CEPTON®

Intelligence at the Speed of Light™

Introduction to Lidar and SDK

Cepton Webinar Series #1

2022-01-24

Table of Contents

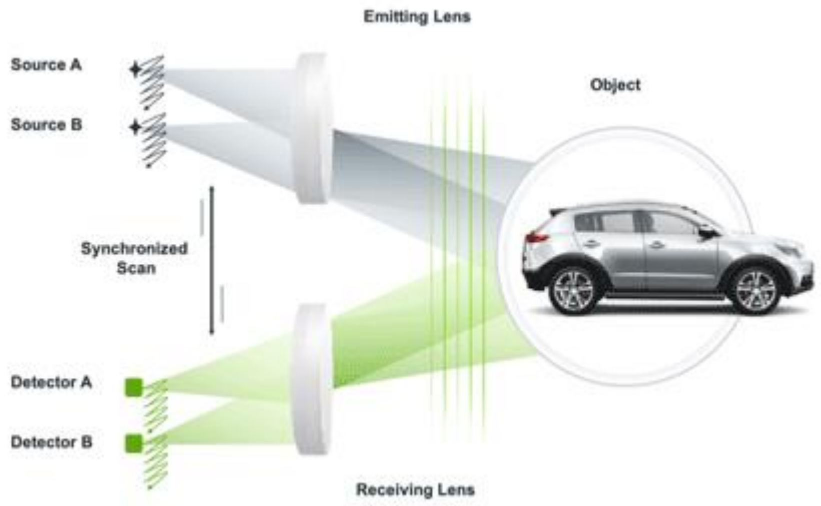
1. How lidar works
2. Lidar communication interfaces
3. Cepton SDK
4. Inspecting and debugging point cloud
5. ROS2 integration
6. Advanced topics
7. Topics for future episodes
8. We are hiring interns and new college grads!

Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page

How Lidar Works

How Lidar Measures Distance

- Time of flight (TOF) based on speed of light
  - Each nano-second is 30cm
- Active device with its own lighting (compared to camera)
  - Not all light comes back
  - Noise from other light sources
- Dynamic range is a challenge with practical implications
  - Saturation and low SNR (signal-to-noise ratio)
- Rolling shutter (compared to global shutter in most cameras).
  - Motion compensation can be important



How Lidar Works (continued)

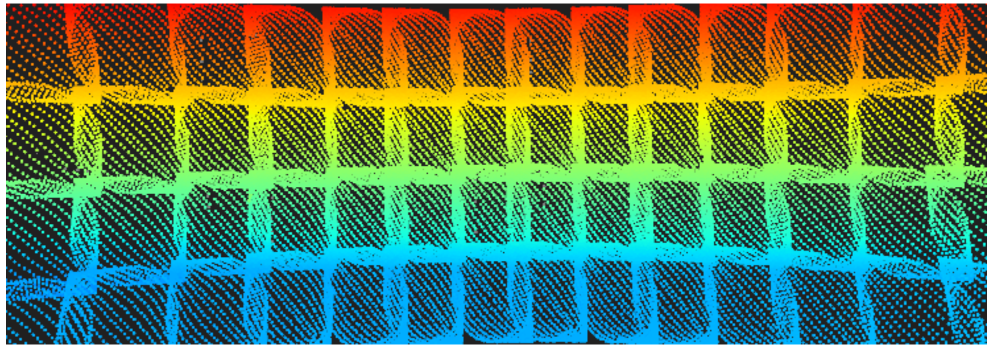
Scanning modality (MMT)

Key question to answer: How to send lasers to *all* directions?

	Cepton's MMT	Competitors
Is something spinning around?	No	Some are
Does it have mirrors?	No	Most non-spinners do. (MEMS or macro)
If not reflective, is it refractive?	No	Some ideas out there, no viable product seen.
It is completely not moving?	No	Some claim to be bend light with solid state medium. No such lidar is known to work well.
Is it related to special laser emission techniques?	No	Unknown
How does MMT work then?	Join Cepton to find out.	

Scan pattern

- Not a square grid
- Raw data is not always uniform in density
- Concept of "channel"
- Scan pattern impact perception algorithms.



Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page

How Lidar Works (continued)

Lidar Specifications and Limitations

- Lidar point density: up to 1M points per second, roughly 600x150 resolution.
  - Not normalized, raw mode only.
- Eye safety requirement dictates how strong the laser can be
  - Lidar uses infrared invisible laser that can still be strong
  - Always be careful with lidar even though it is eye safe

CAUTION
Class I invisible Laser Radiation Present.
Avoid long-term viewing of laser.

Compare lidar with camera

Lidar and camera augment each other, you need both.

