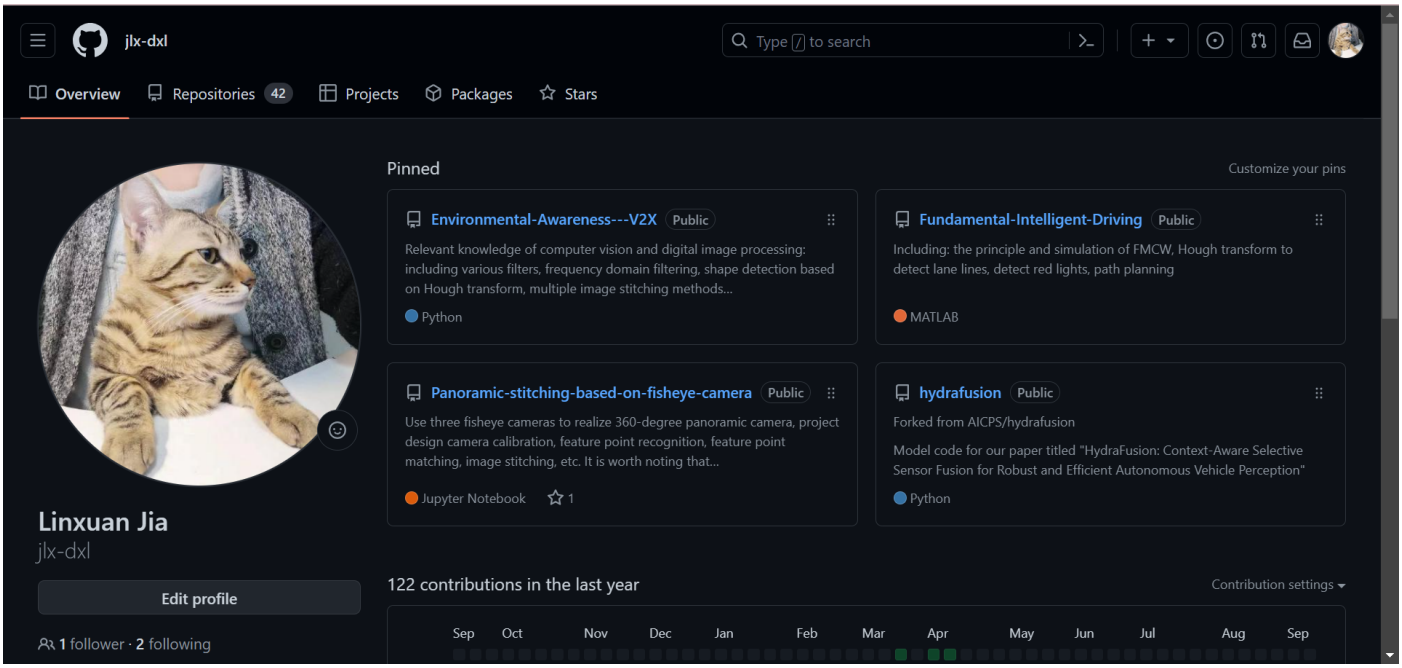
# Portfolio of Linxuan Jia

GitHub: https://github.com/jlx-dxl




## Contents

Portfolio of Linxuan Jia ..... 1
1. Design of Automatic Chain Braiding Machine ..... 2
2. ZEAL ECO-Power Energy-Saving Racing Vehicle ..... 3
3. Four-Wheel Independent Steering and Driving Vehicle (4WISD-V) with Redundant Steering System ..... 4
4. Panoramic stitching based on fisheye camera ..... 6
5. HydraFusion: Context-Aware Selective Sensor Fusion for Robust and Efficient Autonomous Vehicle Perception (Cooperated with Mohammad Al Faruque@UCI) ..... 7
6. Simulation of the whole process of autonomous driving based on Carla (Ubuntu) & Python API ..... 8
7. Interpolation and Prediction of Self-driving Lidar Point Cloud Data based on Deep Learning Neural Network ..... 10

# 1. Design of Automatic Chain Braiding Machine

Description: This is an interactive 3D model, which can automatically curling and shearing the raw steel wire into the steel rings that make up the chain in the simulation environment. It contains It includes: feeding mechanism, transmission mechanism, clamping mechanism, twisting mechanism and shearing mechanism. The difficulty with this project was getting the various agencies to work with each other in terms of timing. Through numerical analysis, I designed a series of cam curves that allowed the parts to fit together. Finally, I verified the feasibility of the mechanical work through Simulink and Inventor co-simulation. (Video Link: https://b23.tv/VCZG2dT)

