Lidar-Camera Overlay

Anthony Beca

September 2023

Contents

1 Introduction

This report summarizes the LiDAR-Camera overlay system implemented for the 2022-2023 Queen's AutoDrive Team. The system's primary goal is to correlate an [x,y,z] position for each pixel in the camera frame. Downstream this was used to find the 3D position of the centre of detected bounding boxes.

2 Context

This section will provide context on the Queen's AutoDrive system, and impose design constraints on the overlay module.

2.1 Sensor Suite

The sensor configuration mounted to the Queen's AutoDrive AV is shown in Figure 1. It consists of two Cepton P60 LiDARs on either end, a Cepton X90 LiDAR in the centre, and four Triton 3.2MP cameras (details in Figure 2.1). Skewed sensors are at a 30° angle with respect to the X90.

The LiDARs are solid-state - an array of lasers oscillates in all three axes to create the scan pattern.

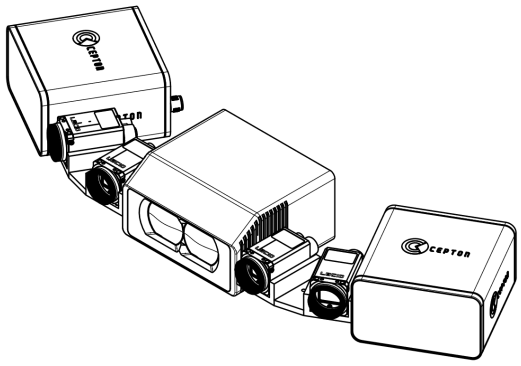

Figure 1: The sensor configuration

	X90	P60	Triton 3.2MP
Range (m)	200	200	-
HFOV (°)	90	60	-
VFOV (°)	25	22	-
Points Per Second	1M	0.3M	-
Resolution	0.13°	0.25°	2048x1536 px
FPS (Hz)	40	10-16	35.4

Table 1: Table of sensors and selected specifications

2.2 Additional Concerns

Due to time constraints, the Perception Team did not have time to manipulate point cloud data; a requirement was made to correspond camera-to-LiDAR pixel-to-point instead of a polar correspondence or range images, for example. Time Complexity of the algorithm is a concern, as approximately 1.6M LiDAR points must be projected to four images close to real-time. Additionally, hardware triggering of the sensors was not completed, and thus must be compensated for in software. Additionally, as Solid State LiDARs are new in the field, there were no existing resources online to perform a LiDAR-Camera calibration to a fixed FOV, Solid State LiDAR.

To receive any points for the perception system at competition, 3D position of detections must be reported; due to a compounding series of delays, the timeline for this project was only a month.

3 Functional Requirements

- The system must be implemented in the ROS2 Foxy framework.
- The system must compensate for a lack of hardware triggering.
- The system must overlay all point clouds onto all four images at any moment in time.
- The system must operate at a faster rate than the perception stack.
- The calculated 3D position is accurate to within 20cm (approximately the width of a traffic light).
- The system returns position for corresponding pixels in the form of [u,v,[x,y,z]].

4 System Architecture

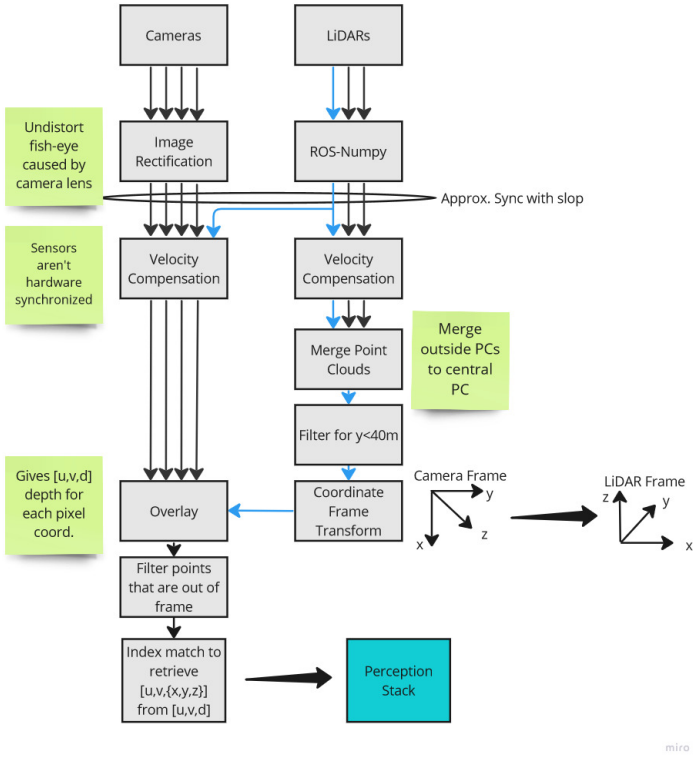

Figure 2: System Architecture; blue arrow indicates data from central LiDAR (X90)

4.1 Cameras

The LUCID arena SDK for the cameras was built to support only one camera at a time. Driver modification and networking configurations resulted in a reliable connection to all four cameras. The cameras each had to be launched in their own ROS container.

