Lighthouse v1.0
  - Lighthouse v2.0
  - Oculus Constellation Trackers
- ~~Algorithms~~ <- deferred to part 2 due to amount of material

Basics

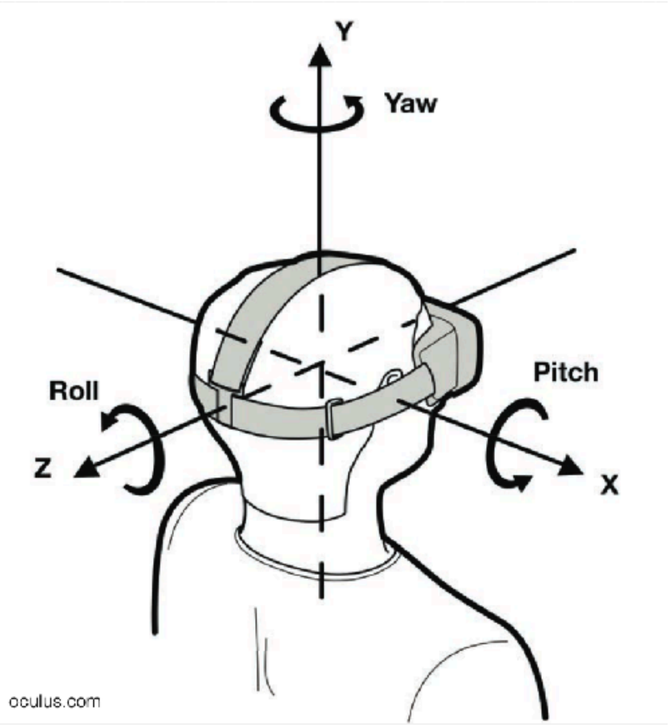
Tracking overview

- Tracking algorithm in a nutshell:
  - Device pose reconstruction in 3D space (at some frequency)
  - Based on above, pose prediction:
    - Pose interpolation for past moment in time
    - Pose extrapolation for future moment in time
- In most cases prediction is async to reconstruction

Basics

Tracking types by DOF (Degrees of Freedom)

- We divide tracking into two types:
  - 3DOF - Orientation only (device orientation, thus rotation in 3 axes)
  - 6DOF - Full tracking composed of:
    - 3DOF orientation
    - 3DOF position (translation in 3 axes)



Why orientation-only tracking makes people sick?

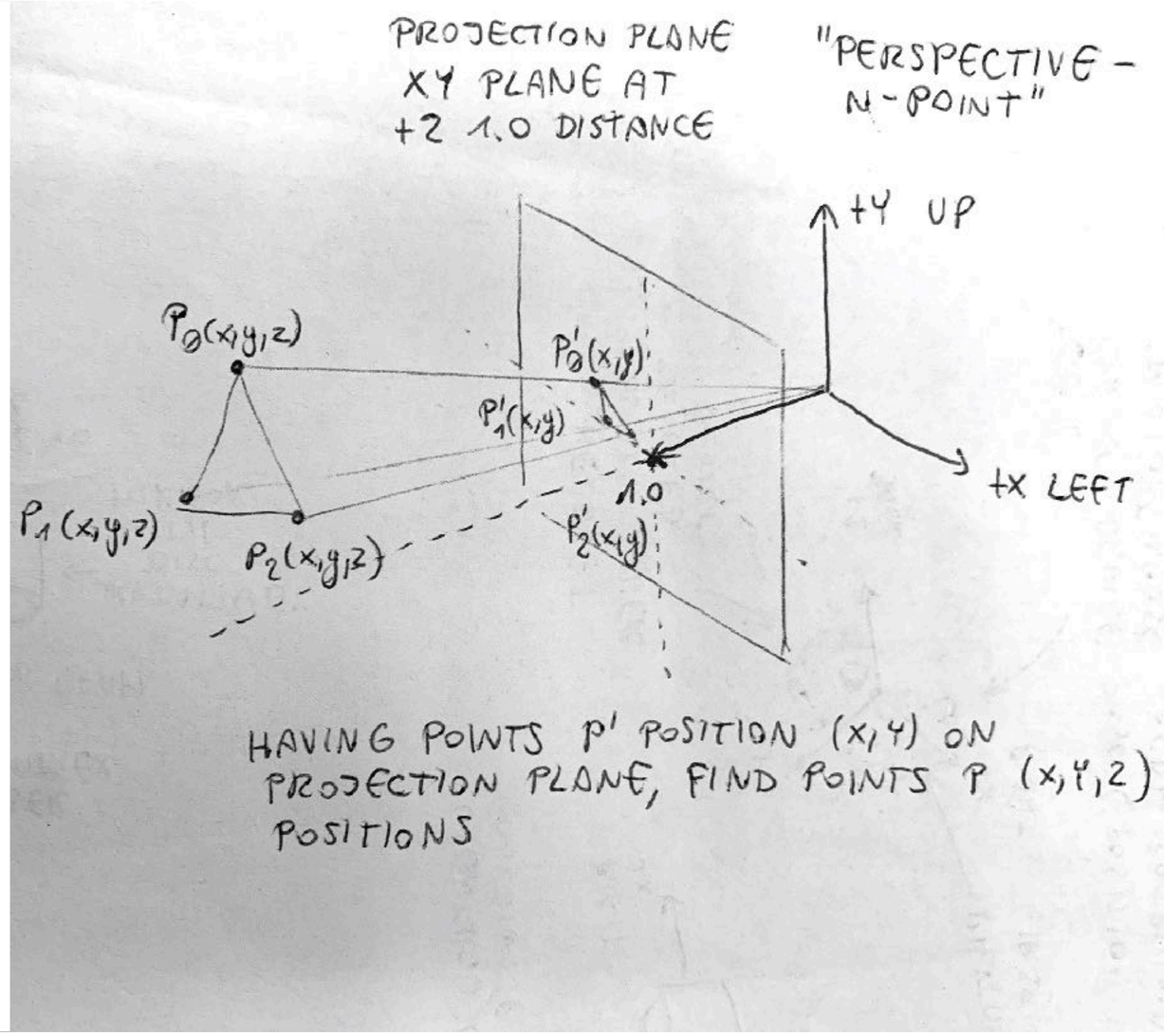
Without position component, there is no difference between turning head and leaning forward. This results in disjoint between observed simulation and experienced space.

Basics

Pose reconstruction

- All pose reconstruction algorithms boil down to solving “perspective-n-point” problem (Oculus Tracking, Valve Lighthouse, VIO/SLAM, etc.).
- We have some observer that sees 2D projection of observed 3D object (projection of it’s “characteristic points”).
- Based on that 2D projection we want to reconstruct object position in 3D space.
- We know object 3D shape.

https://en.wikipedia.org/wiki/Perspective-n-Point



HW

Devices & Sensors

- IMUs
- Lighthouses v1.0
- Lighthouses v2.0
- Oculus Constellation Tracking

IMU

Inertial Measurement Unit

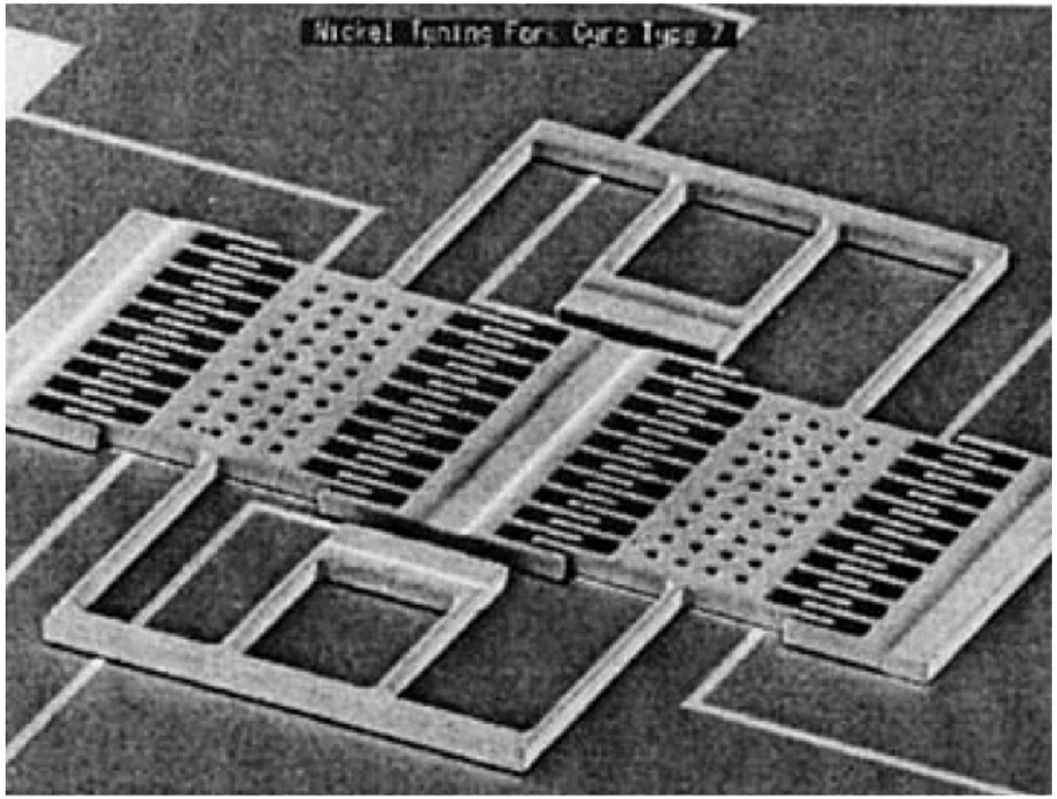
- Two types of most commonly used IMUs:
  - 6DOF - Accelerometer and Gyroscope
  - 9DOF - Accelerometer, Gyroscope and Magnetometer
- Reported values:
  - Gyroscope - angular velocity in deg/s (convert to rad/s)
  - Accelerometer - linear acceleration in Earth G's (convert to m/s^2)
  - Magnetometer - magnetic field strength in Gauss (convert to micro Teslas)

https://stanford.edu/class/ee267/lectures/lecture9.pdf

IMU

Inertial Measurement Unit - Gyroscope

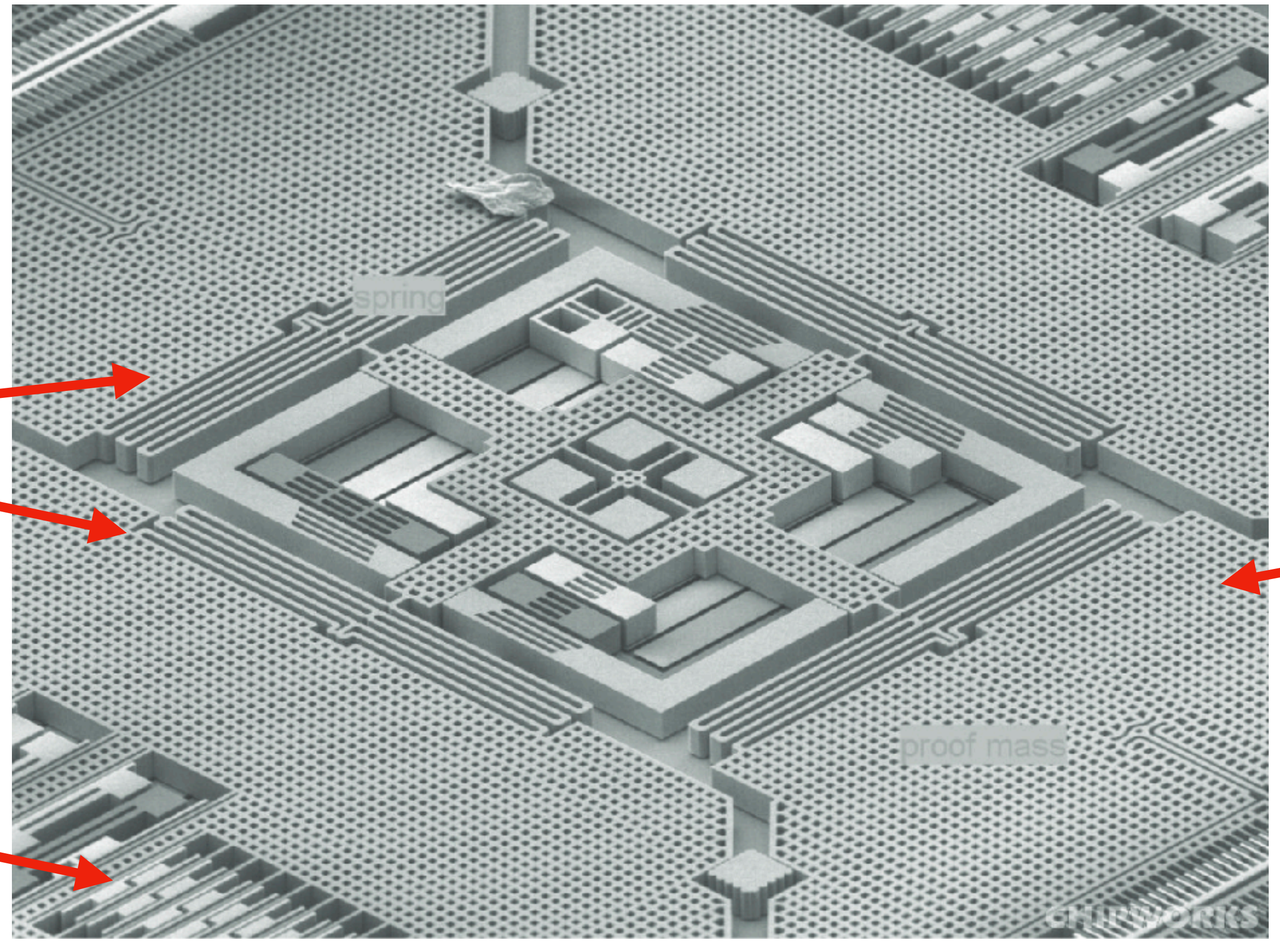
- Gyroscope reading is composed of actual angular velocity, bias and noise.
- Bias depends on device temperature, and can change at run time.
- Over time, bias causes gyroscope reading to drift.