**Skills: Autodesk Inventor, MATLAB Simulink**

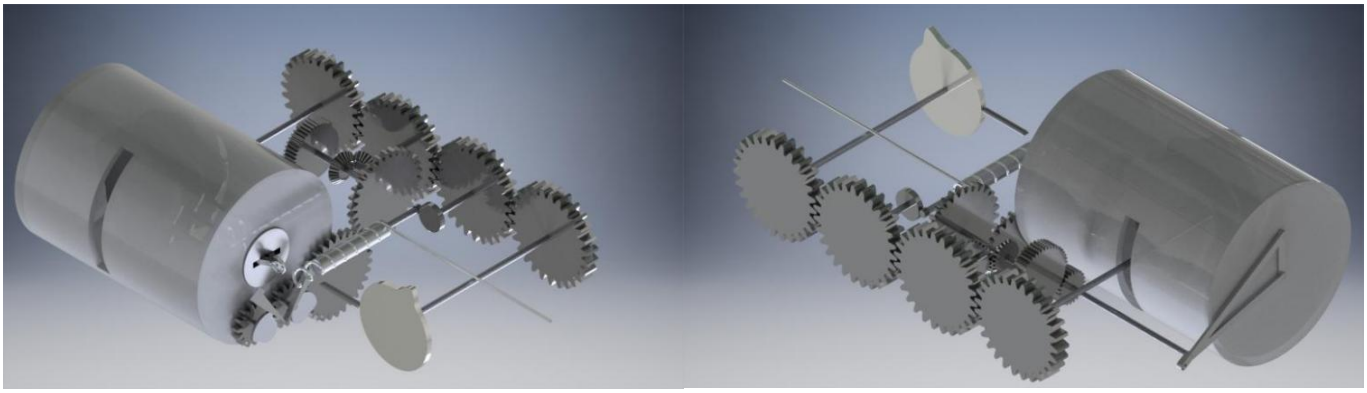


**Figure 1. Overall 3D Model (with enclosure)**

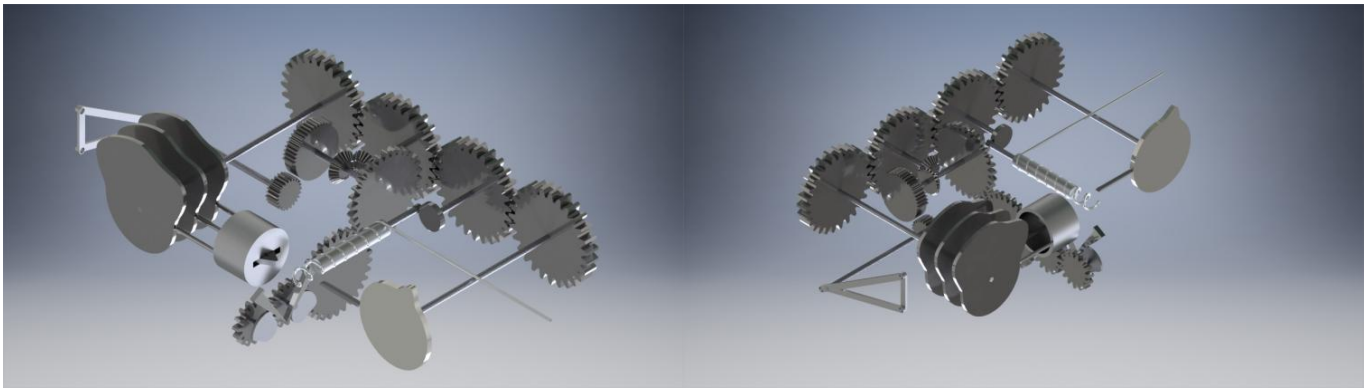


**Figure 2. Overall 3D Model (without enclosure)**

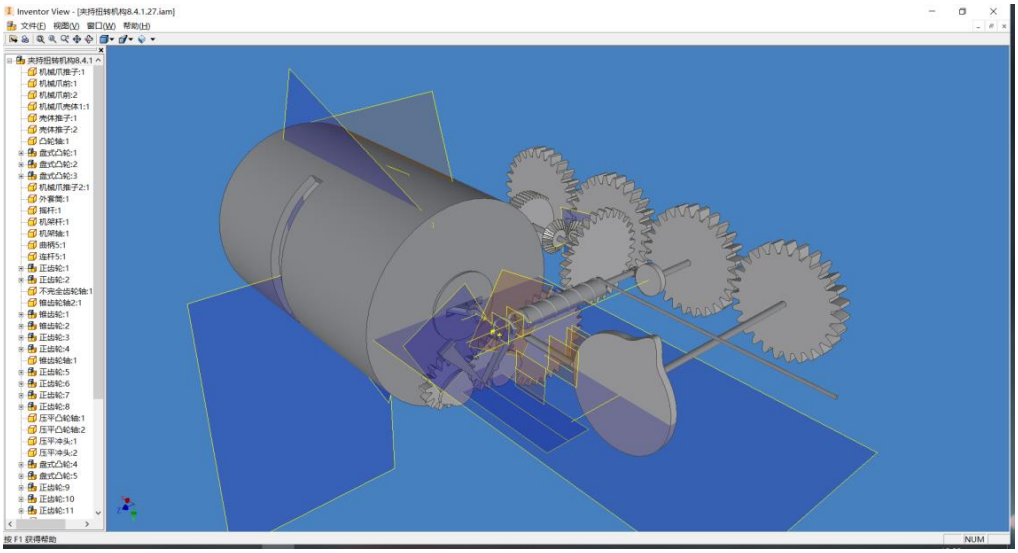


**Figure 3. Motion relationship analysis interface**

# 2. ZEAL ECO-Power Energy-Saving Racing Vehicle

Description: ZEAL ECO-Power is a team dedicated to the development of fuel-efficient racing cars, in which I was responsible for the design of the chassis and shell. The main task in the chassis design was to reduce the weight, while the main task in the shell design was to reduce the wind resistance (which was accomplished through simulated wind tunnel experiments with Hypermesh). The design work was mainly done through Inventor modeling, and the whole vehicle was fabricated.