4.2 LiDARs

The Cepton LiDAR SDK allowed for multiple sensor data streams, differentiated by serial port.

4.3 Image Rectification

Before LiDAR-Camera overlay can be done, image rectification must be done using the camera intrinsic parameters to undo the distortion caused by the curvature of the lens. The ROS2 Camera Calibration package was used to obtain the intrinsic parameters and OpenCV was utilized to rectify the images. Latency was a concern, so the OpenCV C++ rectification function was wrapped in ROS2. Figure 3 demonstrates the image rectification process.

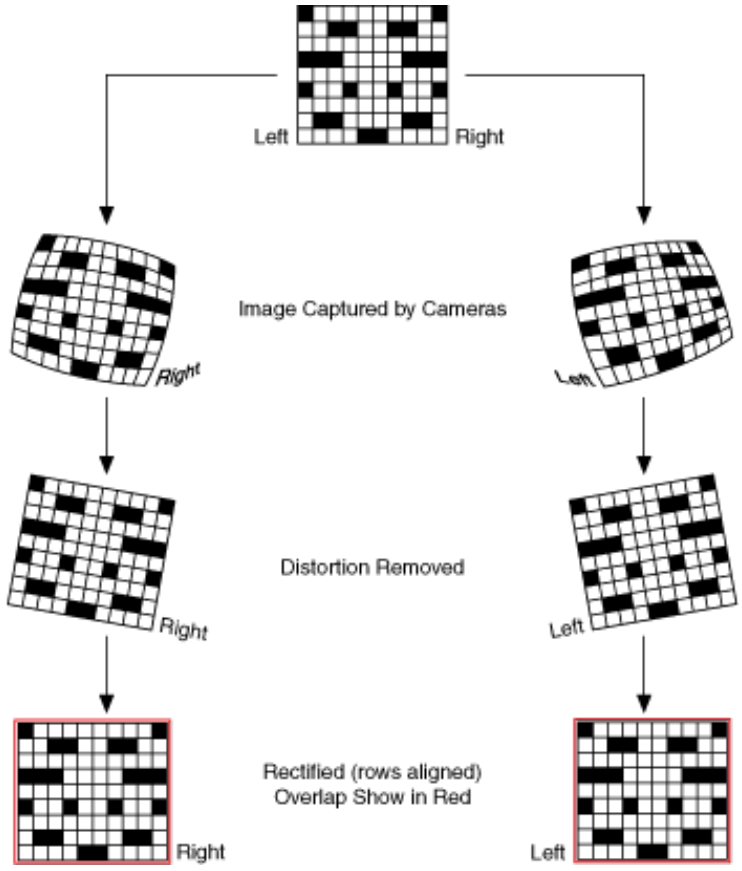

Figure 3: Demonstration of Image Rectification process.

4.4 ROS-Numpy

The Point Cloud is encoded according to the custom Cepton ROS2 message definition which has the below structure:

Before the point cloud can be manipulated, it must be converted to a Numpy array.

4.5 Merge Point Clouds

The extrinsic transformation parameters, obtained from the CAD of the sensor mount were used to rotate and translate the outer LiDAR point clouds to the central X90.

4.6 Filter for Distance

LiDAR points at a distance greater than 40m were removed, as objects further than the scoring rubric did not require that range.

4.7 Velocity Compensation

Since none of the sensors are hardware triggered (due to time constraints and hardware limitations) or have fixed frame rates, the extrinsic translations between the sensors is dependant on velocity and time. The sensors will not sample at the same instant as the central X90, therefore the data must be 'time-shifted' accordingly (See Figure 4).

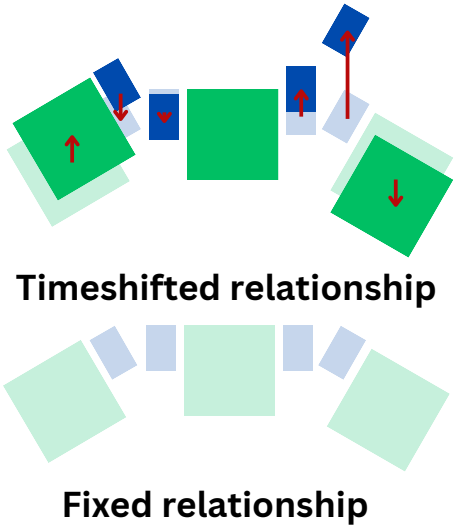

Figure 4: Illustration demonstrating why timeshifting is necessary, as sensors sample at different times, and the vehicle isn not stationary.