IMU

Inertial Measurement Unit - Accelerometer

MEMS - Micro Electro Mechanical System



Springs

Capacitive plates

Proof mass

IMU

Inertial Measurement Unit - Accelerometer

- Accelerometer reading include Earth gravity vector, as it’s in device local coordinate frame. This means that for most motions, gravity component will dominate reported acceleration vector (leaving user generated acceleration less precision).
- Noisy during short motions, but provides gravity vector as axis of reference.
- Accelerometer combined with Gyroscope allows tilt correction: Roll and Pitch correction. There is no reference vector defining “forward” direction in XZ plane (floor plane) to correct Yaw drift.

IMU

Inertial Measurement Unit - Magnetometer

- Magnetometer reading provide information about strength of Earth magnetic field. That filed is not uniform across the globe, thus it cannot be used to point global direction (like North or South).
- Additionally any electronic devices in surroundings, cables, etc. can impact magnetic field reading. Thus Magnetometers are very noisy, and additionally they operate at much slower frequencies (10-100Hz).
- When used with Accelerometer and Gyroscope readings, they can provide missing reference vector defining “forward” direction in XZ plane (floor plane), and thus allow Yaw drift correction.
- All three combined together, allow device orientation reconstruction (3DOF).

IMU

Models used in headsets

- Oculus DK1 use Invensense MPU-6000 6DOF IMU (now owned by TDK) plus Honeywell HMC5983 (magnetometer), giving full 9DOF combo.
  - See: https://www.ifixit.com/Teardown/Oculus+Rift+DK1+Teardown/13682
  - See: http://msl.cs.illinois.edu/~lavalle/papers/LavYerKatAnt14.pdf
- Oculus DK2, HTC Vive, VivePro - all use Invensense MPU-6500 6DOF IMU
  - See step 11: https://www.ifixit.com/Teardown/HTC+Vive+Teardown/62213
- Oculus Rift switched to Bosch Sensortec BMI055 6DOF IMU
  - See step 9: https://www.ifixit.com/Teardown/Oculus+Rift+CV1+Teardown/60612
- Finally unreleased Oculus Del Mar uses latest TDK ICM-20601 that has increased ranges of gyroscope (4000 deg/s) and accelerometer (32g). Probably for purposes of VR games like Beat Saber.

IMU

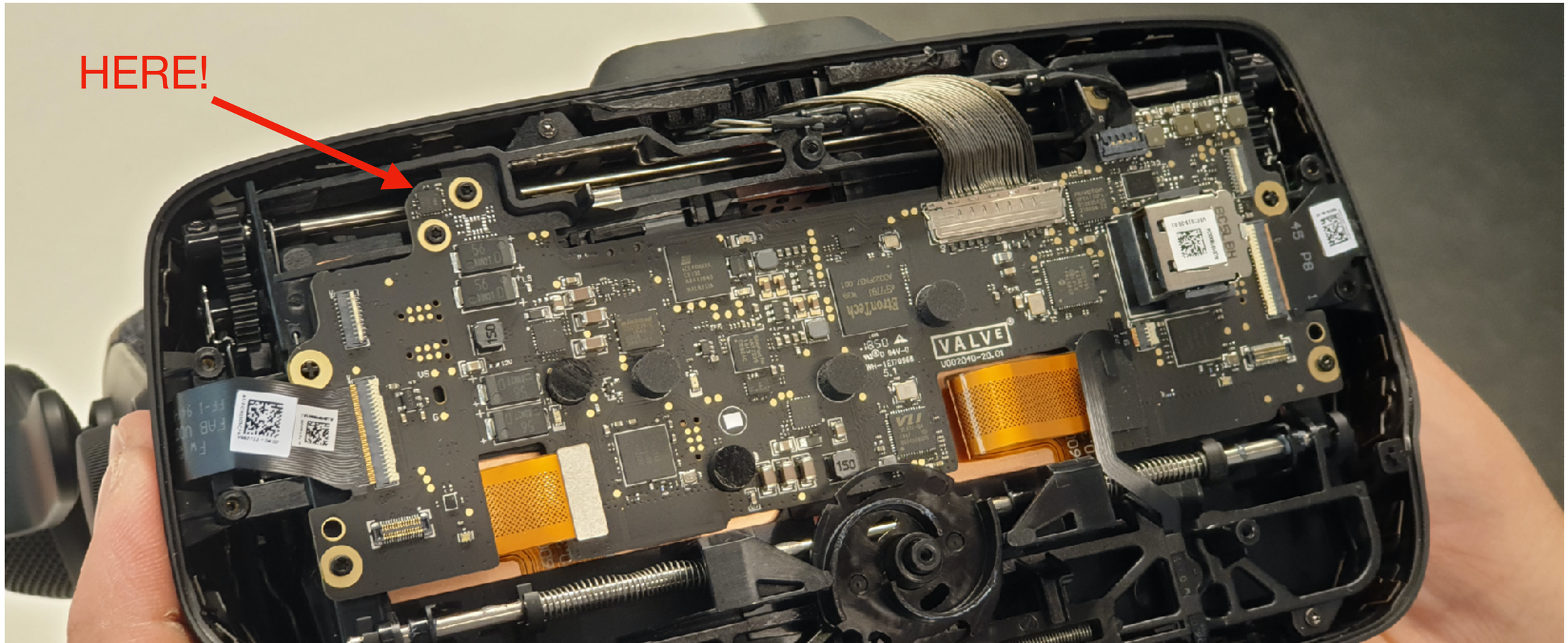
Models used in headsets

- Why only Oculus DK1 have 9DOF IMU?
  - Because it doesn’t have any external tracking. In case of full positional tracking (6DOF), IMU Yaw drift is corrected by external tracking (for e.g. Lighthouse, or SLAM). Thus manufacturers skip 9DOF IMUs in favor of cheaper 6DOF ones.
- By default all IMUs provide Accelerometer and Gyroscope samples at 1000Hz. This minimizes latency from sample acquisition to pose reconstruction and dT of pose prediction.
- HTC and Valve devices using Valve FPGA deliver last three IMU samples from ring buffer (mitigates stalls on packet processing). Oculus headsets deliver samples at 500Hz (in pairs).

IMU

Inertial Measurement Unit

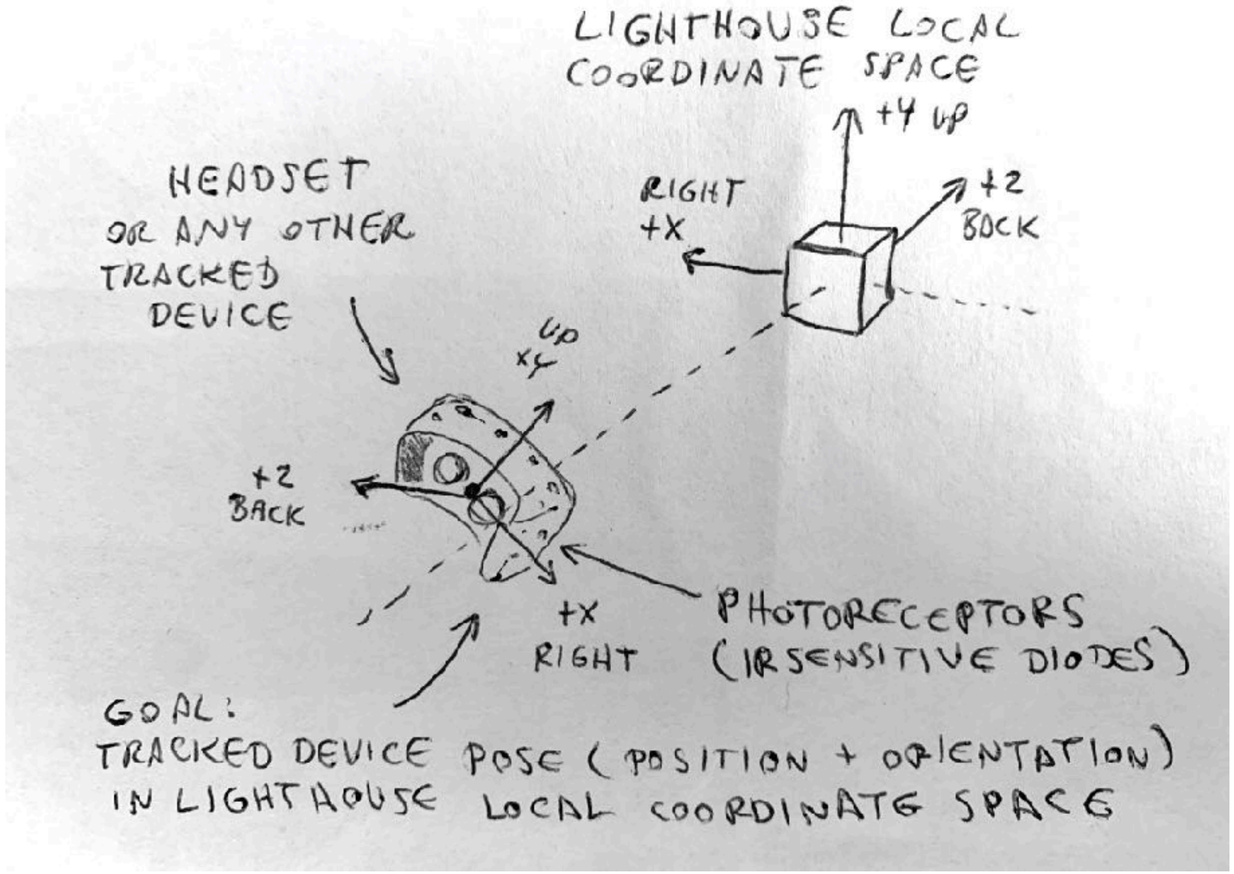
- How it is physically oriented in device? (Valve Index)