**Skills: Autodesk Inventor, Hypermesh, Finite Element Analysis**

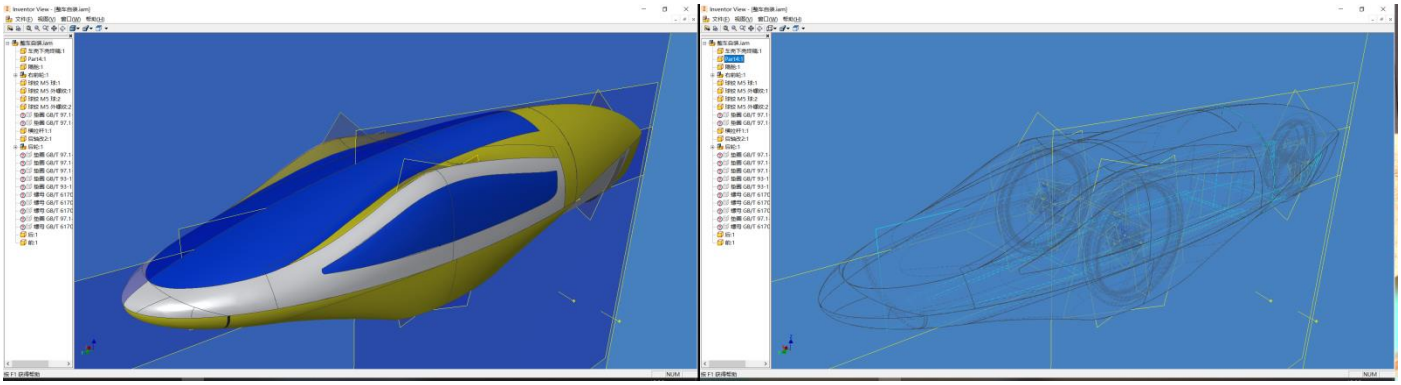


**Figure 1. Vehicle model and internal structure**

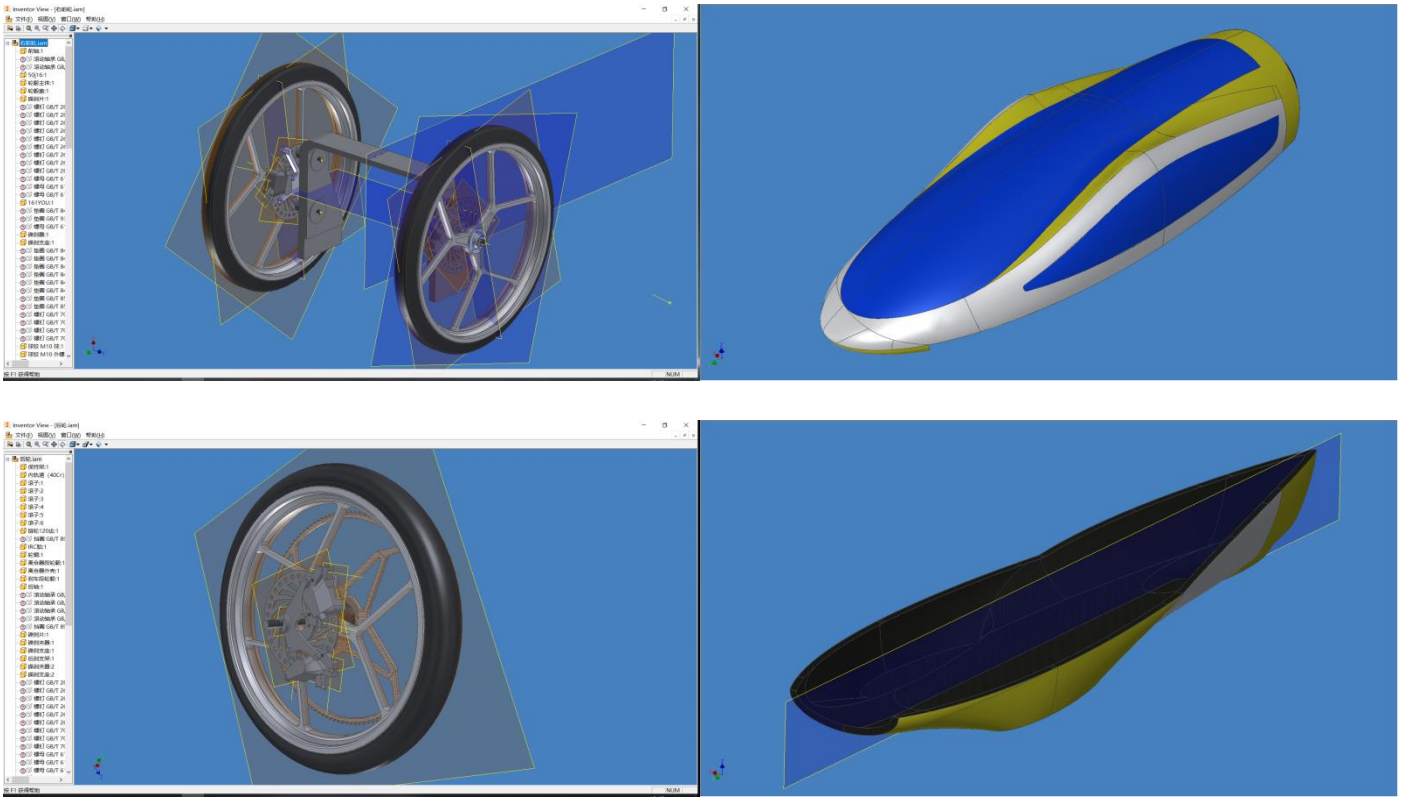


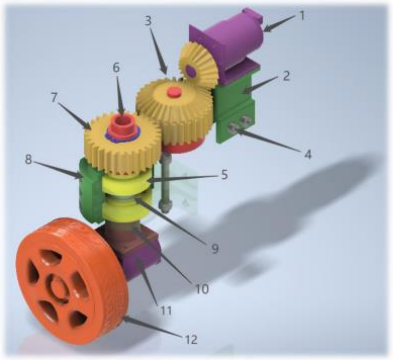
**Figure 2. Some Details**

# 3. Four-Wheel Independent Steering and Driving Vehicle (4WISD-V) with Redundant Steering System

Description: 4WISD-V is special vehicle whose wheels can independently rotate 360 degrees. Therefore, the design of the vehicle's corner module (Fig.1) is crucial in the overall vehicle design, in addition, how to make the four wheels work together is also a major difficulty. I developed a coupling control algorithm capable of multi-wheel coordination and embedded it into a designed microcontroller (STM32F2). In addition, I designed a redundant steering mechanism to cope with possible safety issues with the 4WISD-V electronically coupled steering, and developed fault-tolerant control algorithms for it. With the fabrication and assembly of the complete vehicle and the application of the remote-control module, the vehicle was able to validate its intended driving and safety features in real-world experiments. This project ultimately won **the first prize in the China National College Student Engineering Innovation Competition.**

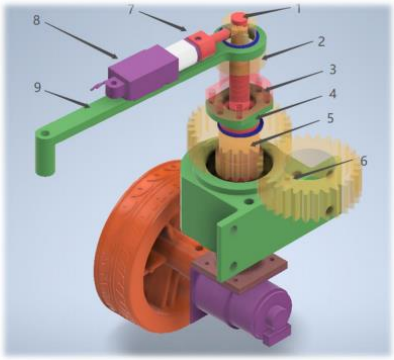
**Skills: CNC turning and milling, 3D printing, Altium Designer (PCB Design), Autodesk Inventor, STM32F2(programming in Keil)**

Corner Module based on Gear Set




1. DC deceleration motor (with photoelectric encoder); 2. Motor base; 3. Bevel gear set; 4. Bolt set; 5. Bearing; 6. Kingpin shaft; 7. Spur gear set; 8. Bearing seat; 9. Shaft ring; 10. King pin seat; 11. DC geared motor (with Hall encoder); 12 load-bearing wheels

Kingpin Locking Mechanism




1. Screw rod; 2. screw cover; 3. (screw) nut; 4. Lower flange; 5. King pin shaft; 6. Orthodontic set; 7. Cone head pin; 8. DC push rod; 9. Steering arm;

