title: Object Detection on Point Cloud Streams
slug: proposal-qiliu0523-object-detection-on-point-cloud-streams
author: Eddie Liu
date: 2025-03-23 21:00:00 UTC-07:00
tags: proposals, hard, 350 hours, LivePose
type: text

Name and Contact Information

- name: Eddie Liu
- website: GitHub, LinkedIn
- gitlab username: qiliu0523
- timezone: PDT (Pacific Daylight Time) San Francisco

Title

Object Detection on Point Cloud Streams

Synopsis

This project comprises two parts: prototyping a real-time Python pipeline for object detection on point cloud data, and integrating it into LivePose.

The goal is to build a functional pipeline that reads streaming point cloud data from various sources (e.g., Orbbec Femto Megas, Intel RealSense, stereo cameras), runs deep learning inference using models like PointNet++, applies filters for smoothing and tracking, and streams the results via OSC. During prototyping, the system will be evaluated across multiple models, input types, and hardware configurations to assess real-time performance and reliability. Once validated, the pipeline will be integrated into LivePose, leveraging its modular infrastructure for input handling, filtering, and output.

By the end of the project, LivePose will support end-to-end processing of streaming 3D point cloud data in at least one working configuration, with clear documentation and performance benchmarks to guide future development and extension.

Benefits to Community

This project brings real-time 3D point cloud processing to LivePose, enabling developers and researchers to work with depth-aware spatial data in an open, extensible pipeline.

Today, using state-of-the-art 3D models in live applications remains a major technical hurdle, due to fragmented tools, hardware dependencies, and limited real-time support. This project offers an end-to-end solution that significantly lowers the barrier to entry. Developers get a working, modular setup that connects sensors, models, filtering, and output in a way that's easy to adapt and extend.

This ease of use also opens up creative and socially impactful possibilities. Artists and performers can incorporate real-time 3D object presence into immersive installations, like reactive visuals, sound, or spatial interactions based on physical objects in space. Educators and accessibility designers can prototype touchless spatial interfaces, where objects or gestures trigger meaningful actions.

By contributing this to the LivePose ecosystem, the work empowers a broad community, technical and creative alike, to explore how machines can perceive and respond to the 3D world around them.

Deliverables

Milestone 1: Research and Planning

Before official start Apr 8 - May 7 (4w)

- Research various cameras and the SDKs to read inputs
  - RGB-D cameras like Intel Realsense, Kinect, Orbbec
  - The team is currently using Orbbec Femto Megas with the Orbbec SDK for reading point cloud data, so it'll be good to get familiar with it
  - Stereo cameras like Stereolabs ZED
  - LiDAR
- Download and get familiar with existing datasets like KITTI and nuScenes
- Test Open3D and OpenCV to visualize point clouds from existing datasets

Before official start May 8 - May 25 (2w)

- Research the models to use; initial ideas are:
  - PointNet++, classical deep learning on point clouds
  - ONNX models listed in this repo
  - Convert PyTorch models in MMDetection3D to ONNX format
- Download all the ONNX models
- Test running the models

Before official start May 26 - Jun 1 (1w)

- Study the LivePose code repo
  - Run LivePose with different configs
  - Get familiar with the code structure

- This is technically not necessary for prototyping, but it's good to have this in mind when building the prototyping pipeline

Week 1 Jun 2 - Jun 8

- Finalize research and planning

Deliverable 1 Due Jun 8 - Research and Planning

- [Required] A plan for prototyping
  - Decide what to prototype for each of the four main components: camera, backend, filters, outputs
- [Required] Basic familiarity with related techniques
  - Open3D / OpenCV
  - Relevant SDKs, e.g., Orbbec SDK, pyrealsense2, OpenNI2
  - ONNX framework
  - OSC / WebSocket
  - LivePose
- [Required] Get ready for prototyping
  - Dev environment set up
  - Cameras set up and running
  - Datasets downloaded and tested
  - Models downloaded and tested

Milestone 2: Prototyping Pipeline

Summary: 2w for cameras, 3w for models, 1w for filters, 1w for outputs, 1w for documentation.