Lighthouse tracking

Introduction to problem

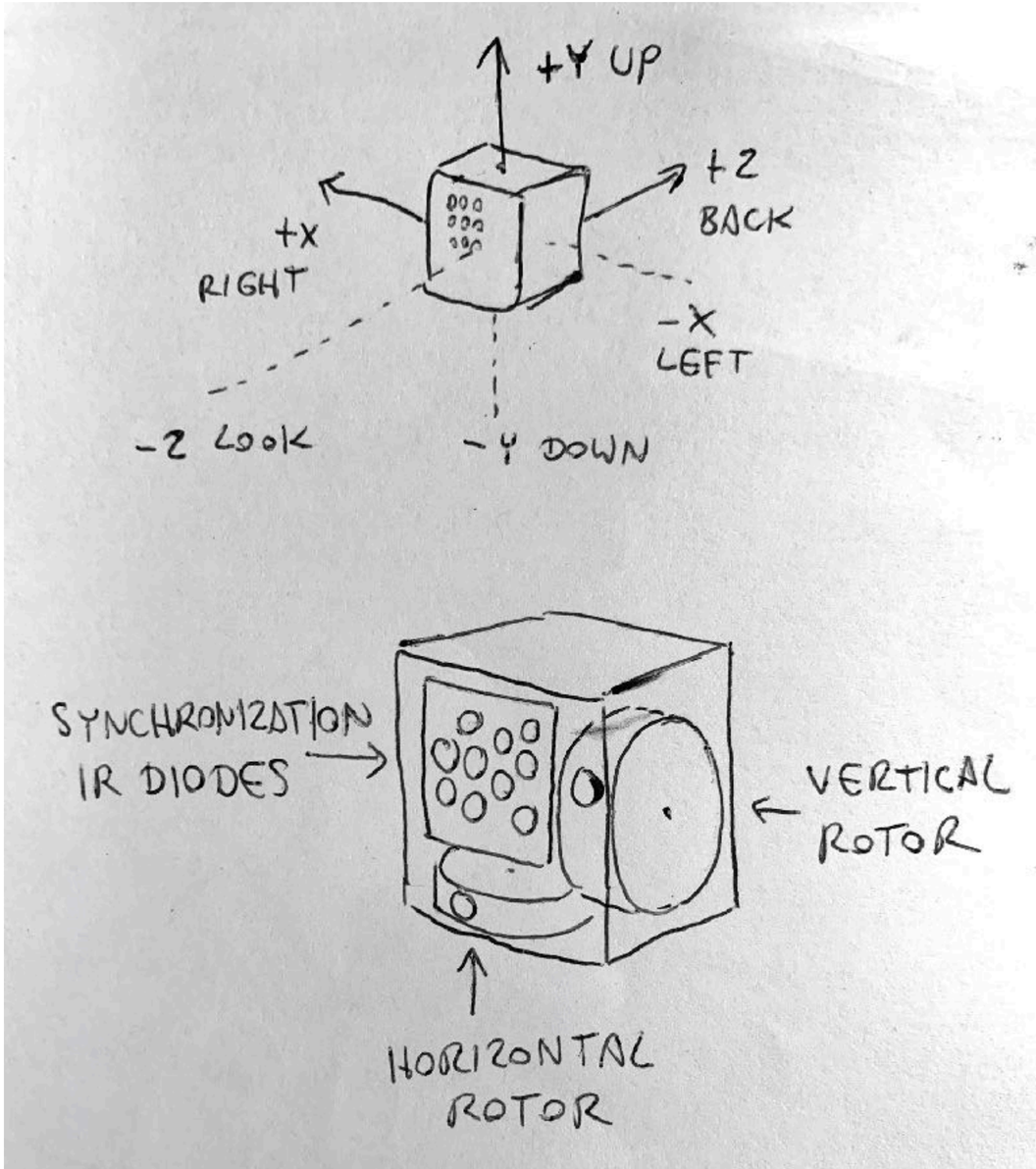
- We want to calculate tracked device position, in reference to stationary coordinate system (frame of reference) defined by Lighthouse.
- Lighthouse becomes observer in Persepctive-N-Point problem.
- Tracked device is “observed” through Lighthouse laser sweeps.
- Device photoreceptor sensors become “characteristic points”.



Lighthouse v1.0

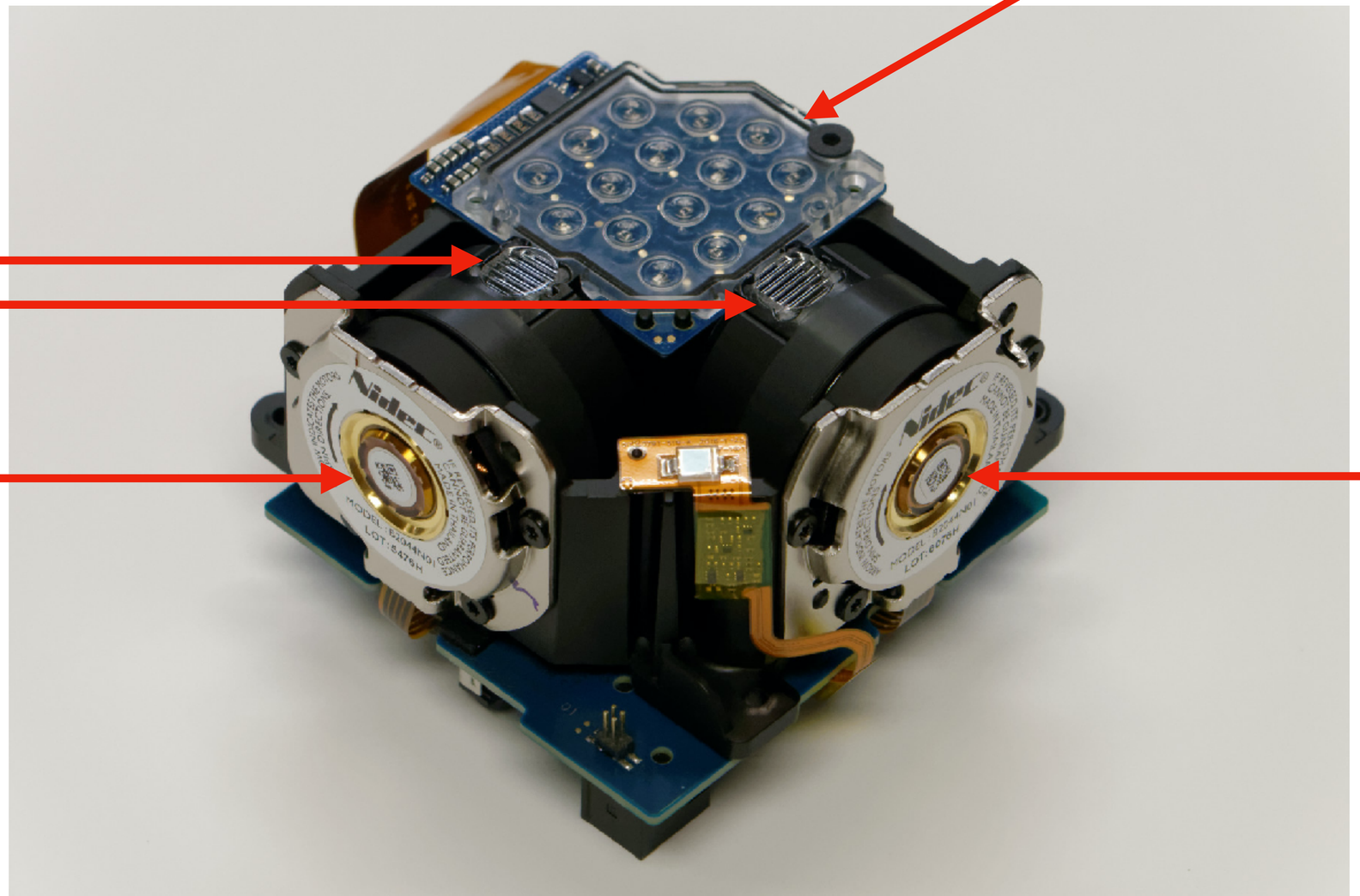
Main components

- Lighthouse is composed of 3 main elements:
  - Horizontal rotor
  - Vertical rotor
  - Synchronization IR diodes



Lighthouse v1.0

Main components

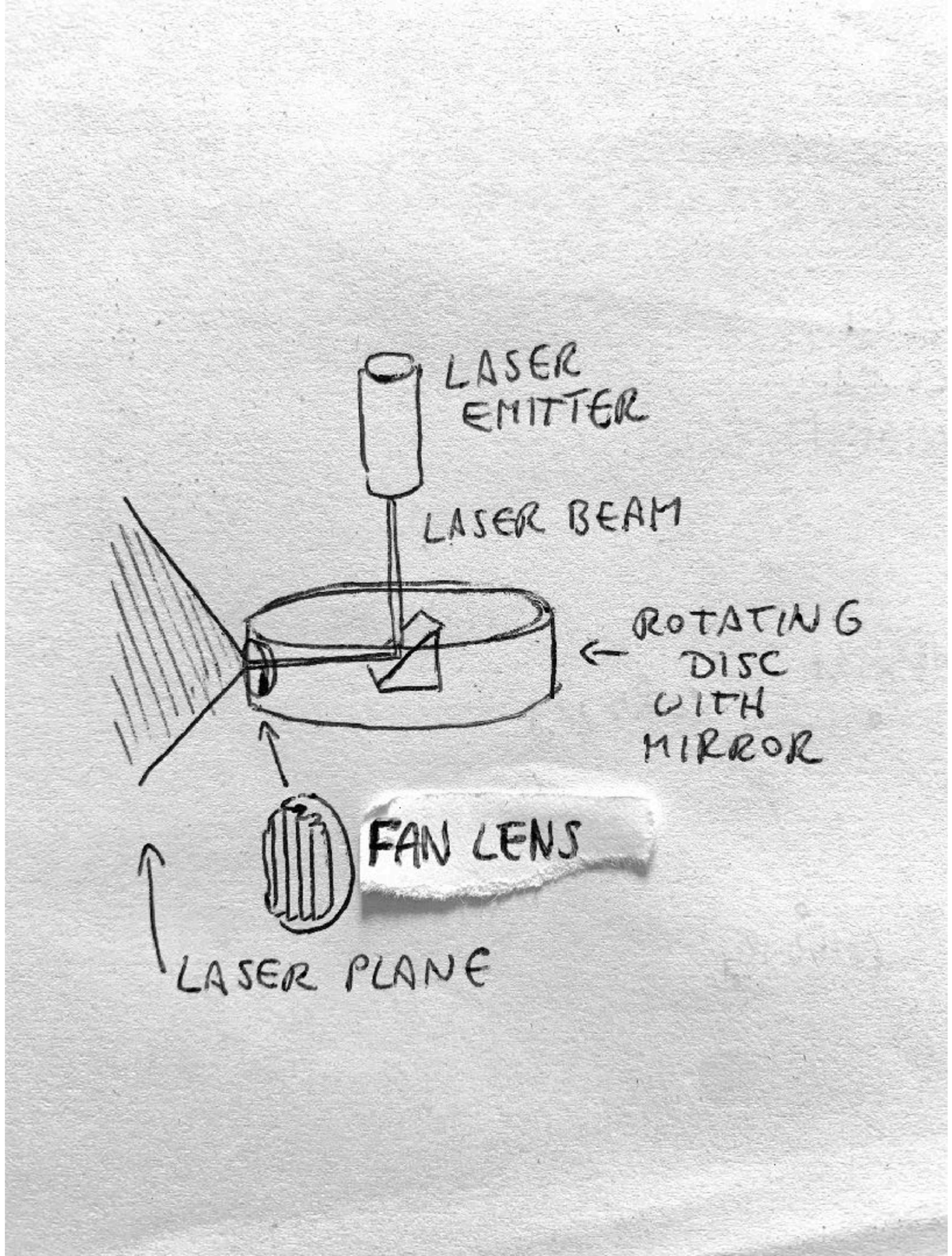


Lighthouse v1.0

Horizontal and Vertical rotors

- Rotors are the same type of drives like in HDD
- They rotate at 3600 RPM / 60 RPS / 60Hz (which is enough to correct orientation drift from IMU)
- Each rotor has fan lens on side of it’s disc, through which laser plane is shaped and emitted (4mm thick)
- System requires rotor position to be known with precision of 94nm
- Rotation speed is dynamically adjusted by controller to mitigate mechanical imperfections (rotor jitter), and ensures that angular position is corrected with precision of ~2500 atoms.