The ROS2 ApproximateTimeSynchronizer was used to collect the most recent data from all sensors. The timestamp from each camera and LiDAR frame, as well as the velocity from the Novatel GNSS, were used to 'time-shift' point clouds with respect to the central X90. If the timestamp of a point cloud is more recent than the timestamp of the last X90 scan, the point cloud is "shifted" backwards by dt*v, and vice versa. Each LiDAR was timeshifted to the central X90, and the combined point cloud was timeshifted to each camera.

4.8 Coordinate Frame Transform

A coordinate frame transform must be done to manipulate all data in a global coordinate system. The camera sensor uses the [x,z, -y] frame while the LiDARs use the [x,y,z], as illustrated in Figure 3.

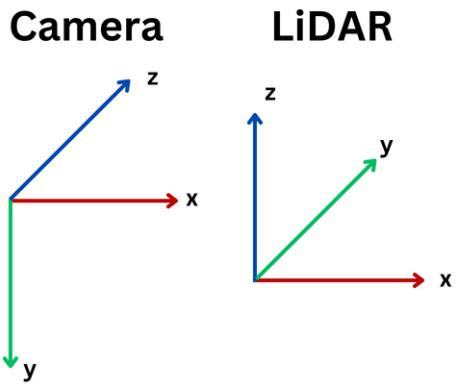

Figure 5: Camera and LiDAR respective coordinate frames.

4.9 Overlay

To correspond each 3D point to a pixel in the camera frame, the equation denoted in Figure 6 is applied. In short, to obtain a vector [u,v,1], a matrix multiplication is done for the intrinsic parameters of each camera, the rotational and translational extrinsics between the cameras and the X90, and each point in the point cloud.

This process results in many elements [u,v] that are not contained in the camera frame (ex. points from the leftmost P60 will not be in the frame of the rightmost camera), and these must be deleted to improve speed.

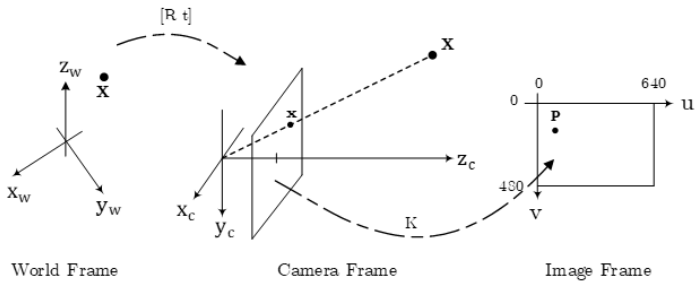

Figure 8: The different coordinate systems and mapping for a camera.

Therefore, the camera equations are:

$$\lambda p = PX = K\,[R \mid t]\,X$$

where λ : scale factor (usually in metric units)
p : point in the u-v image frame
P : projective transformation
X : 3D point in the world space
K : camera intrinsic matrix
[R | t] : rotation and translation matrices
which can be further expanded to the following form:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where u, y : x and y components of a pixel in the image frame (px)
f_x, f_y: focal length of the CCD array in the x and y direction (mm)
c_x, c_y : principal point (px)
t_x, t_y, t_z : translation in x-y-z that occurred to the 3D-world point

Figure 6: Contextual theory on calibration matrices (Source: Hany Ragab, team GRA).

4.10 Index Matching

To obtain the vector [u,v,d] from [u,v,1], an index-matching process is conducted between the overlaid points and the point cloud data. When filtering points that are out of frame for a given camera, the same points are deleted from the corresponding point cloud.

4.11 Perception Stack

To extract the position of a detected object such as a traffic light or pedestrian, the Perception team determined the centroid of a bounding box, and searched a radius of pixels to find a [u,v,[x,y,z]] match - indicating 3D position.

5 Results

Figure 7 is a visualization of the overlay output, with hue indicating depth. The system was observed to perform at 35 fps at competition, on the competition hardware. Compared to ground truth, the overlay performed well for depth (less than 10 cm of y position), which is an expected result for index matching. As evident in the figures, there is some misalignment along the x and z axes, attributable to extrinsic parameters being estimated from CAD instead of calculated or determined algorithmically. This approach was taken because of time constraints, and the lack of available tools for calibrating Solid State LiDARs. However, for all objects except traffic lights, these results are satisfactory, as there will be an overlaid point corresponding to the centroid of the bounding box.