	Lidar	Camera
How to measure	Active (own lighting)	Passive (rely on ambient mostly)
Imaging mode	Rolling shutter continuous measurements	Global shutter
Raw data	Following scan pattern, non-uniform	Usually hexagonal, normalized to square
Usage	Accuracy is important. Measure in absolute dimensions	Distortion is common and DNN works just fine
Algorithms	Classical size/shape works well. Reliability in 99% or better	Usually require DNN. Reliability in ~90%

Lidar Communication Interfaces

Ethernet (RJ45)

- Point data
  - Continuous stream of UDP point data (Not TCP)
  - UDP data are broadcast by default. Can be configured to multicast or unicast
  - Beware of high bandwidth usage
- INFO data
  - Self-discovery and reporting
  - Low frequency (1-2Hz)
- PTP/ARP etc.

COM port (DB9)

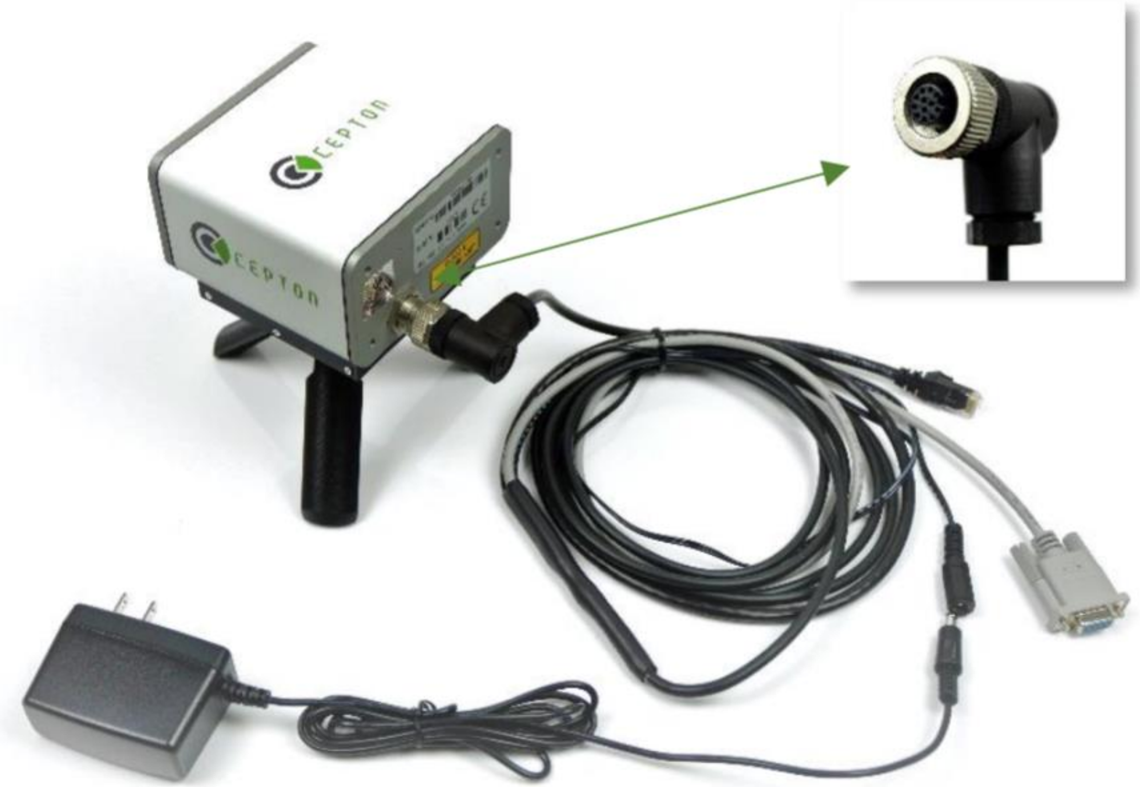
- Used to send GPS or IMU data
  - Can “pass-through” to ethernet
- Don’t use it. Use PTP instead if you can.
  - Not supported in the future products
  - PTP is better

PPS pin (pin9 of DB9)

- Don’t use it. Use PTP instead.

12V Power

- Consider power-over-ethernet on system level



Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page

Cepton SDK

Cepton's unified SDK

- All different lidars from Cepton work with the same SDK.

SDK is an ethernet receiver

- Ethernet code is user land. You only need SDK, no device driver required
- Ethernet code is asynchronous: By default, SDK starts a new thread and uses that thread to make callbacks
- SDK supports synchronous mode acting only as a parser

SDK supports advanced capture/replay

- Difference between live sensor and replay:
  - Pause/resume. Speed. Replay is ideal for data processing.
- Half the APIs are for replay functionalities.