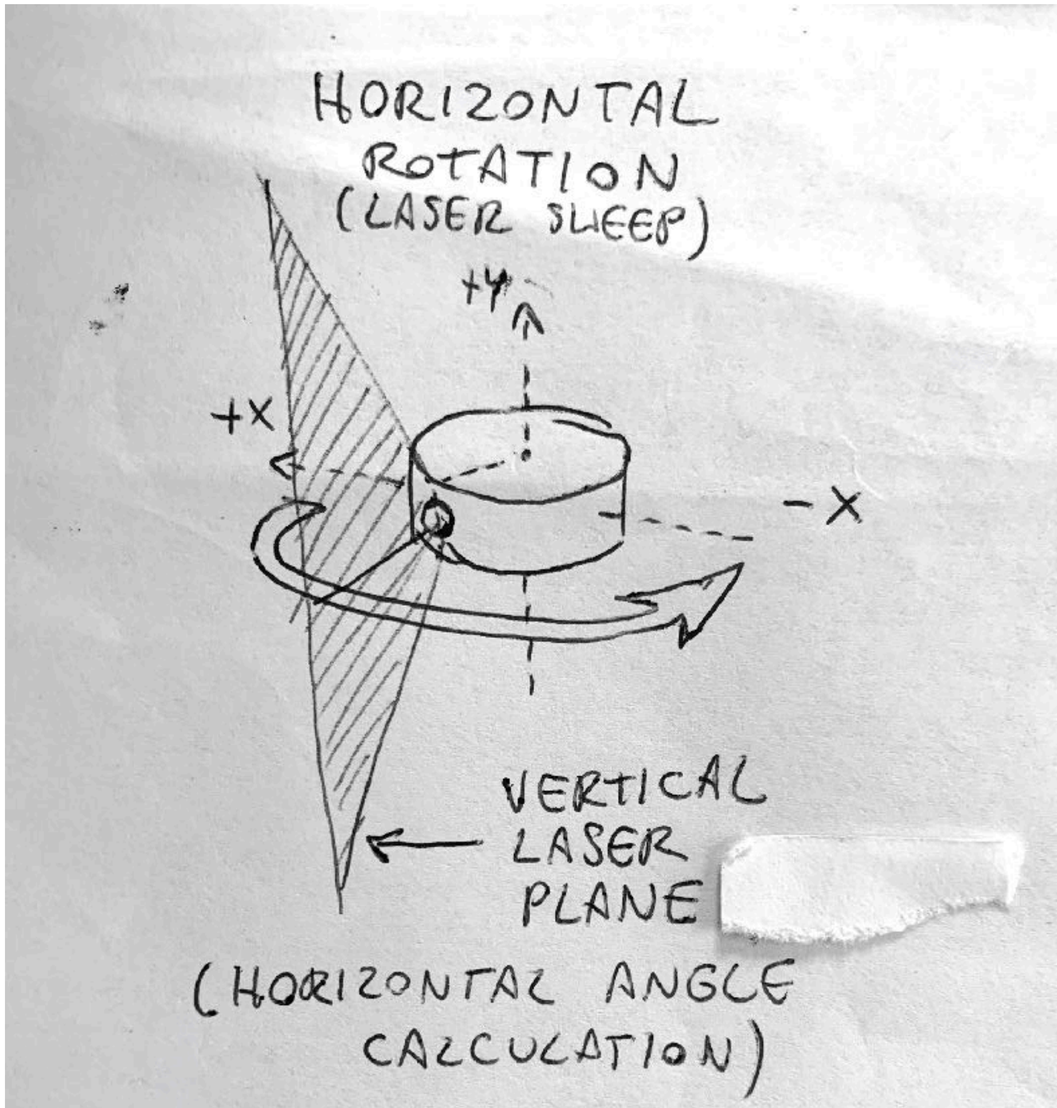
https://www.youtube.com/watch?v=75ZytcYANTA



Lighthouse v1.0

Horizontal Laser Sweep

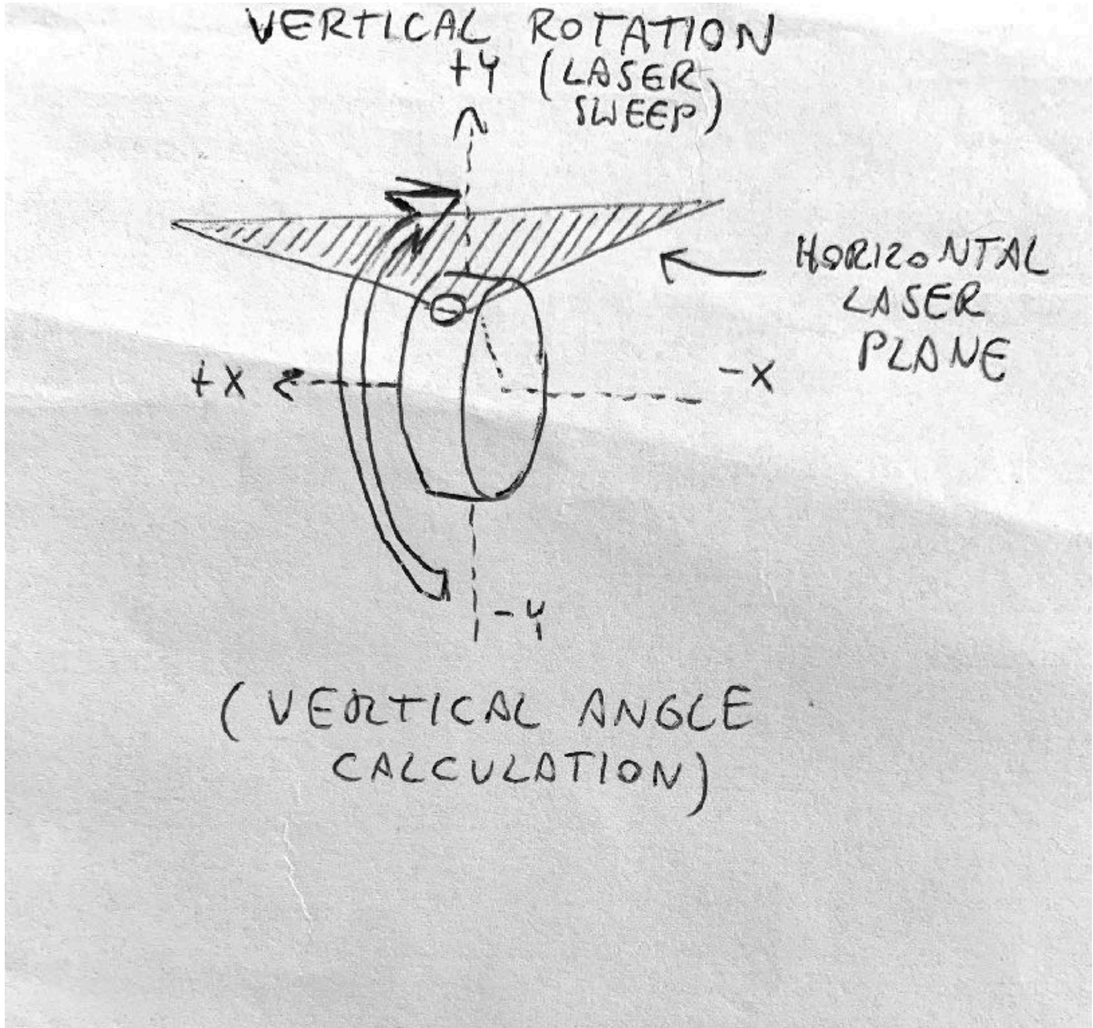
- Horizontal rotation
- Rotates from +X to -X (Right to Left)
- Sweeps space with vertical laser plane
- Allows horizontal angle calculation (sensor angular position from YZ plane, lighthouse look vector).



Lighthouse v1.0

Vertical Laser Sweep

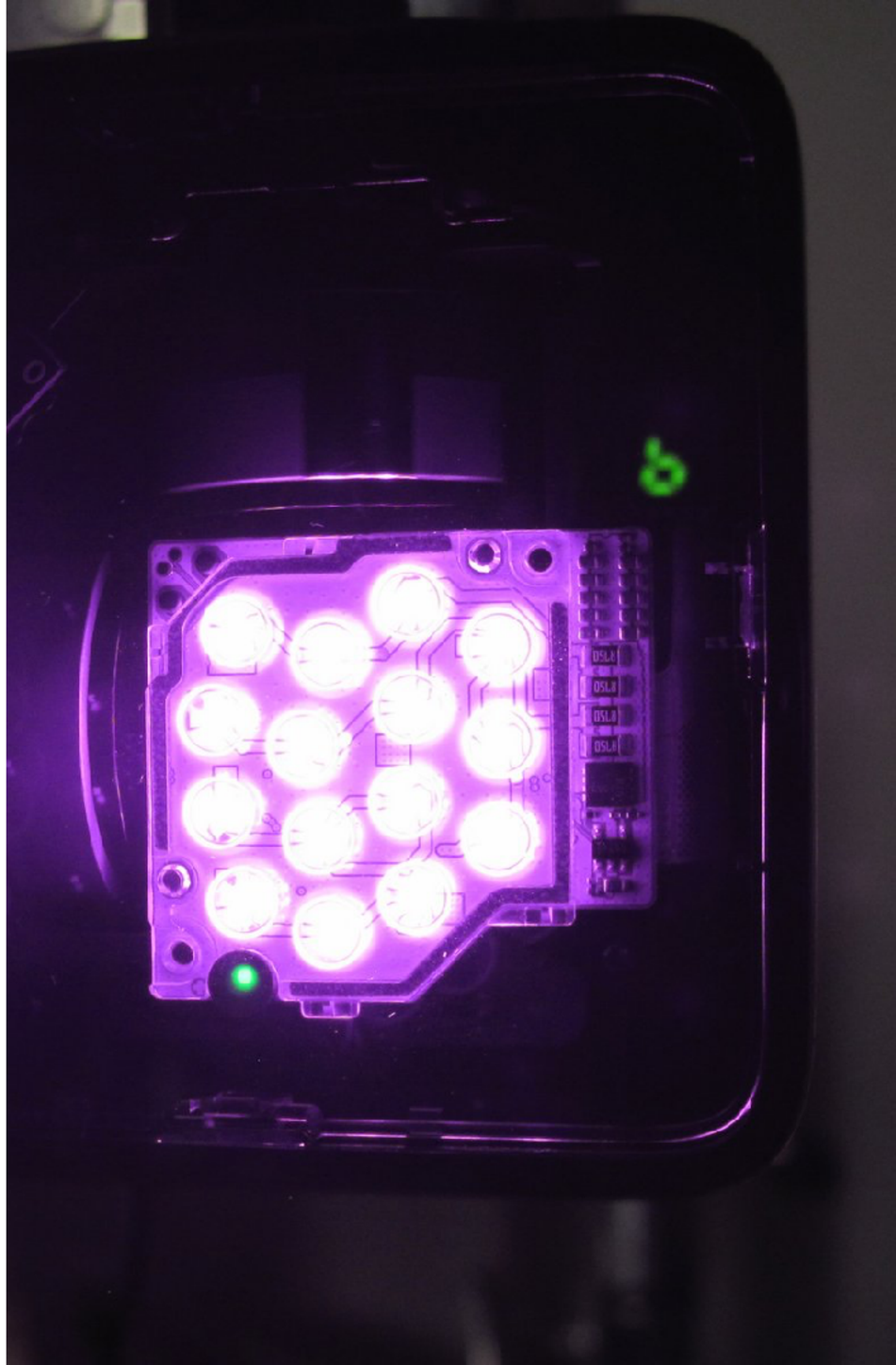
- Vertical rotation
- Rotates from -Y to +Y (Bottom-Up)
- Sweeps space with horizontal laser plane
- Allows vertical angle calculation (sensor angular position from XZ plane, lighthouse look vector).



Lighthouse v1.0

Synchronization IR diodes

- IR diodes flash, marking moment in time when given rotor laser emitter is pointing directly along one of axes (marking laser sweep start):
  - For horizontal rotor, points towards +X (right)
  - For vertical rotor, points towards -Y (bottom)
- That flash is registered by most of IR photoreceptors (sensors) on tracked device, and is distinguished from laser sweeps based on it’s duration
- That also synchronizes optically two base stations



Lighthouse v1.0
Synchronization flash

How do we know what synchronization flash represents?

- Lasers perform 60 full rotations around their respective axes per second. Which gives angular speed of 21.600 deg/s
- Here is youtube vide recorded at 10000 FPS: https://youtu.be/AbUOVswY5Lo?t=32
- This means that each frame corresponds to rotation by 2.16 degree
- Lighthouse v1.0 has 120 FOV, which means that there is 30 degrees of invisible FOV from rotation start.
- So from frame that recorded synchronization pulse, to frame on which we start to observe laser sweep, there should be: 30 degrees / 2.16 ~= 13 frames of latency
- If you step frame by frame from this flash moment at 0:32 (using period key when paused), it takes exactly 13 frames to see laser sweep confirming that theory :)

Lighthouse v1.0

Synchronization flash - sequence and data

- Synchronization flash has modulated duration
- There are 8 possible flash durations
- Duration index [0..7] represented as binary [000b .. 111b] specifies state of 3 separate data channels:
  - 1bit - Rotor on which laser sweep will happen
  - 1bit - If laser will be emitted or skipped
  - 1bit - Single bit of OOTX data descriptor

Lighthouse v1.0

Synchronization flash - OOTX data

- Synchronization flashes happen at 120Hz
- This gives OOTX data transfer rate of 15 Bytes/s
- With typical OOTX packet size of 33 bytes (v1.0), this means Lighthouse needs to be visible for minimum 2 seconds before it can be correctly identified
- OOTX data carries Lighthouse factory calibration data (needed to correct calculated angle from laser sweep), and its orientation in World Space (from Lighthouse IMU).

Lighthouse v1.0

Synchronization pulse vs rotation

- So if rotors spin at 60Hz, why do we have synchronization pulse at 120Hz?
- Thats because laser sweep is visible only on 120 degrees range of full rotation (120 out of 180 degrees on Lighthouse front).
- When one rotor emitter gets hidden in the back of Lighthouse, second rotor emitter starts to be visible (laser sweeps are shifted by half rotation and thus interleaved).
- Thus we have two overlapped laser sweeps at 120Hz which allows single pose calculation at 60Hz.