**Figure 1. Detailed diagram of the corner module**

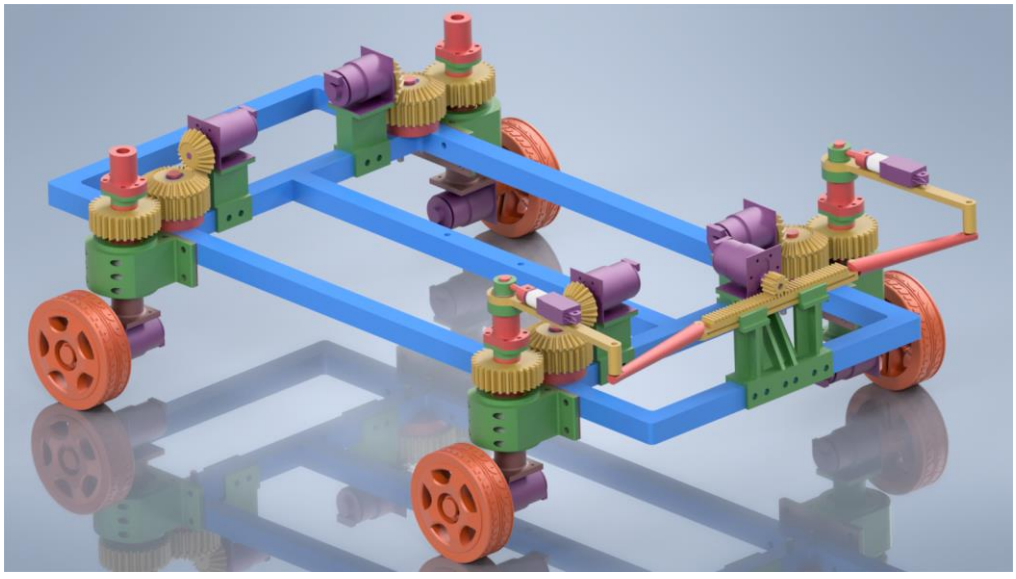


**Figure 2. 3D model of the 4WISD-V**

**Figure 3. Sample of the 4WISD-V**

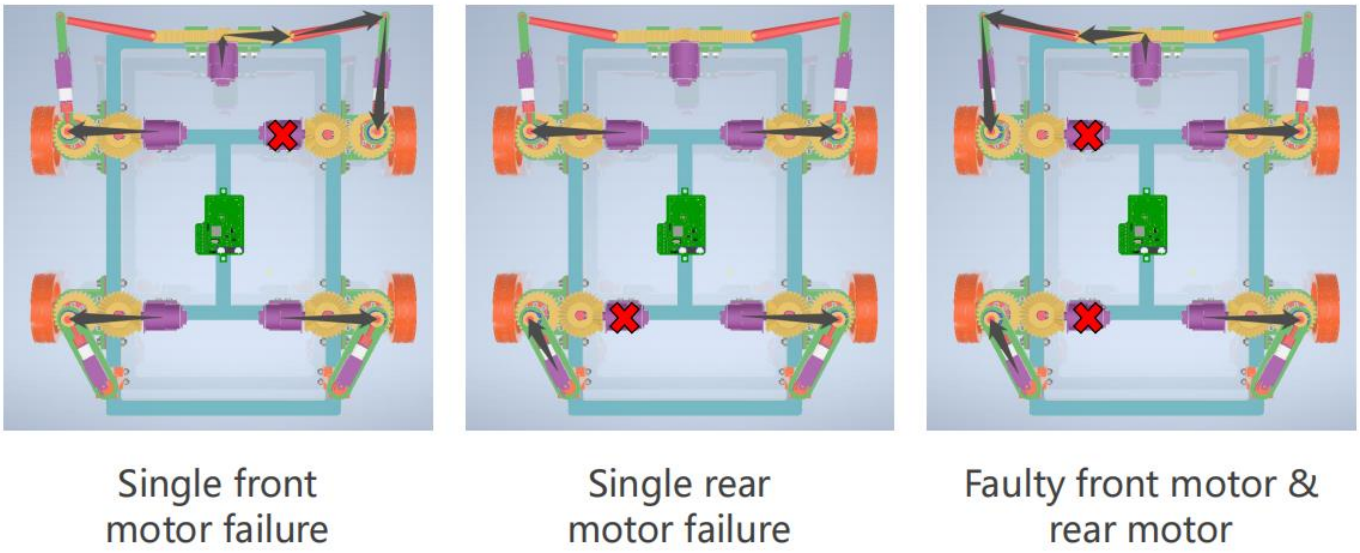



**Figure 4. Error Adaptive Control Scheme**

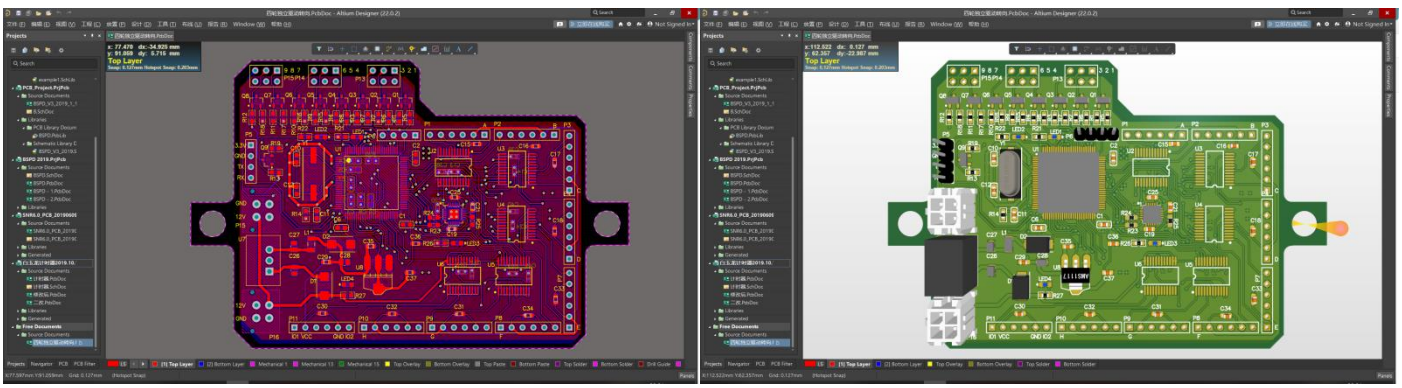


**Figure 5. Circuit diagram of the control chip**

# 4. Panoramic stitching based on fisheye camera

Description: This project is to build an in-vehicle surround view panoramic camera, by stitching and fusing images obtained from multiple fisheye cameras in a very short period of time to obtain a panoramic video stream with a 360-degree viewing angle. The techniques involved are, IO communication, SIFT feature point detection, image stitching and fusion.