SDK support matrix

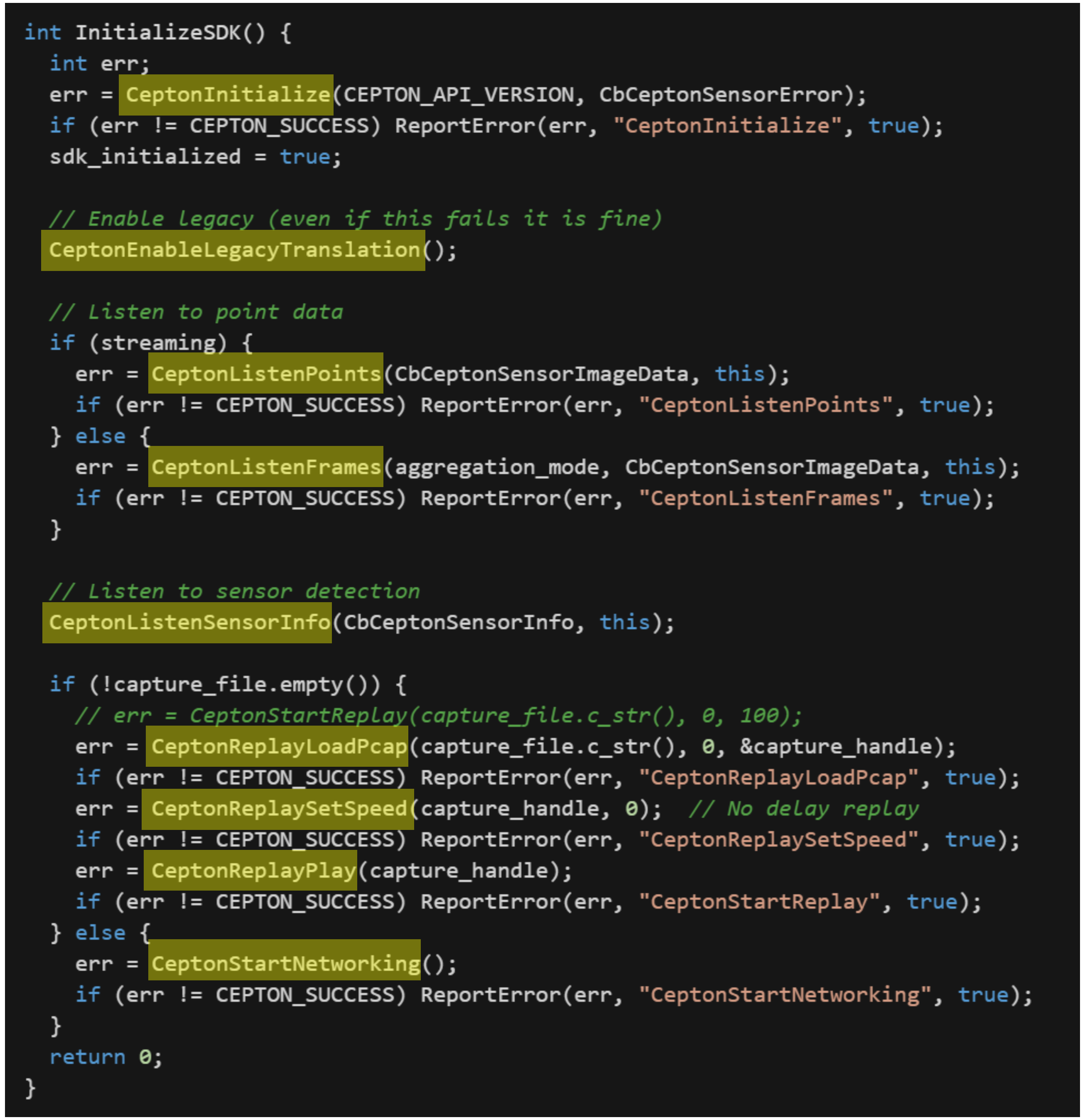
Operating Systems:
- Linux Ubuntu 18/20 (default)
- Linux Centos/RHEL/other (by request)
- Windows 10/11 (default)
- MacOS: M1 (by request)

Language Bindings
- C/C++: Native
- Python: Provided
- JavaScript: by request
- C#: by request

(We are happy to support more)

Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page

Introduction to SDK (continued)



```
int InitializeSDK() {
  int err;
  err = CeptonInitialize(CEPTON_API_VERSION, CbCeptonSensorError);
  if (err != CEPTON_SUCCESS) ReportError(err, "CeptonInitialize", true);
  sdk_initialized = true;

  // Enable legacy (even if this fails it is fine)
  CeptonEnableLegacyTranslation();

  // Listen to point data
  if (streaming) {
    err = CeptonListenPoints(CbCeptonSensorImageData, this);
    if (err != CEPTON_SUCCESS) ReportError(err, "CeptonListenPoints", true);
  } else {
    err = CeptonListenFrames(aggregation_mode, CbCeptonSensorImageData, this);
    if (err != CEPTON_SUCCESS) ReportError(err, "CeptonListenFrames", true);
  }

  // Listen to sensor detection
  CeptonListenSensorInfo(CbCeptonSensorInfo, this);

  if (!capture_file.empty()) {
    // err = CeptonStartReplay(capture_file.c_str(), 0, 100);
    err = CeptonReplayLoadPcap(capture_file.c_str(), 0, &capture_handle);
    if (err != CEPTON_SUCCESS) ReportError(err, "CeptonReplayLoadPcap", true);
    err = CeptonReplaySetSpeed(capture_handle, 0);  // No delay replay
    if (err != CEPTON_SUCCESS) ReportError(err, "CeptonReplaySetSpeed", true);
    err = CeptonReplayPlay(capture_handle);
    if (err != CEPTON_SUCCESS) ReportError(err, "CeptonStartReplay", true);
  } else {
    err = CeptonStartNetworking();
    if (err != CEPTON_SUCCESS) ReportError(err, "CeptonStartNetworking", true);
  }
  return 0;
}
```

Notes:

- C-style interface inside dynamic library
- Listener model:
  - Get called back as soon as data arrive.
  - Exclusion: No more data arrive until current callback returns
  - Choose streaming vs. frame mode
- Explicit call to `StartNetworking()`
  - This is after the `ListenFrames()` is called
  - No networking needed when processing captures
  - Flexible networking parameters without cluttering the API.
- Use `EnableLegacyTranslation()` to support earlier parts like Vista-P
- Advanced network packet parsers and hooks (not shown in code)

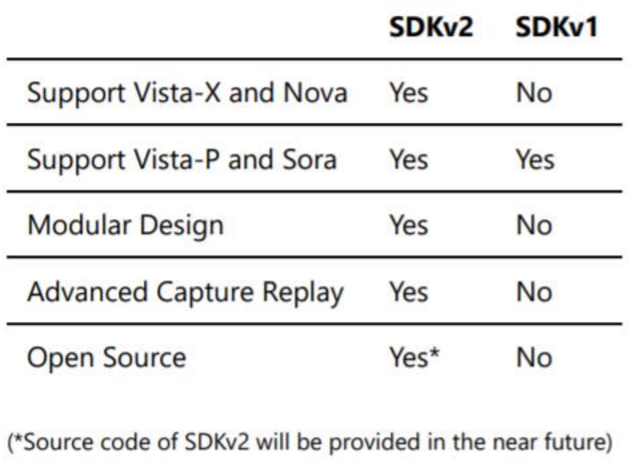
Introduction to Cepton SDK (continued)

Coordinate system

- Cartesian coordinate system with origin at the geometric center of the sensor box (excluding connectors)
  - Standing behind the senor, each axis from small to large
    - X: left to right
    - Y: near to far
    - Z: low to high

A word on legacy SDK (SDK 1.x)