Lighthouse v1.0

Lighthouse order of operation

- Synchronization flash
  - Horizontal laser sweep
- Synchronization flash
  - Vertical laser sweep
- Synchronization flash
  - Idle rotation (no laser emission) <- Why idle ??
- Synchronization flash
  - Idle rotation (no laser emission)

Lighthouse v1.0

Lighthouses synchronization

- Till now we've discussed single Lighthouse, but they can work in pair, synchronized either by cable (A-B modes), or by mentioned synchronization flash (B-C modes).
- Lighthouses perform their laser sweeps in turn, thus after each two sweeps, Lighthouse is idle for next two, to let second one do it's job.
- When only one Lighthouse is used (B mode), it still skips two sweeps, which means that pose reconstruction can be executed at only 30Hz.

Lighthouse v1.0

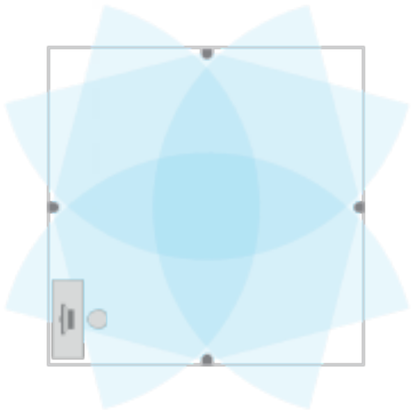
Putting it all together

- Time delta, between synchronization pulse, and laser sweep detection by photodiode sensor, is converted to angle (horizontal and vertical). Device registers those times using 48MHz clock, which gives needed precision.
- Those angles are then corrected, based on Lighthouse calibration data from OOTX.
- Having both angles, vector from Lighthouse center to device sensor is defined. We can then calculate sensor projected position (X,Y) at distance of 1.0 unit in front of lighthouse.
- This gives us base to running pose reconstruction algorithm based on perspective-n-point problem.
- We need projected positions calculated for only 4 sensors to resolve device orientation.

Lighthouse v2.0

Motivation

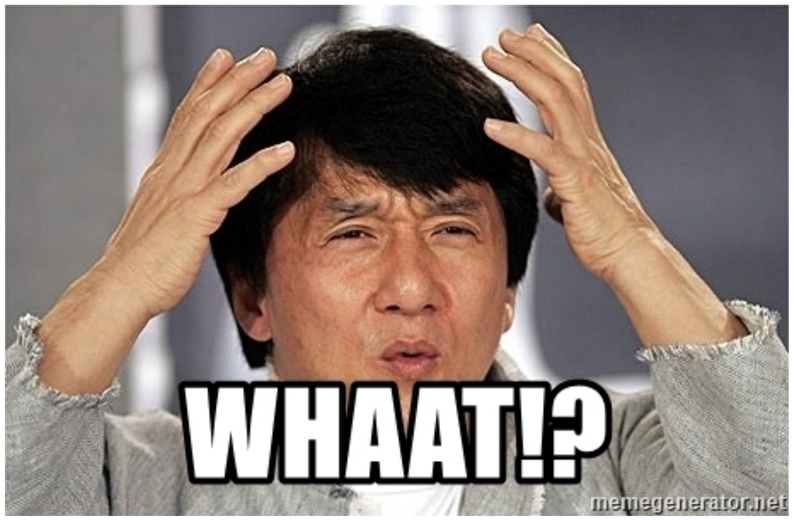
- Lighthouse v1.0 synchronization flash requires huge amounts of energy (half of whole energy used by the lighthouse goes to charge diodes)
- Lighthouse v1.0 moving parts (rotors) are first to break, ideally Lighthouse shouldn’t have any mechanical components
- More than 2 Lighthouses to scale tracking
- Bigger range of each Lighthouse

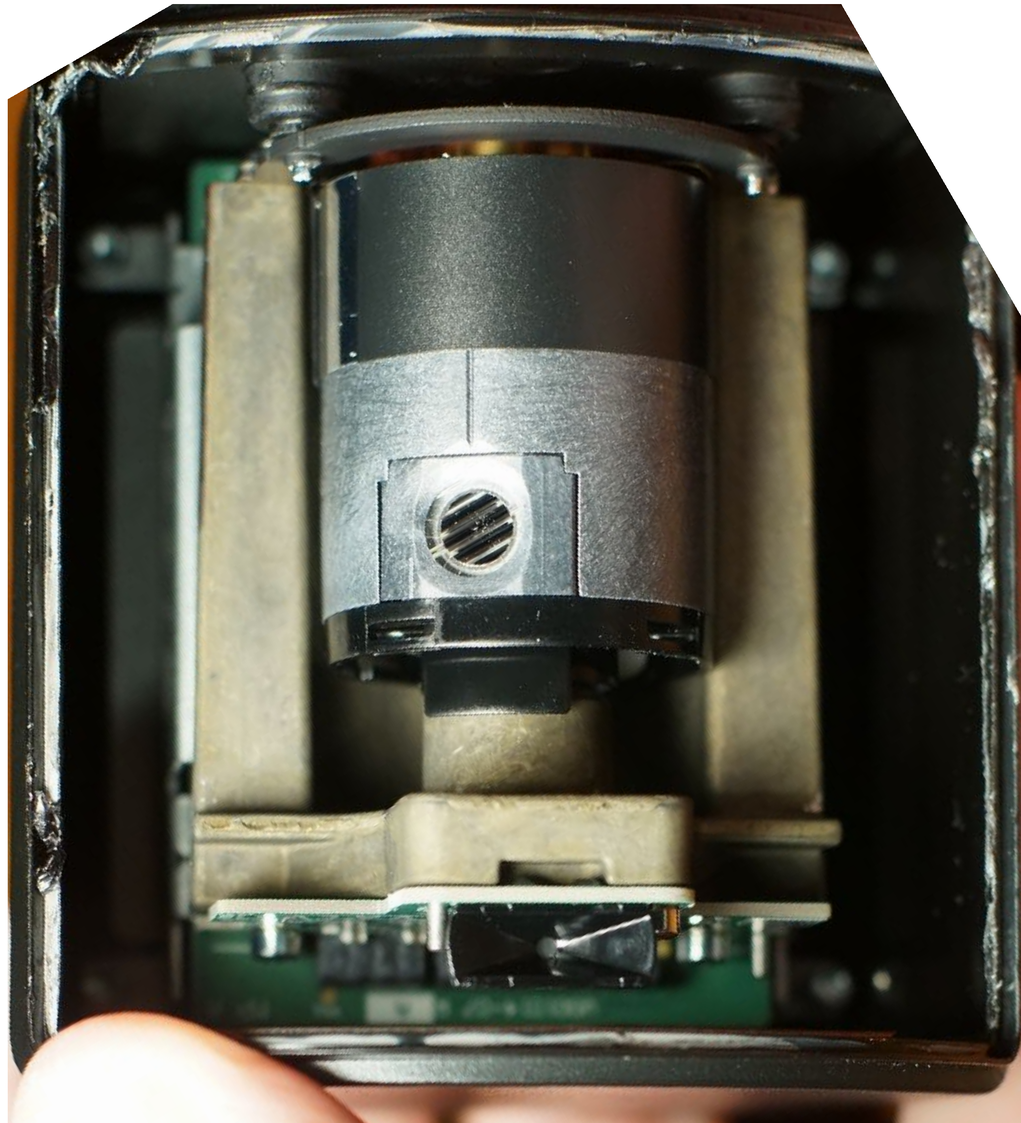


Lighthouse v2.0

Changes in construction

- No synchronization IR diodes?
- No Vertical rotor?
- Single horizontal rotor with 2 emitters?