**Skills: Python, ROS, OpenCV, Camera Calibration**

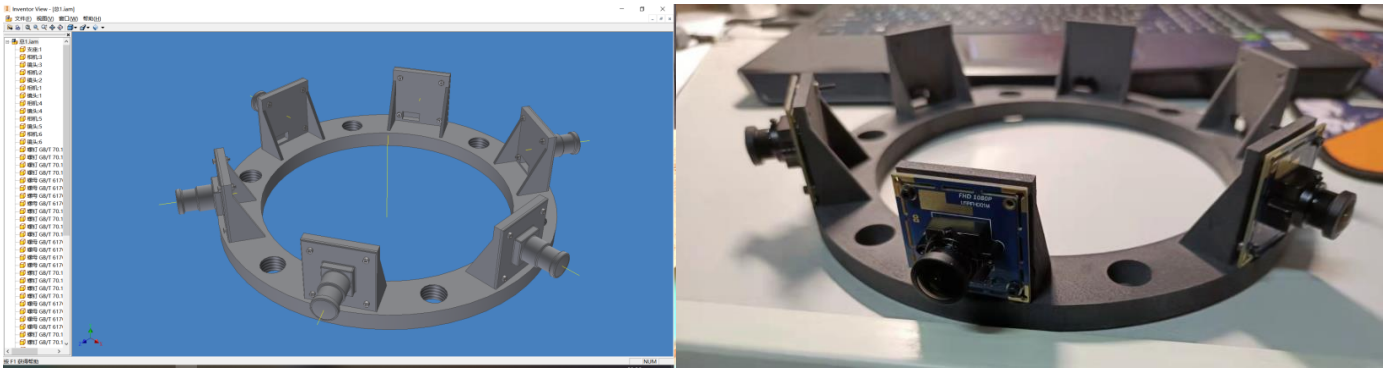


**Figure 1. The 3D model and the physical drawing of hardware devices**

**Figure 2. Fisheye camera calibration and undistortion**

**Figure 3. The effect of image stitching**

**Figure 4. Implementation of 360-degree panoramic camera**

## 5. HydraFusion: Context-Aware Selective Sensor Fusion for Robust and Efficient Autonomous Vehicle Perception (Cooperated with Mohammad Al Faruque@UCI)

Description: It is well known that each sensor has different sensing capabilities under different weather and environmental conditions, and this project aims to train multi-sensor fusion strategies based on deep learning for different weather conditions to improve the multi-sensor fusion performance as well as the accuracy of the most important downstream application: vehicle detect.