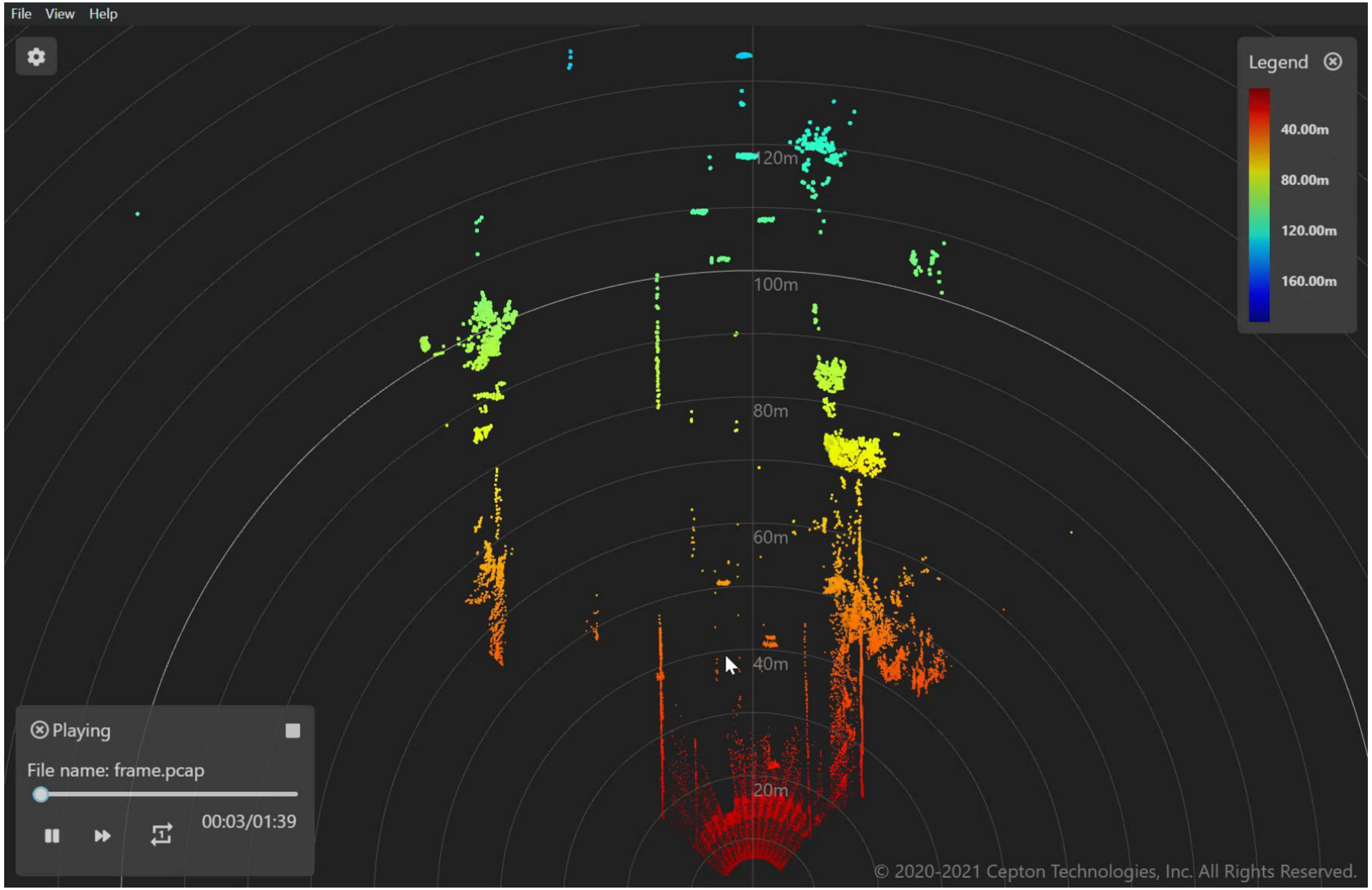
- All the SDK we talked about are the new SDK (v2)
- Please use Cepton SDK v2 to write your lidar applications even if you are working with Vista-P series products.



	SDKv2	SDKv1
Support Vista-X and Nova	Yes	No
Support Vista-P and Sora	Yes	Yes
Modular Design	Yes	No
Advanced Capture Replay	Yes	No
Open Source	Yes*	No

(*Source code of SDKv2 will be provided in the near future)

Inspection and Debugging Point Cloud



File name: frame.pcap

© 2020-2021 Cepton Technologies, Inc. All Rights Reserved.

Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page

Inspection and Debugging Point Cloud (continued)

Data Captured

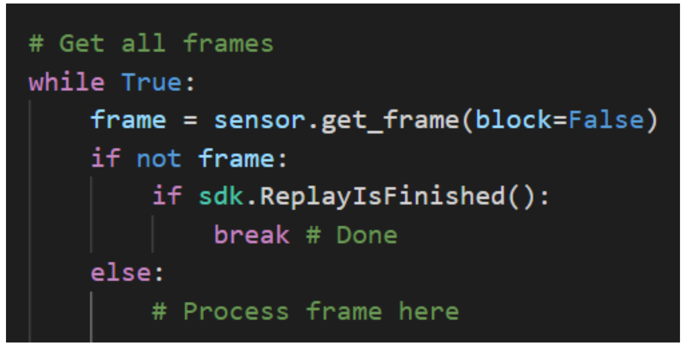
- Raw network traffic (PCAP)
- Cepton Replay format (CR)
  - CR supports object and other perception primitives
- Rosbag if you use ROS

Use Cepton SDK (C/C++)

- Play/Pause/Seek
- Fixed speed (correct time) or offline mode (as fast as possible)
- Through callbacks of frames or packets