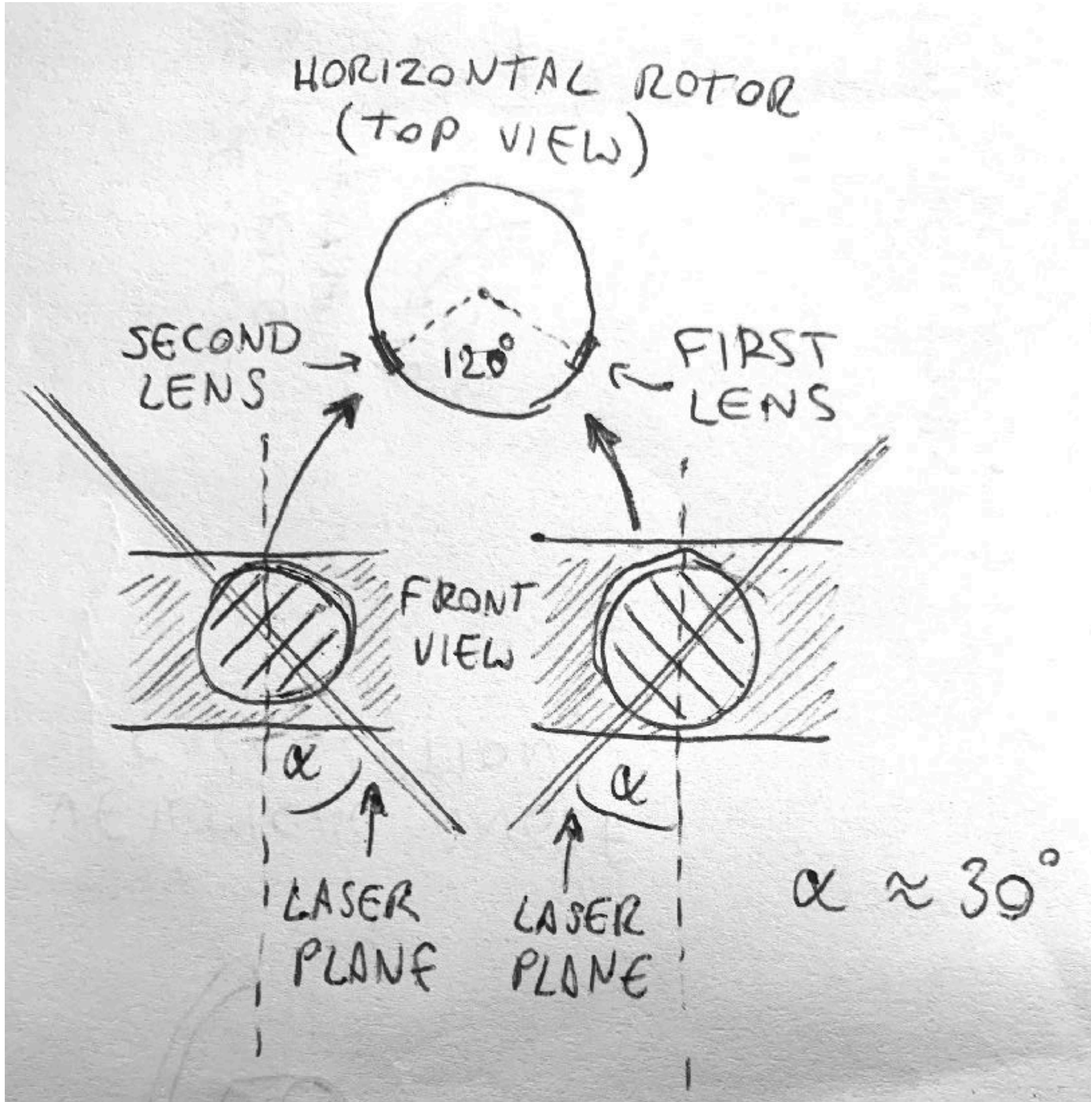




Lighthouse v2.0

Horizontal rotor and emitters

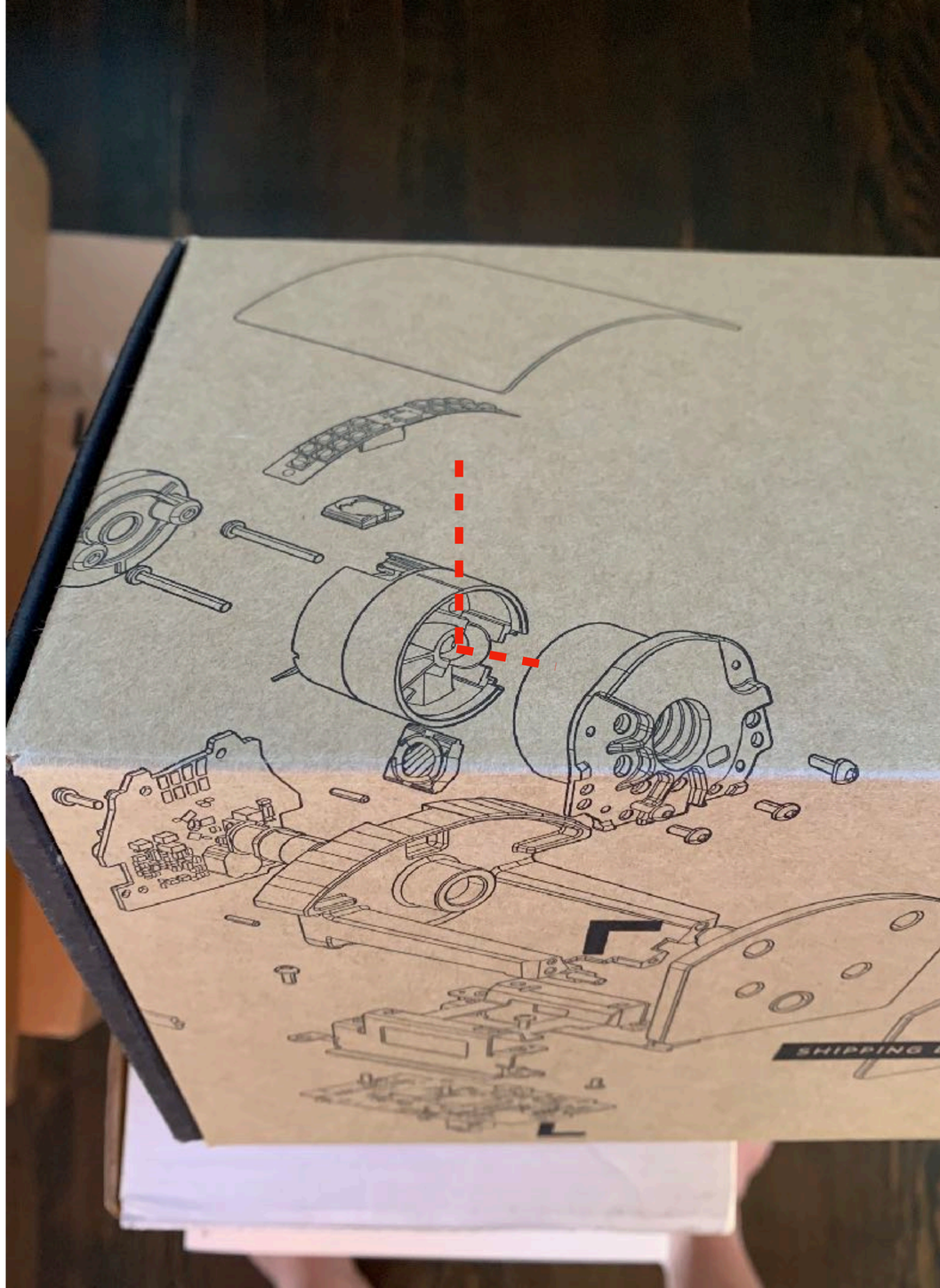
- Horizontal rotor has two lenses through which two laser planes are shaped.
- Lenses are separated at 120 degrees, and tilted by ~ +/- 30 degrees from vertical.
- This means that each laser plane is tilted as well, which gives interesting property that allows angular position calculation with only one rotor.



Lighthouse v2.0

Horizontal rotor and emitters

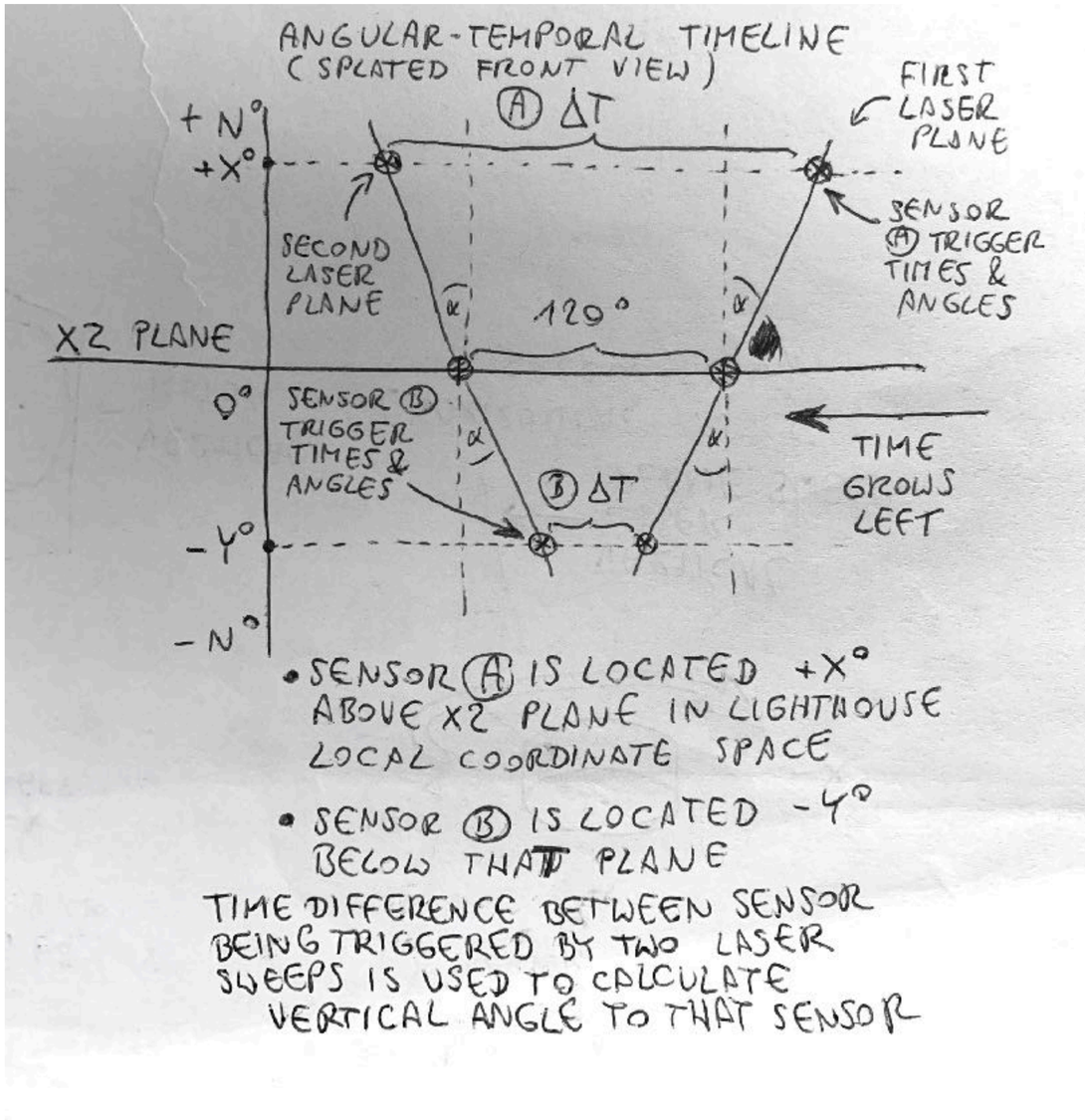
- How do we know lenses separation degree?
- Either we destroy Lighthouse to find out (which costs $200 each), or we check on the prototype box :)
- From diagram we can see it’s close to 120 degrees so now we need to check if math matches that assumption.
- PS: It does!



Lighthouse v2.0

Vertical angle calculation

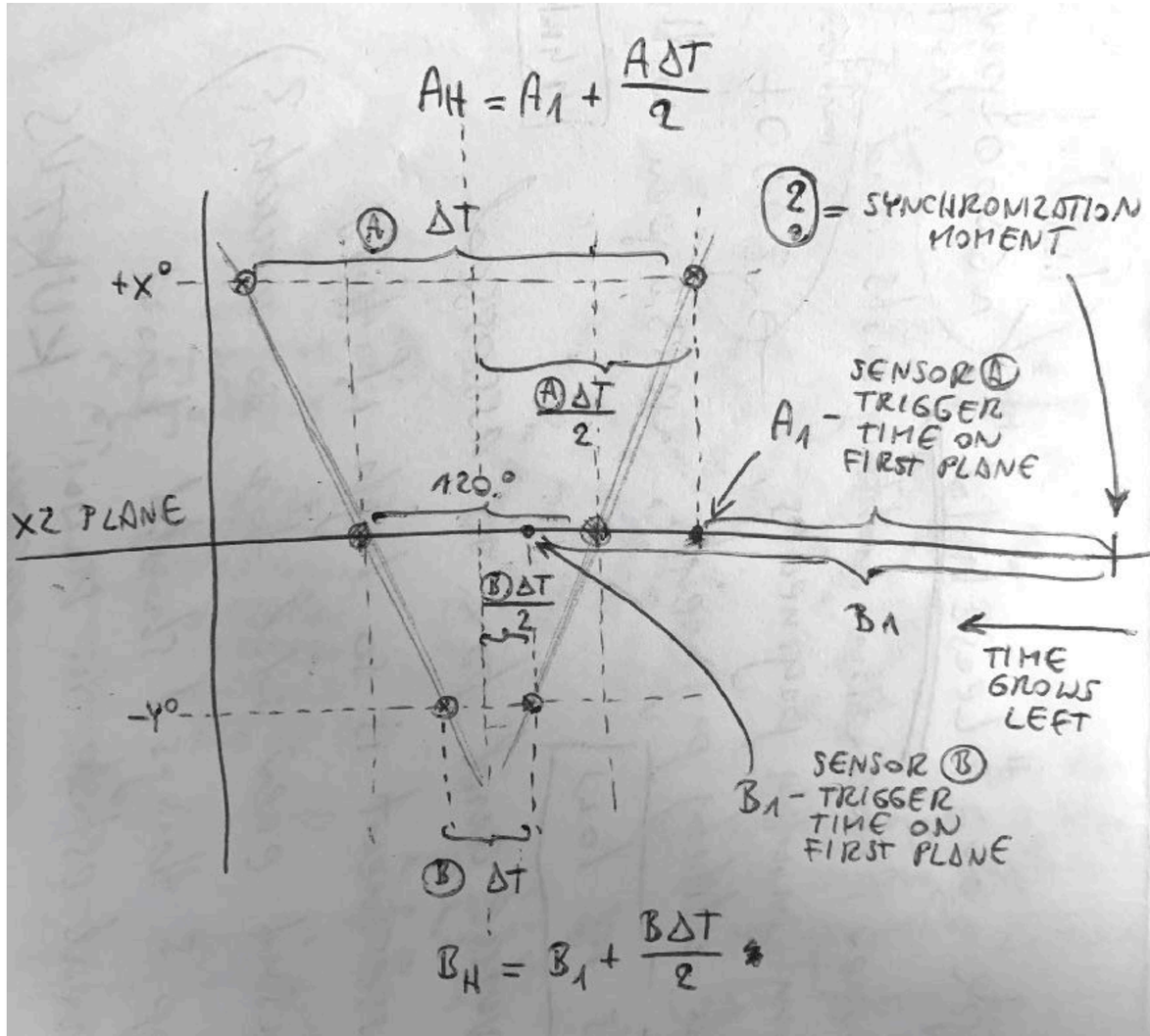
- Because laser planes are tilted, each device sensor will observe different time delta between two trigger moments
- The higher sensor is located, the bigger delta will be. Similarly the lower it is located, the smaller the delta.
- Knowing rotation speed, and dT, angular delta can be calculated, and from it and planes tilt, vertical angle from XZ plane at which sensor is visible.



Lighthouse v2.0

Horizontal angle calculation

- Horizontal angle is calculated as mid-point between two laser sweeps.
- Knowing dT between two laser sweep over given sensor, we can calculate it's horizontal angular position as half of that dT plus time since sync time.
- So where is synchronization pulse?
- How do we know that moment?
- How lighthouses synchronize without it?



Lighthouse v2.0

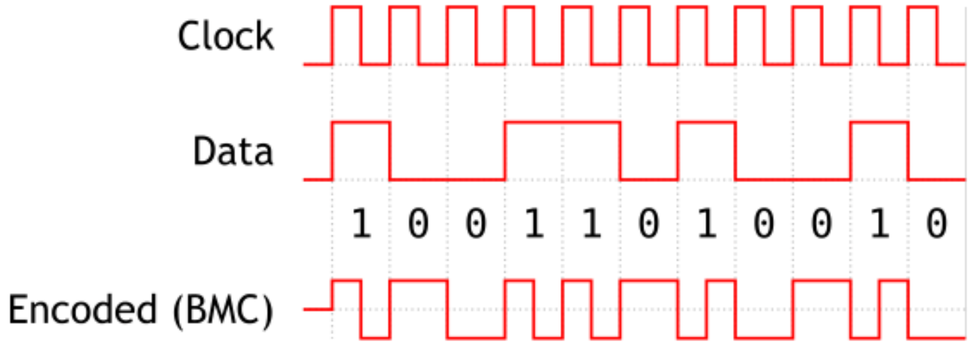
Lighthouses synchronization

- How they synchronize? They don’t :)
- Each Lighthouse rotates at slightly different frequency between 50Hz and 54Hz. Rotation speed is dictated by channel on which Lighthouse operates.
- There is 16 distinct channels. It is expected that device FW, processing laser sweeps detected by sensors, distinguishes to which LH given laser sweep belongs, based on those frequency differences.
- In theory this would allow for tracking area of any size.
- Current FW can detect up to 4 v2.0 lighthouses at given time.