**Skills: Camera, Lidar, Radar, Deep learning, PyTorch**

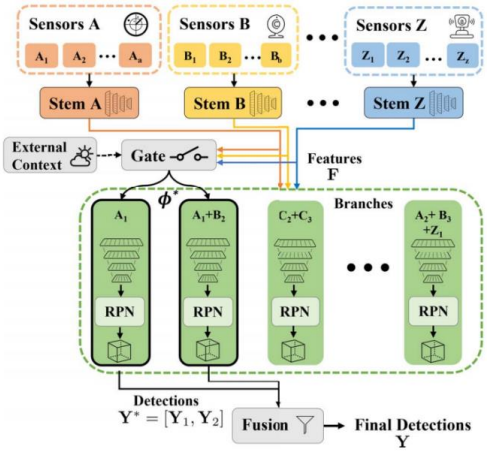



**Figure 1. HydraFusion Architecture**

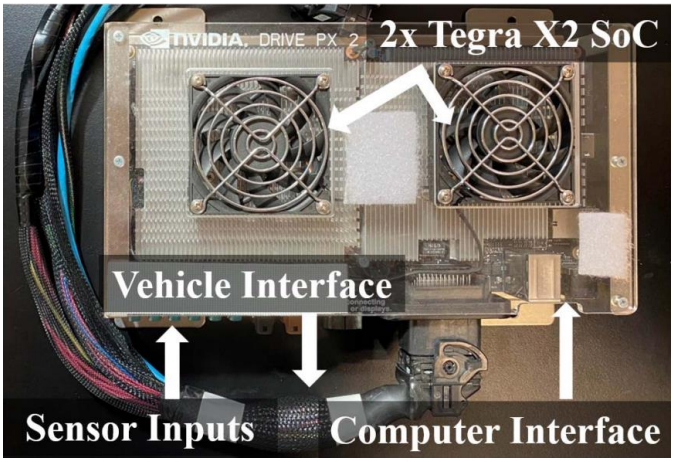



**Figure 2. Nvidia Drive PX2 Testbed**

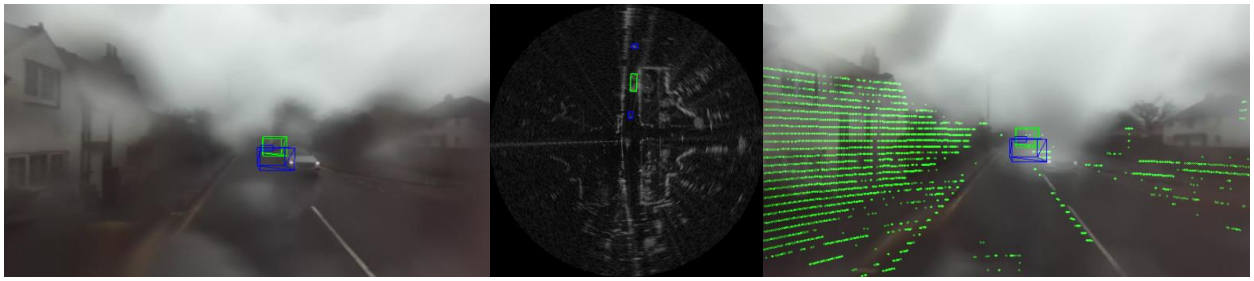


**Figure 2. Vehicle detection results after sensor fusion (rainy)**

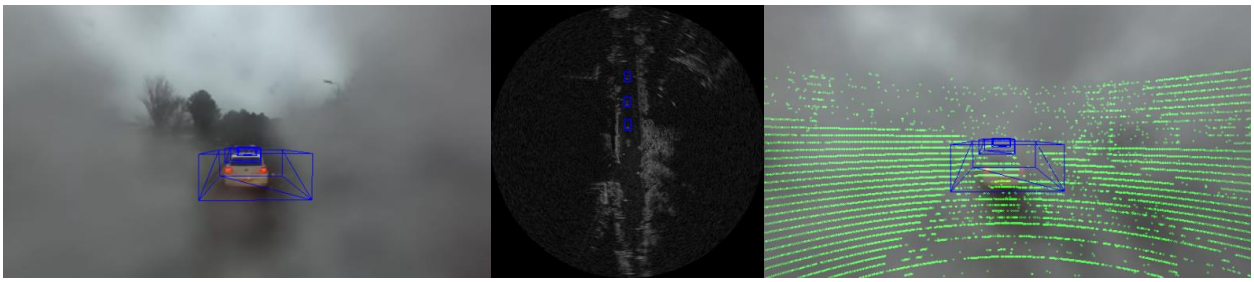


**Figure 3. Vehicle detection results after sensor fusion (snowy)**

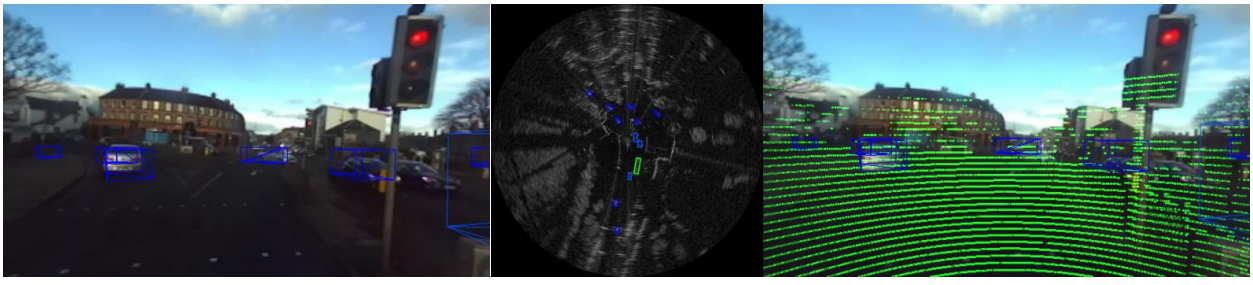
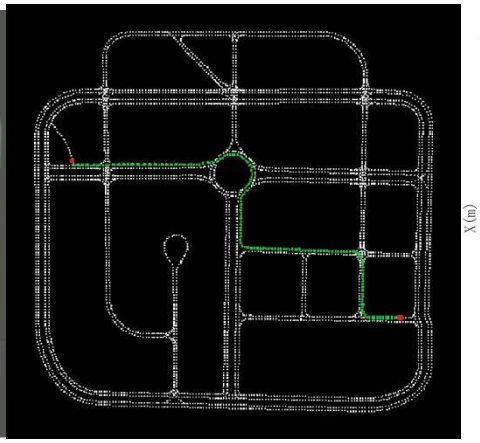


**Figure 3. Vehicle detection results after sensor fusion (sunny)**

# 6. Simulation of the whole process of autonomous driving based on Carla (Ubuntu) & Python API

Description: This project aims to realize a complete perception-planning-decision-control autonomous driving pipeline in a simulation environment. The main algorithms used in it include: (for perception) **YOLO v5**(car and pedestrian detection), **LaneNet** (lane detection); (for planning) **Fast RRT** (path planning), **Hybrid A* & State Lattice Planning** (motion planning); (for decision) **Decision Tree** (Behavioral Gaming at Traffic Intersections); (for control) **PID** (longitude control), **LQR** (lateral control)
**Skills: Algorithms above, Linux (Ubuntu), Python, DL(PyTorch)**



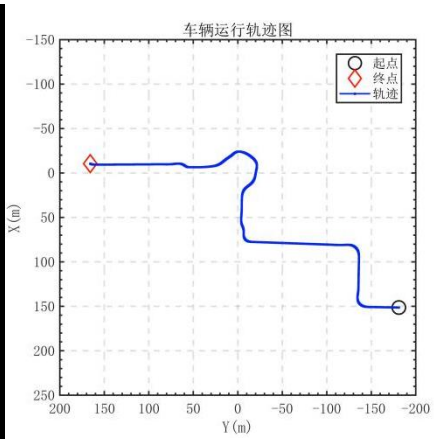
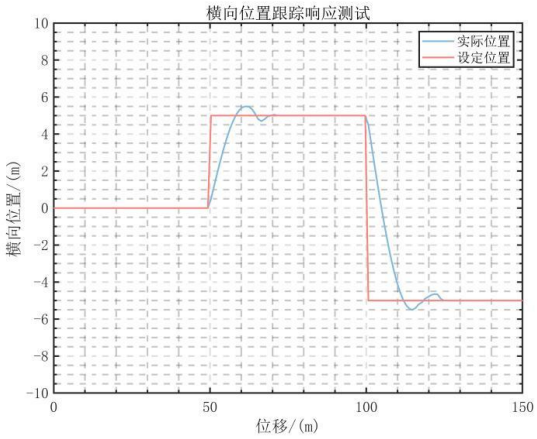