Use Cepton's Python SDK

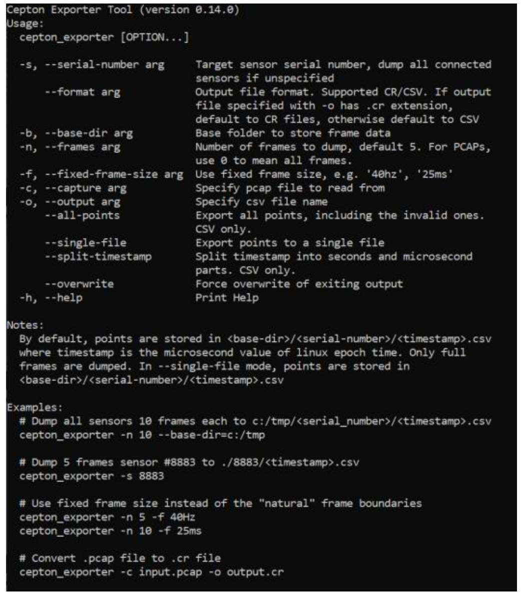
- Direct Mode



```
# Get all frames
while True:
    frame = sensor.get_frame(block=False)
    if not frame:
        if sdk.ReplayIsFinished():
            break # Done
    else:
        # Process frame here
```

Use cepton_exporter tool

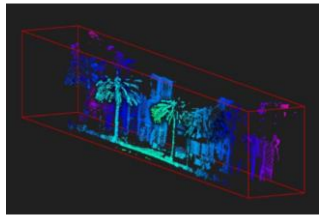
- Convert PCAP to CSV



```
Cepton Exporter Tool (version 0.14.0)
Usage:
  cepton_exporter [OPTION...]

  -s, --serial-number arg    Target sensor serial number, dump all connected
                             sensors if unspecified
      --format arg           Output file format. Supported CR/CSV. If output
                             file specified with -o has .cr extension,
                             default to CR files, otherwise default to CSV
  -b, --base-dir arg         Base folder to store frame data
  -n, --frames arg           Number of frames to dump, default 5. For PCAPs,
                             use 0 to mean all frames.
  -f, --fixed-frame-size arg Use fixed frame size, e.g. '40hz', '25ms'
  -c, --capture arg          Specify pcap file to read from
  -o, --output arg           Specify csv file name
      --all-points           Export all points, including the invalid ones.
                             CSV only.
      --single-file          Export points to a single file
      --split-timestamp      Split timestamp into seconds and microsecond
                             parts. CSV only.
      --overwrite            Force overwrite of exiting output
  -h, --help                 Print Help

Notes:
  By default, points are stored in <base-dir>/<serial-number>/<timestamp>.csv
  where timestamp is the microsecond value of linux epoch time. Only full
  frames are dumped. In --single-file mode, points are stored in
  <base-dir>/<serial-number>/<timestamp>.csv

Examples:
  # Dump all sensors 10 frames each to c:/tmp/<serial_number>/<timestamp>.csv
  cepton_exporter -n 10 --base-dir=c:/tmp

  # Dump 5 frames sensor #8883 to ./8883/<timestamp>.csv
  cepton_exporter -s 8883

  # Use fixed frame size instead of the "natural" frame boundaries
  cepton_exporter -n 5 -f 40Hz
  cepton_exporter -n 10 -f 25ms

  # Convert .pcap file to .cr file
  cepton_exporter -c input.pcap -o output.cr
```

Use Cepton Viewer

- Best for intuitive understanding
- Works for PCAP/CR
- Can inspect area of interest in detail