Week 2-3 Jun 9 - Jun 22

- Implement camera reading pipeline. It should read streaming RGB + depth images from cameras, convert them to point cloud, and visualize the point cloud or print simple stats to confirm successful reading.
- Work with different input sources in the following order (list may be adjusted in Milestone 1):
  - [Required] Static point cloud (no streaming)
  - [Required] Existing datasets
  - [Required] Orbbec Femto Megas
  - [Required] Intel Realsense
  - [Optional] Stereo cameras
  - [Optional] LiDAR
- There is ongoing research in LivePose using Orbbec Femto Megas, which I could leverage to record some point cloud data and use it for validation

Week 4 Jun 23 - Jun 29

- Implement model inference with PointNet++. It should output a per-frame object detection.
- Evaluate real-time performance, including FPS supported for GPU vs CPU.

Week 5-6 Jun 30 - Jul 13

- Work with more complex models like PointPillars and CenterPoint.
- Evaluate real-time performance for different model + compute resource configs.
- [Optional] Evaluate accuracy and provide recommendations on what works well in different scenarios
- GSoC mid-term evaluation

Week 7 Jul 14 - Jul 20

- Implement the filter component. The pipeline may aggregate frames here for smoothing, movement tracking, or speed estimate.

Week 8 Jul 21 - Jul 27

- Implement the output component
  - [Required] OSC
  - [Optional] WebSocket
- The output part is more certain than other parts because LivePose already has it. Given this is a prototype, I'll probably showcase it with OSC and leave WebSocket for LivePose integration.

Week 9 Jul 28 - Aug 3

- Final testing and debugging
- Documentation

Deliverable 2 Due Aug 3 - Prototyping Pipeline

- [Required] A working pipeline including the four main parts: camera, backends, filters, outputs.
- [Required] Evaluation of real-time performance
  - [Required] Speed and FPS with different models / cameras / compute resources
  - [Optional] Accuracy
- [Required] Detailed documentation of the components tried, instructions to run pipeline, evaluation results, and possible future improvements.

Milestone 3: Integration into LivePose

Week 10 Aug 4 - Aug 10

- Update inputs to accept RGB-D cameras and convert to point cloud

Week 11 Aug 11 - Aug 17

- Add PointNet++ inference
- [Optional] Add more models

Week 12 Aug 18 - Aug 24

- Update filters and outputs accordingly
- By this week LivePose should run end-to-end for at least one config

Week 13 Aug 25 - Sep 1

- Testing and debugging
- Documentation (could reuse from the prototype documentation)
- GSoC final evaluation

Deliverable 3 Due Sep 1 - Integration into LivePose

- [Required] At least one camera and one model integrated end-to-end into LivePose.
- [Required] Detailed documentation of the added flow.
- [Optional] Add more cameras / models / filters tried in prototyping to LivePose.

Related Work

This project requires utilizing existing technologies in point cloud acquisition, processing, and analysis, and integrating them into a comprehensive end-to-end pipeline. Key related technologies include:

Point Cloud Acquisition: Sensors and SDKs

Acquiring accurate and real-time point cloud data is foundational to this project. The following sensors and their associated SDKs are considered:

- **Intel RealSense Cameras**: These cameras capture high-resolution depth and color data. The pyrealsense2 SDK facilitates depth and color streaming, providing intrinsic and extrinsic calibration information. It also offers synthetic streams and supports recording and playback of streaming sessions.

- **Orbbec Cameras**: Orbbec's 3D cameras, such as Orbbec Femto Megas, are designed for various applications requiring depth sensing. They can be accessed through Orbbec SDK, and are also compatible with OpenNI2, a framework that provides access to depth sensors.

- **Stereolabs ZED Cameras**: These stereo cameras generate depth maps and corresponding point clouds, enabling real-time 3D perception. The ZED SDK provides tools for spatial mapping and 3D reconstruction.

Point Cloud Processing Libraries

Efficient processing of point cloud data is crucial for real-time applications. Relevant libraries include:

- **Open3D**: This library provides data structures and algorithms for 3D data processing, including point cloud visualization, registration, and segmentation. Open3D integrates with RealSense (librealsense SDK v2), allowing users to access RealSense cameras through both C++ and Python APIs without a separate SDK installation.
- **OpenCV**: A versatile library for computer vision tasks, OpenCV supports integration with 3D sensors. For instance, users can access Orbbec's 3D cameras using the OpenCV VideoCapture class without relying on proprietary SDKs, simplifying the development process.