Lighthouse v2.0

Laser modulation

- Laser beam is modulated. When it sweeps through the sensor, modulation is converted to stream of bits. Data broadcasted in laser sweep is encoded using Biphase Mark Code FM1 (BMC)
- Each data burst is sequence generated using 20 bit Linear Feedback Shift Register (LFSR). The longer the register, the bigger gap (between readings) can be detected and patched.



https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Conditioned_Diphase.html

- Two sequences are used, one to encode current dT from sweep start (so synchronization pulse is not needed anymore), and one to encode OOTX bit (so called slow data stream).

Lighthouse v2.0

Laser modulation

- Additionally each LH uses different LFSR state, and that can also be used to distinguish from which LH given laser sweep is coming.
- With slowest LH rotating at 50Hz, one rotation takes ~20ms. As LH CPU has 6MHz frequency (6000000 clock ticks per second) that means that full rotation time from sync to sync takes 120.000 clock ticks.
- 2^17 = 131.072 ticks, so we need to receive only 17 bits to be able to decode dT since sweep start.
- I suspect that thats the reason why v2.0 lighthouses operate at ~50Hz, to extend laser sweep time over sensor (which at 60Hz and 5m distance its only ~1.6us!). With 50Hz, sweep time over sensor is extended to ~1.92us.

Lighthouse tracking

Data decoding, what worked, what didn’t

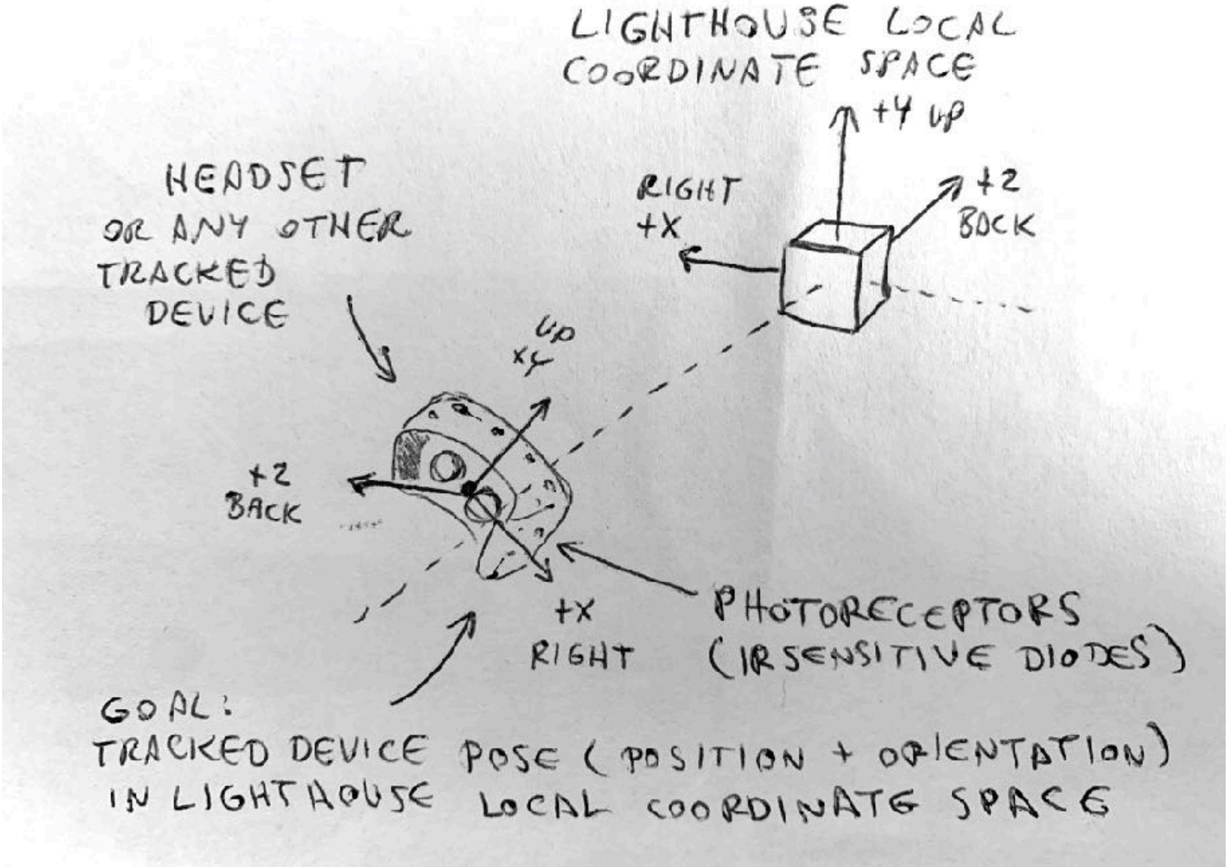
- There are 4 (!) distinctive ways in which Valve FPGA reports laser sweep timing back. They depend on if it’s v1.0 or v2.0 tracking HW, if device is wired or wireless. Currently we understand 3 out of those.
- Keeping host and device CPU clocks synchronized failed, as there is drift between the two that emerges when we try to re-synchronize them on device CPU ticks counter overflow (we get negative time and input order fails).
- Lighthouse tracking is very precise, thus part of calculation, of angular position, based on sweep timing deltas, need to be in fact performed directly on device CPU clock ticks, and later converted to host time and angles.
- Lighthouse tracking v2.0 devices operate on 96MHz clock for added precision.

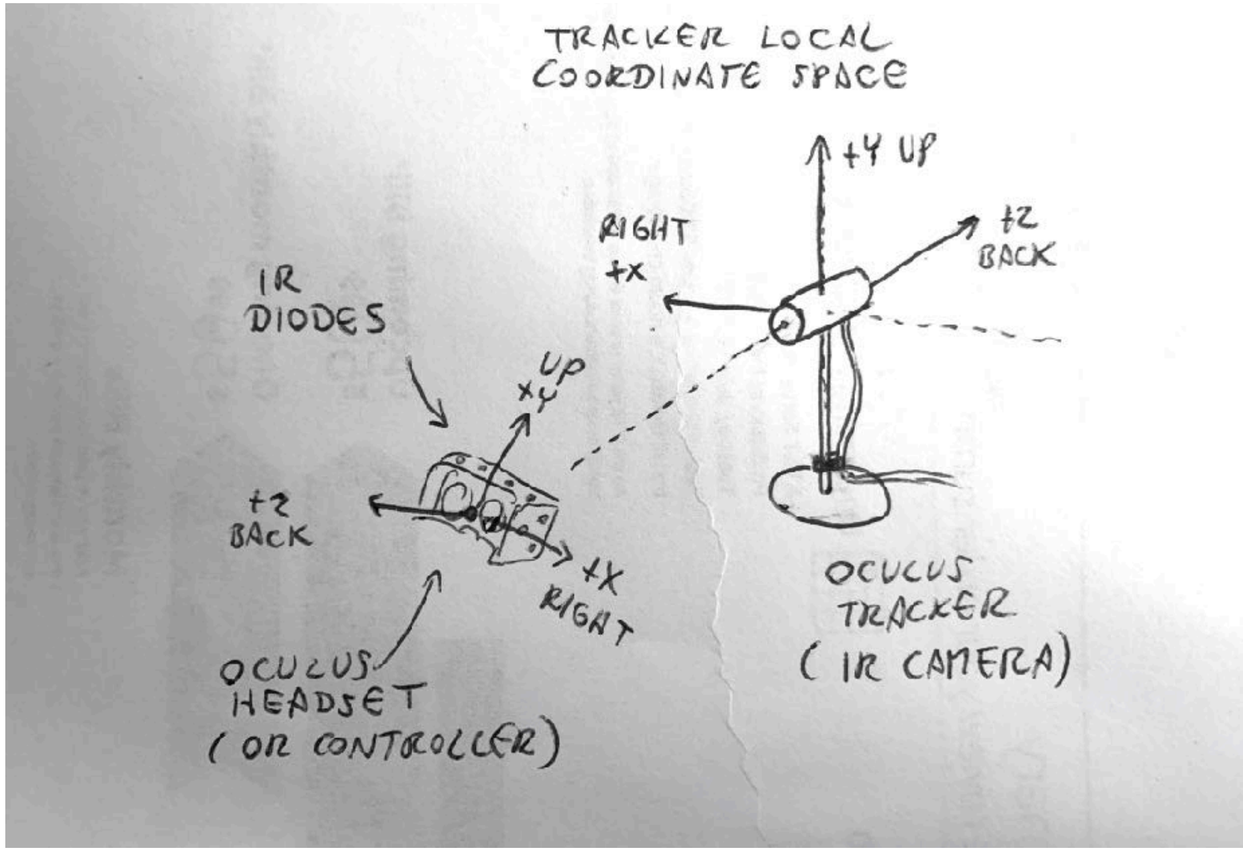
Oculus Constellation Tracking

Oculus Constellation Tracking

Introduction to problem

Persepctive-N-Point problem - Lighthouse (reversed) and Oculus (classic):





Oculus Constellation Tracking

IR diodes

- Oculus devices have IR diodes, hidden behind layer of IR transparent plastic (similar to Lighthouse devices having IR photoreceptors hidden behind such plastic).
- Those diodes are “characteristic points” observed by Tracker camera.

Oculus Constellation Tracking

Tracking camera

- Oculus Tracker is standard UVC webcam, but reporting on USB bus as “vendor-specific class” instead of “video-class” to prevent it’s automatic detection.

- Another protection is fact, that FW reports video feed as ‘YUYV', while in reality it’s 8bit greyscale ‘Y8’ or ‘GREY’.

  DK2 reports: 376x480 YUYV, is 752x480 Y8

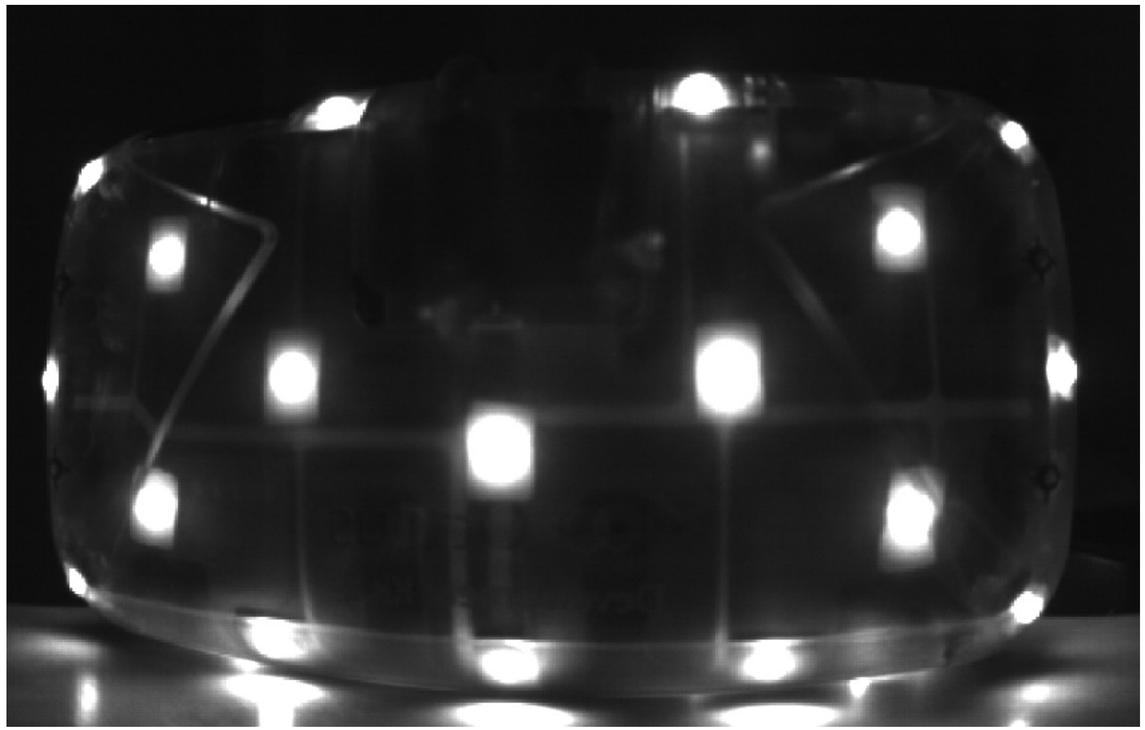
  Rift reports: 640x960 YUYV, is 1280x960 Y8

Color vs Infrared frame

Oculus Constellation Tracking

Tracking camera

- Camera has IR filter, so that only IR light comes through.
- Result is IR camera frame with bright points on it, representing observed IR diodes on tracked device (2D projection).
- This simplifies image post processing to calculate IR diodes (X,Y) position in the image.