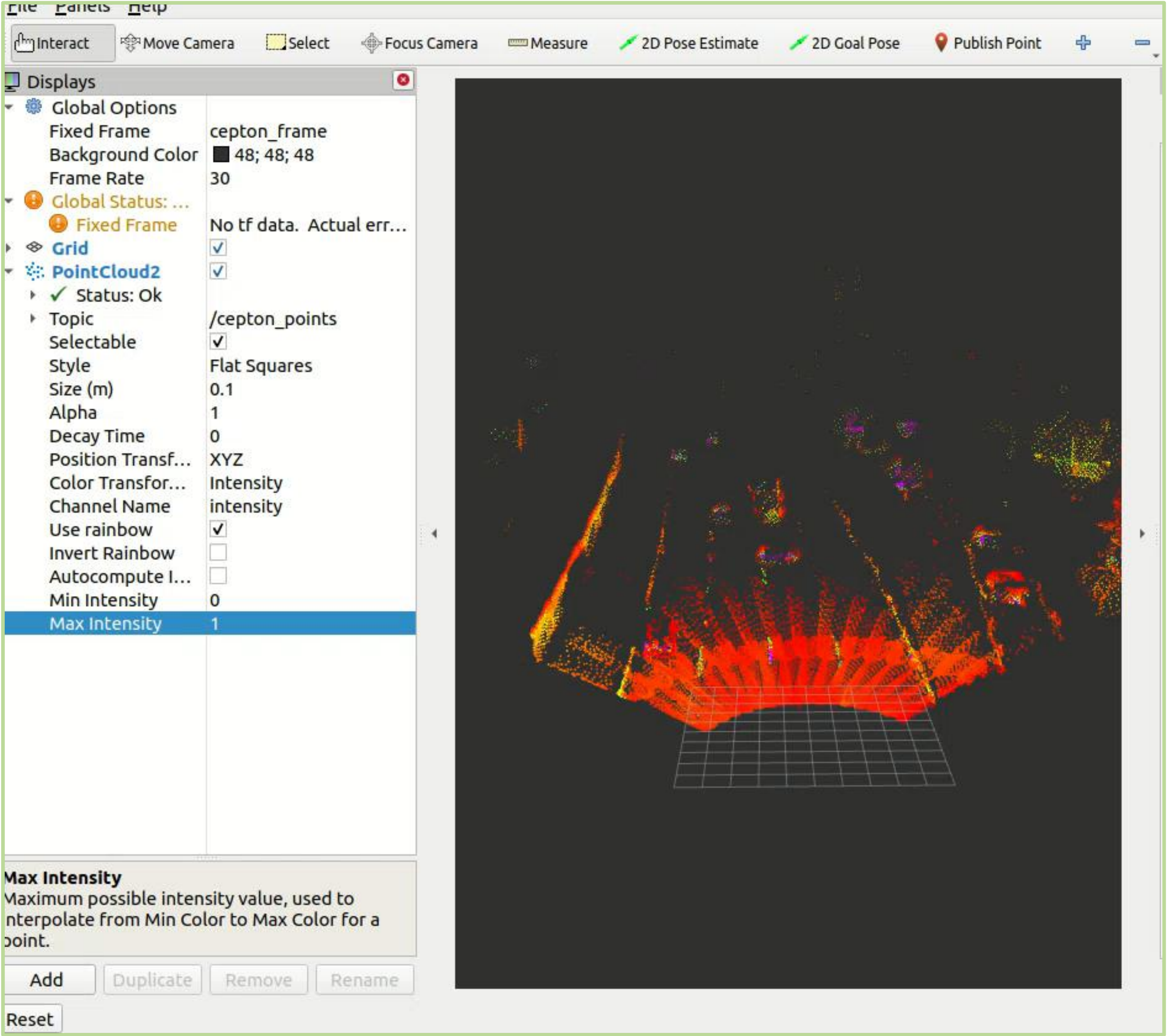


Use Wireshark

- Important debugging tool
- Ideal for trouble shooting live sensor connections
- Get for advanced communication (time sync or device config)

ROS2 Support

- ➢ Supports ROS2 distributions for Ubuntu 18 and 20
- ➢ Publishes Cepton info messages and sensor point cloud topics
- ➢ Compatible with RVIZ
- ➢ Use ROS command line arguments to configure SDK settings and replay PCAP



Global Options	
Fixed Frame	cepton_frame
Background Color	■ 48; 48; 48
Frame Rate	30
Global Status: ...	
Fixed Frame	No tf data. Actual err...
Grid	☑
PointCloud2	☑
✓ Status: Ok	
Topic	/cepton_points
Selectable	☑
Style	Flat Squares
Size (m)	0.1
Alpha	1
Decay Time	0
Position Transf...	XYZ
Color Transfor...	Intensity
Channel Name	intensity
Use rainbow	☑
Invert Rainbow	□
Autocompute I...	□
Min Intensity	0
Max Intensity	1

Max Intensity
Maximum possible intensity value, used to nterpolate from Min Color to Max Color for a point.

Advanced Topics

High data bandwidth programming

- Be careful with copying. Megs of data per frame adds up very quickly.
- Be very careful with dynamic allocations. Use `std::array`, not `std::vector`, if you can help it.
- Avoid anything that is slower than O(NlogN) where N is number of points.

Asynchronous relay

- Data do get lost with live sensor when processing speed is slow. That might be OK if handled properly.
- Decouple processing thread from networking through a FIFO. (checkout `AsyncRelay` supported by SDK)