**Figure 1. Map of Town05 Figure 2. Path planning on the occupancy grid map(A*) Figure 3. actual traveled route**



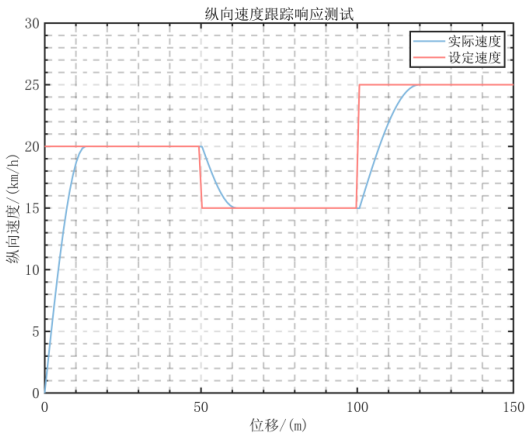




**Figure 4. Results of PID-based lateral position tracking and longitudinal velocity tracking**

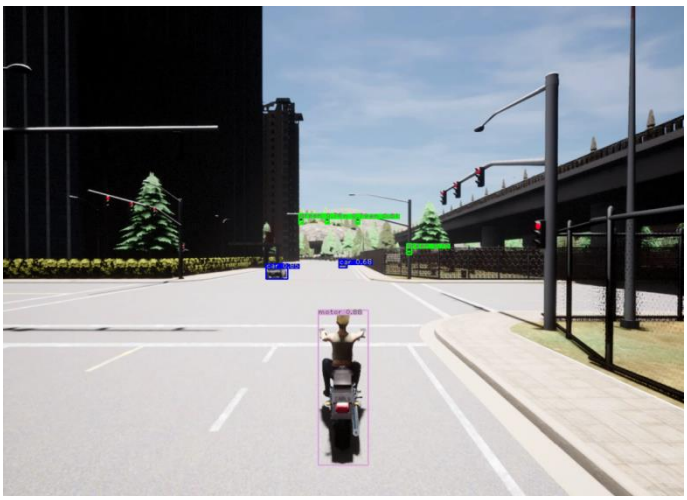



**Figure 5. Results of vehicle and traffic light detection**

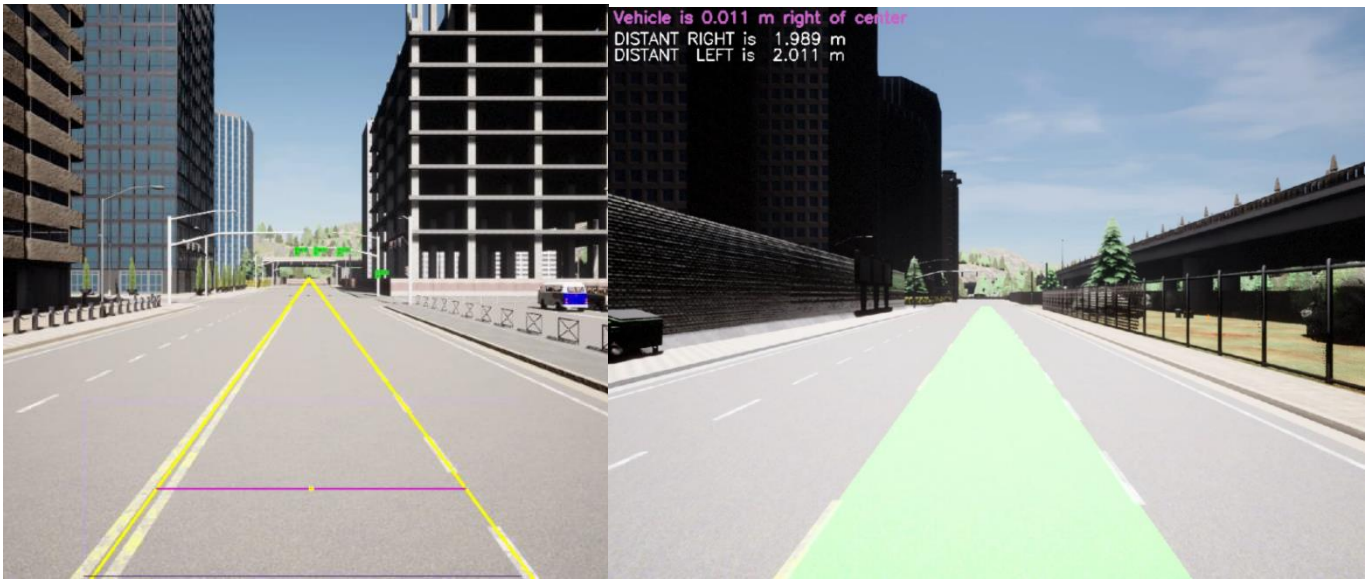



**Figure 6. Lane detection (Left: Hough Transform; Right: Gradient histogram in HSL color space)**

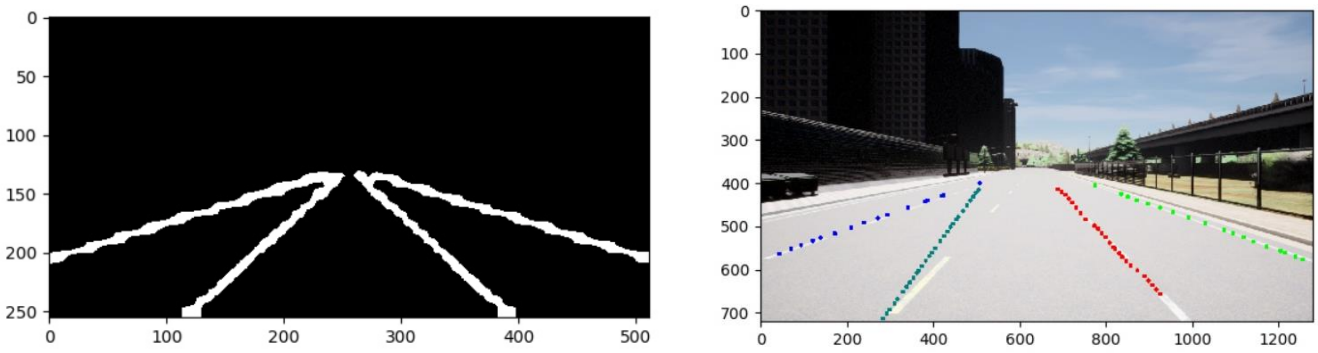



**Figure 7. Lane detection (LaneNet)**

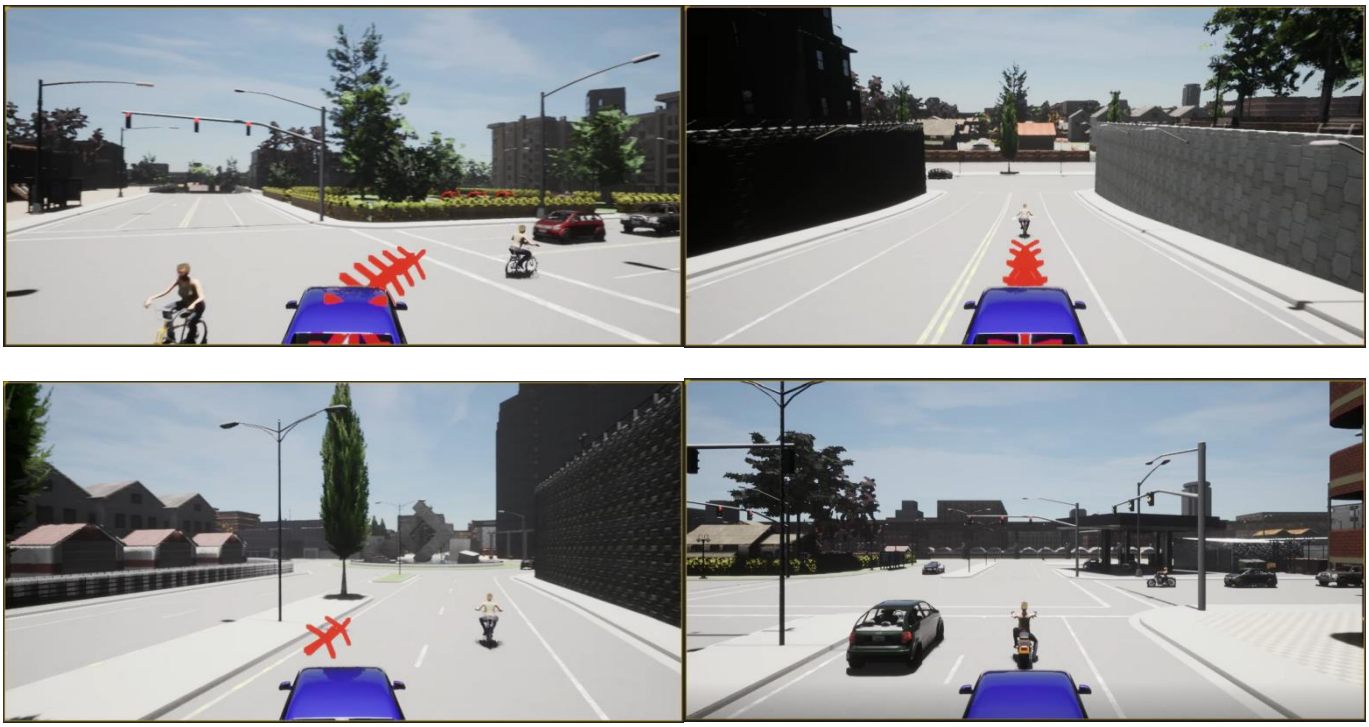


**Figure 8. Autonomous Driving Simulation**

**(Upper Left: Steering; Upper Right: Following; Bottom Left: Lane change; Bottom Right: Responding to traffic lights)**

# 7. Interpolation and Prediction of Self-driving Lidar Point Cloud Data based on Deep Learning Neural Network

Description: Compared to the high frame rates of other sensors used in vehicles, LIDAR has a low frame rate, averaging around 20Hz, which adversely affects tasks such as multi-sensor fusion and target tracking. Although there have been many methods for video frame interpolation, they do not migrate efficiently due to the disordered nature of the point cloud data. The aim of this work is to develop a method for frame interpolation of LiDAR point cloud data through a deep learning approach. This work successfully improved the interpolation accuracy by about 20% over current SOTA methods.
Skills: Python, PyTorch, Python PCL, Open3D