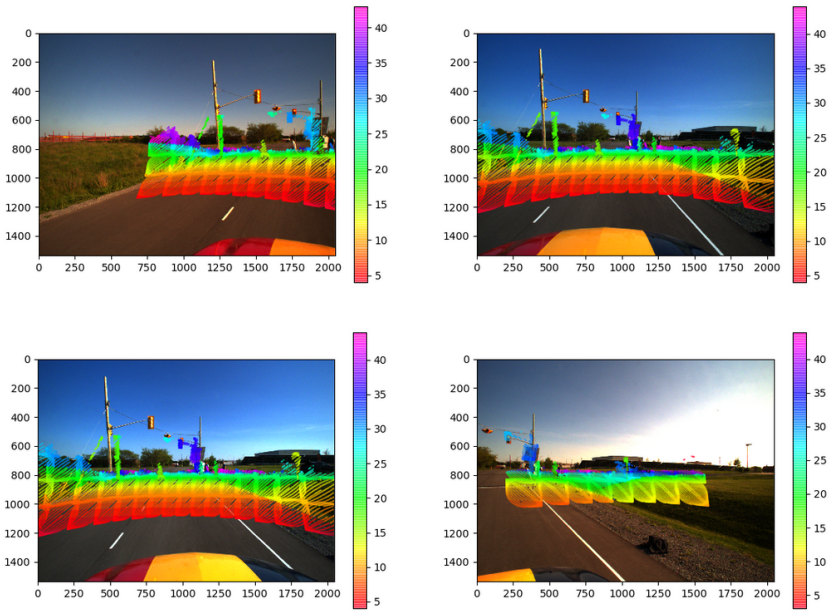

Figure 7: Visualization of overlay performance, with hue indicating depth.

A major weakness of this implementation is attributed to the lack of hardware triggering. Time shifting scans can work well to estimate the position of static objects or objects with longitudinal motion, but objects with lateral motion fall between scans and cannot be interpolated. See Figure 8 for an example of the system reporting an incorrect position of a laterally moving object.

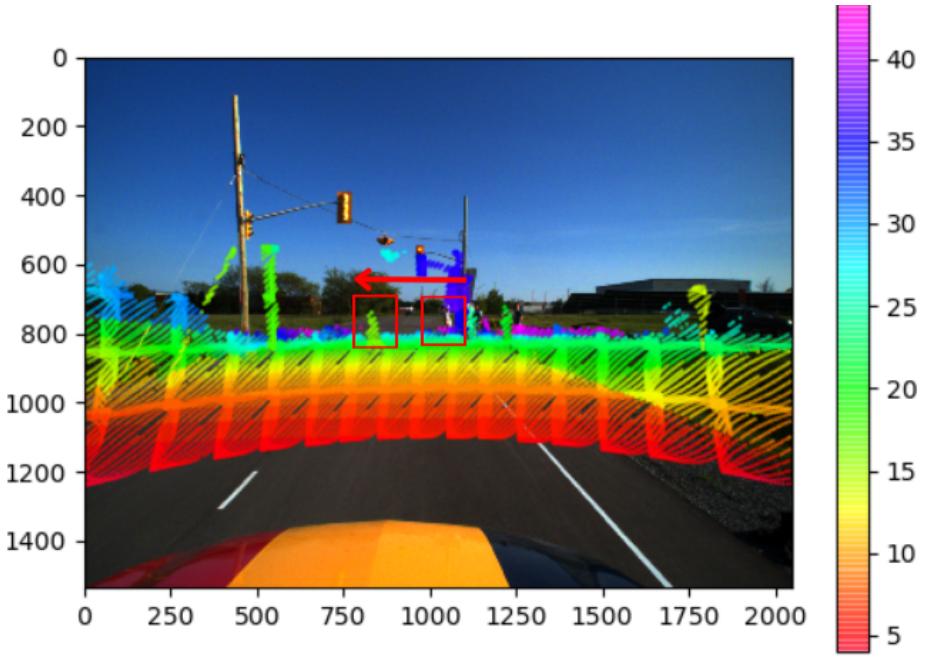

Figure 8: Example of system weakness for lateral objects; pedestrian was walking from right to left.

6 Reflection

To improve performance, there are multiple aspects I would re-implement if I were to design this system again.

1. Instead of recalculating the overlay for each scan of each sensor, it would be more efficient to find a fixed, polar relationship between the camera and LiDAR sensors, like the range image technique implemented by Waymo.

2. To ensure safety and robustness, hardware triggering must be implemented to at least synchronize the cameras, and it must be investigated why a major drop in scan frequency was observed for the LiDARs. This will improve estimates for objects travelling laterally across the scene.

3. The sensor suite should have less redundancy in HFOV, as it increases the computational load and complexity without offering much improvements in terms of accuracy. The design that was implemented was selected because of its ease of manufacturing, but it should be redesigned. Additionally, the vertical FOV must be increased to guarantee overlay of traffic lights.