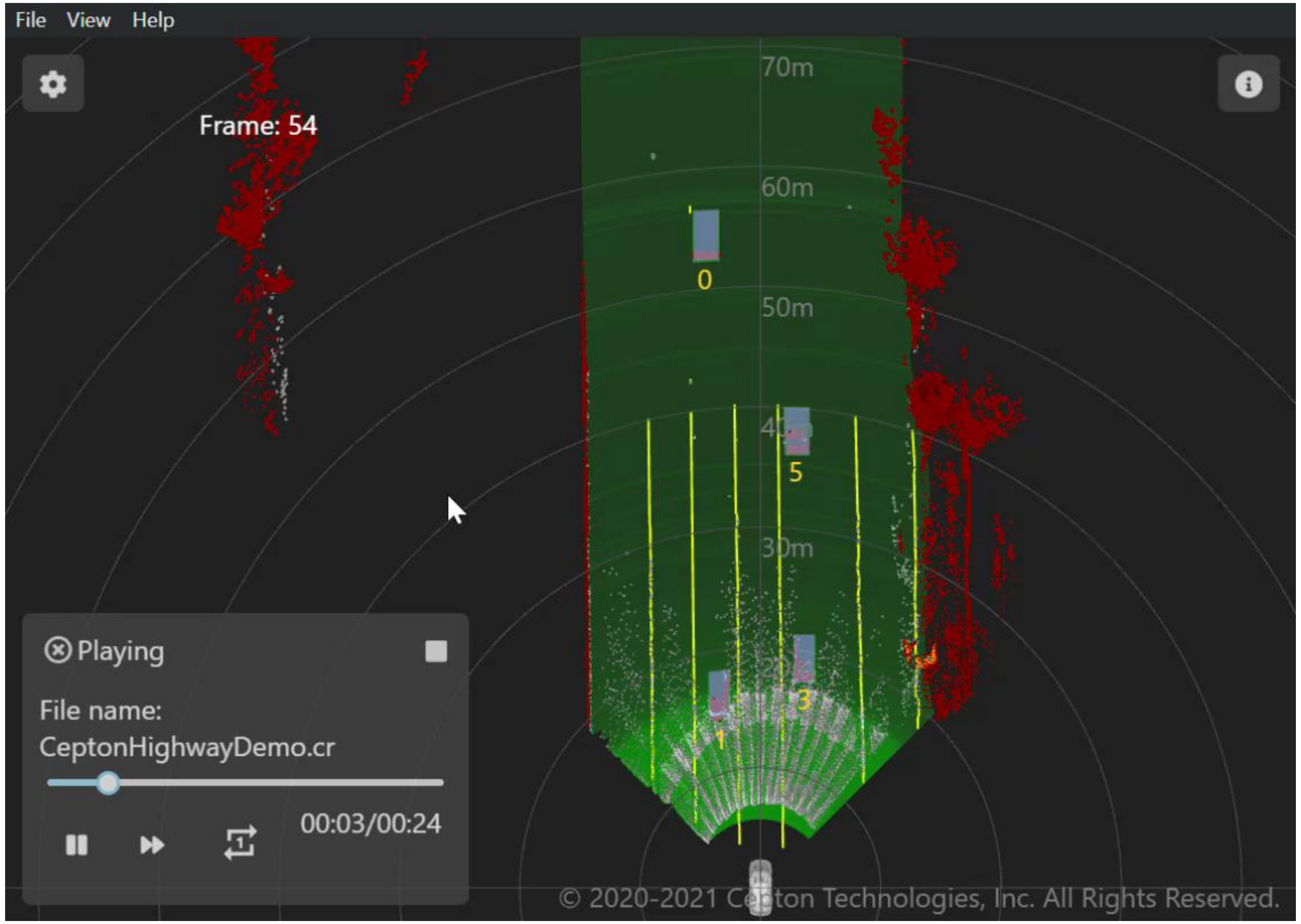
Time synchronization and sensor fusion

- Sensor fusion requires timestamps to be synchronized within a few milliseconds.
- System time for network packet on arrival is not good enough.
  - Congestion or CPU contention will cause packets to arrive in bursts.
- GPS is hard to use: You need to split the wires and do some soldering; cheap GPS brands are not always reliable.
- PTP is recommended.
  - Accurate to sub-microseconds.
  - Supported by Linux (ptp4l).
  - Requires a “hardware timestamp” ethernet chip (no USB dongles).

Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page

Future Topics For Webinar

- Best practice with rolling shutter, motion compensation.
- MMT and scan pattern.
- Automatic alignment across sensors, sensor fusion.
- Advanced SDK and SDK internals.
- ROS2 integration in-depth.
- Python SDK and offline data processing.
- Cepton's perception system and CR file.



File name:
CeptonHighwayDemo.cr

Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page

Resources

- Developer Center (https://developer.cepton.com coming soon…)
  - This and all other webinars
  - Download SDK package
  - Download Cepton Viewer executable
- Source code on github (https://github.com/cepton/)
  - SDK open-source repository (coming soon…)
- Official cepton.com
- JOB postings: LinkedIn and Handshake

Cepton is hiring interns and new college grads. Check out our LinkedIn or Handshake job page