Deep Learning Models for 3D Object Detection

Implementing robust 3D object detection models is central to the project's objectives. Relevant models include:

- **PointNet++**[1]: An extension of PointNet, PointNet++ applies PointNet recursively on a nested partitioning of the input point cloud to capture fine-grained patterns. This hierarchical approach enables the model to learn local features efficiently, making it suitable for tasks like 3D object detection and segmentation.
- **PointPillars**[2]: This model encodes point clouds organized in vertical columns (pillars) using PointNets, enabling end-to-end learning with only 2D convolutional layers. PointPillars achieves a balance between speed and accuracy, making it suitable for real-time applications.
- **CenterPoint**[3]: CenterPoint represents, detects, and tracks 3D objects as points. It first detects centers of objects using a keypoint detector and regresses to other attributes, including 3D size, orientation, and velocity. This approach simplifies 3D object tracking to greedy closest-point matching, resulting in a simple, efficient, and effective detection and tracking algorithm.

Model Inference with ONNX

ONNX (Open Neural Network Exchange) provides a standard format that allows models to be executed across different platforms and runtimes, which enables fast experimentation with pretrained models.

- **ONNX Runtime**: A cross-platform, high-performance engine for running ONNX models. It supports GPU acceleration (via CUDA) and can optionally integrate with TensorRT for improved

performance on NVIDIA hardware.

- **TensorRT Integration**: While not required for prototyping, ONNX Runtime can use TensorRT as an execution provider to further accelerate inference, with optimizations like layer fusion and mixed-precision computation.
- **Model Conversion and Deployment**: Though this project uses existing ONNX models, many models (e.g., from MMDetection3D) can be converted from PyTorch to ONNX using standard export tools. This makes it easier to extend or adapt the system in the future.

Using ONNX allows the pipeline to remain modular and framework-agnostic, and simplifies benchmarking across different models and hardware setups. This flexibility supports fast iteration during the prototyping phase and smooth integration into broader systems like LivePose.

References

[1] Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

[2] Lang, Alex H., et al. "Pointpillars: Fast encoders for object detection from point clouds." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[3] Yin, Tianwei, Xingyi Zhou, and Philipp Krahenbuhl. "Center-based 3d object detection and tracking." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

Biographical Information

I'm Eddie Liu, a first-year master's student in Computer Science at Georgia Institute of Technology, with experience in computer vision, deep learning, real-time systems, and backend development.

Before starting my master's, I worked as a software engineer, where I built scalable data pipelines and real-time dashboards, and implemented ML models using PyTorch and Scikit-learn for text analysis and trend prediction.

As an undergraduate, I conducted research on predicting urban street safety using structured data and image features. I collected metadata from OpenStreetMap and street view images via the Google Maps API, fine-tuned an image segmentation model, and trained deep neural networks to predict safety scores.

Outside of school and work, I enjoy photography, working with cameras and drones, and exploring how emerging technologies can enhance visual storytelling. I'm especially excited about this project's blend of science and art, and how streaming 3D object detection can enable immersive, interactive experiences.

I bring strong engineering skills, a passion for both science and art, and enthusiasm for building open-source, real-time, ML-powered systems. I'm confident I can deliver a reliable pipeline and contribute meaningfully to LivePose and its community.

Skills

- Required: openness to read and learn about new stuff
    - As an example, at first I was not very familiar with some topics in this proposal, like 3D point cloud formats, streaming data frameworks, and specific models (e.g., PointNet). I did my own research and learned a lot about this area. It's been an exciting and rewarding process!
- Required: willingness to ask questions when stuck
    - I asked for suggestions for reading in my introduction issue, and the mentor pointed me toward documentation on RGB-D cameras, which was super helpful to get me started.
- Preferred: some experience with programming (but not a prerequisite!)
    - I'm a CS student and have previous work experience as a software engineer
    - My GitHub

Expected size of project

350 hours

Rating of difficulty

hard