Oculus Constellation Tracking

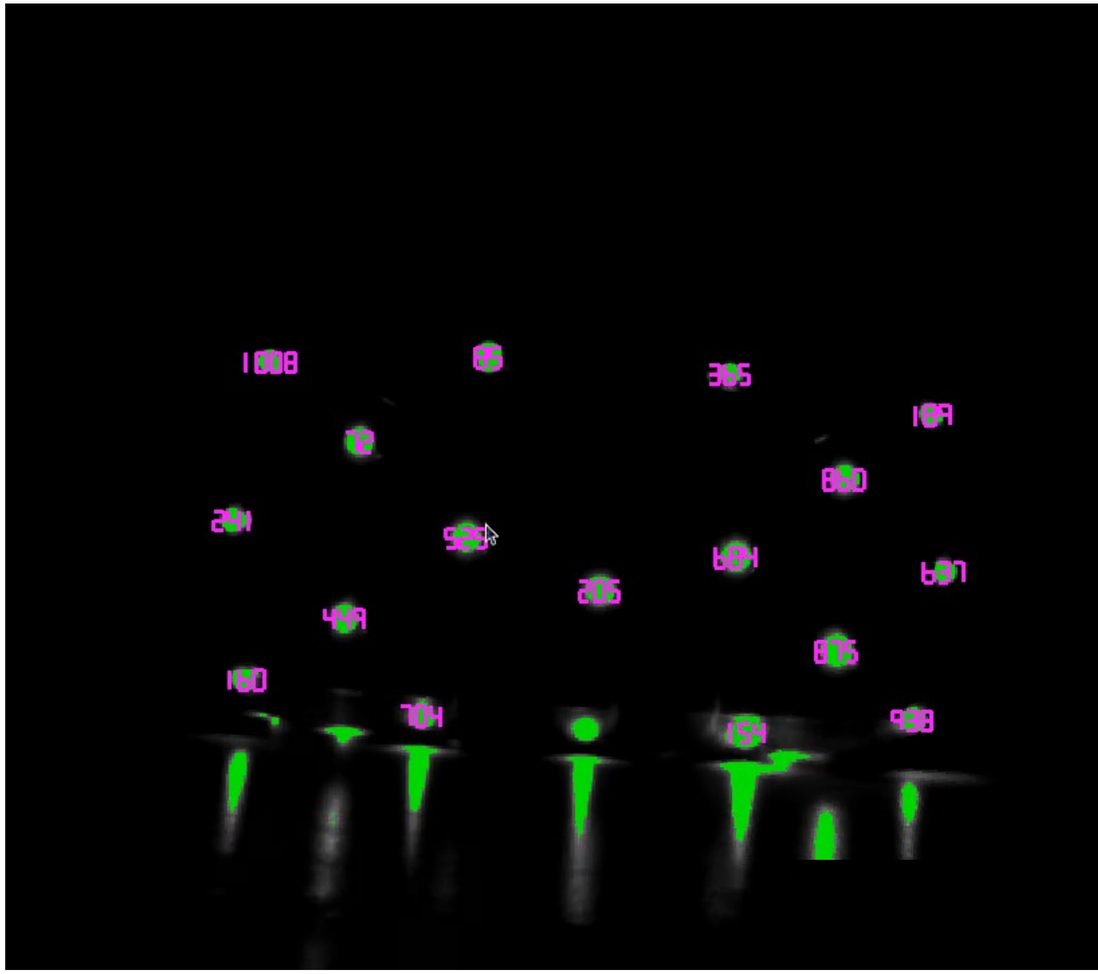
Tracking camera

- Camera exposure time needs to be synchronized with IR diodes flash time.
- In case of Oculus DK2 there is sync cable between headset and camera, and headset is triggering camera exposure (when configured in snapshot mode)
- This way IR diode flashes are in sync with their acquisition by camera.

Oculus Constellation Tracking

IR Diodes and projection correlation

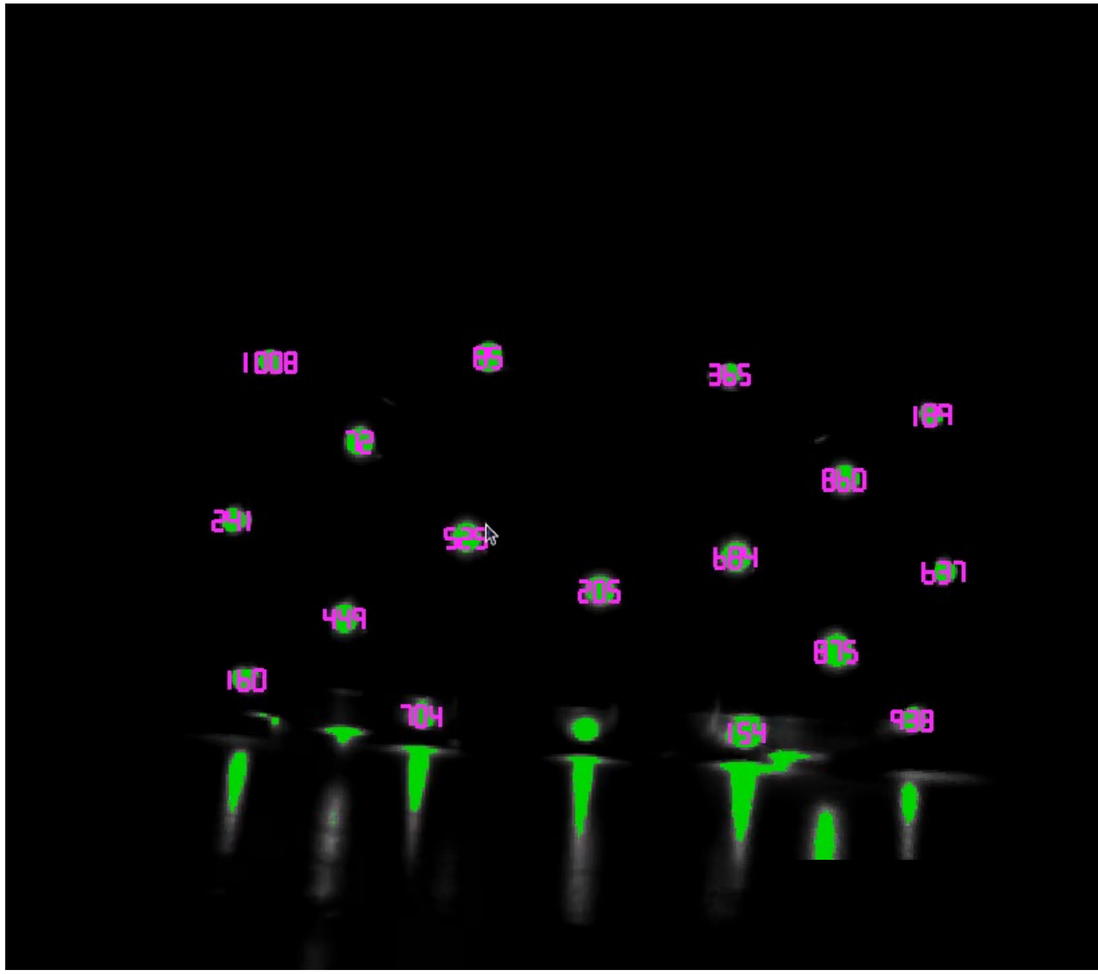
- How we know which dot in the frame represents which IR diode?
- Each IR diode has different 10bit pattern that it repeats (1 - Brighter, 0 - Darker).
- 10bits give 1024 possible combinations, and there is only 40 diodes on HMD, thus there is enough redundancy in case given IR diode is not seen for few frames.
- This also allows correction of mis-identified bits across frames.



Oculus Constellation Tracking

IR Diodes and projection correlation

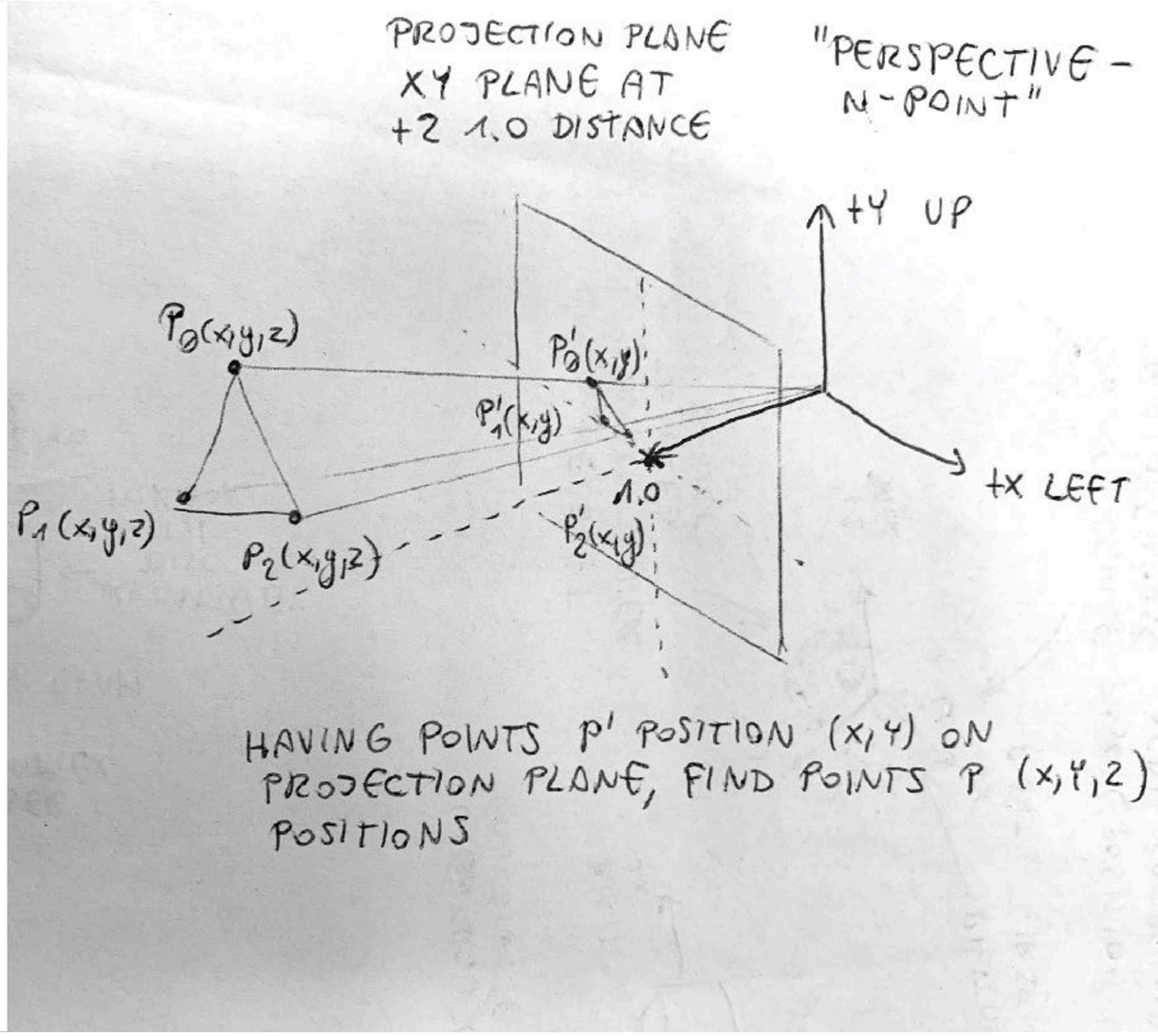
- Each diode 10bit pattern is specified in a way that minimum Hamming distance between any two patterns is 3 bits.
- So for each dot in a frame, we can compare it’s expected bit value to list of patterns and if Hamming distance is 1 or less we know we’ve found our IR emitter (this allows one off bit decoding auto-correction).



Oculus Constellation Tracking

Conclusion

- IR diodes position and orientation in device coordinate space, are read out from device itself (like in case of Lighthouse, based on factory calibr.)
- So now we have each IR diode 3D position in device space, and it’s 2D projection in Tracking Camera space.
- Thats everything we need to solve perspective-n-point problem.



Spatial tracking for VR

Conclusion of part 1

- We’ve went in this part through HW design and behavior, needed to provide spatial tracking capabilities to different VR systems.
- In next part we will go through pose reconstruction algorithms, pose prediction, sensor fusion and whole data path on example of current implementation and our future plans.