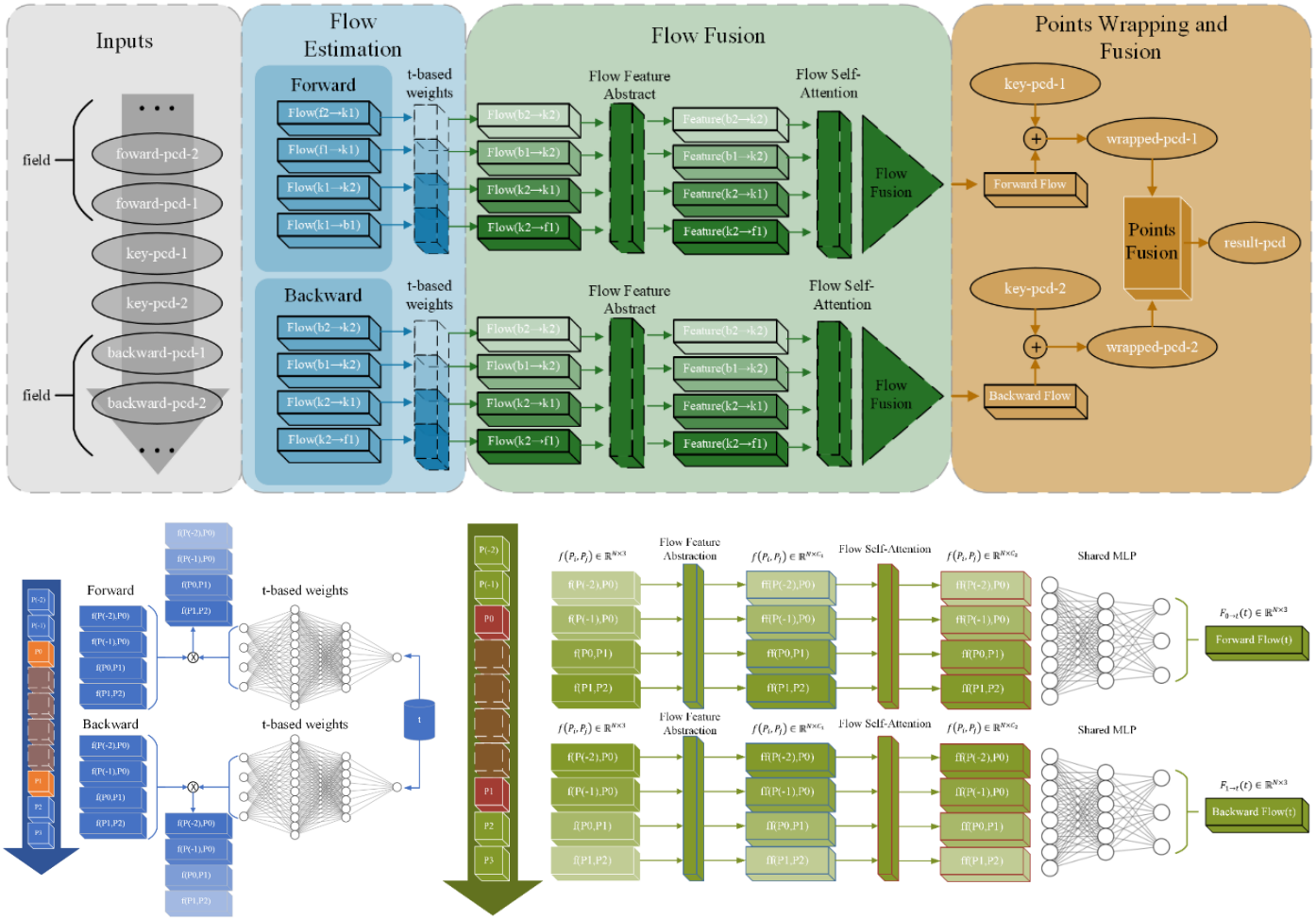
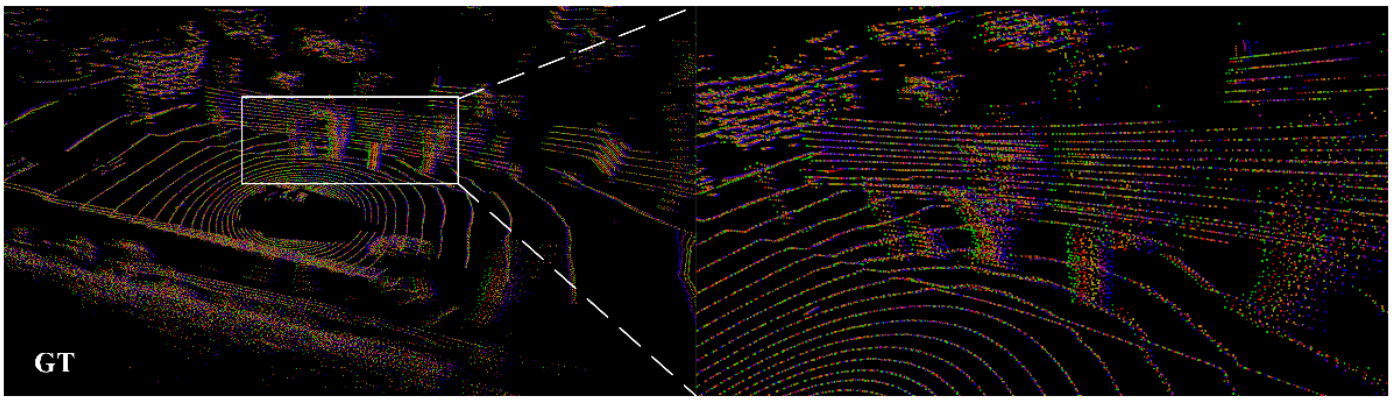



**Figure 1. Structure of the DNN**



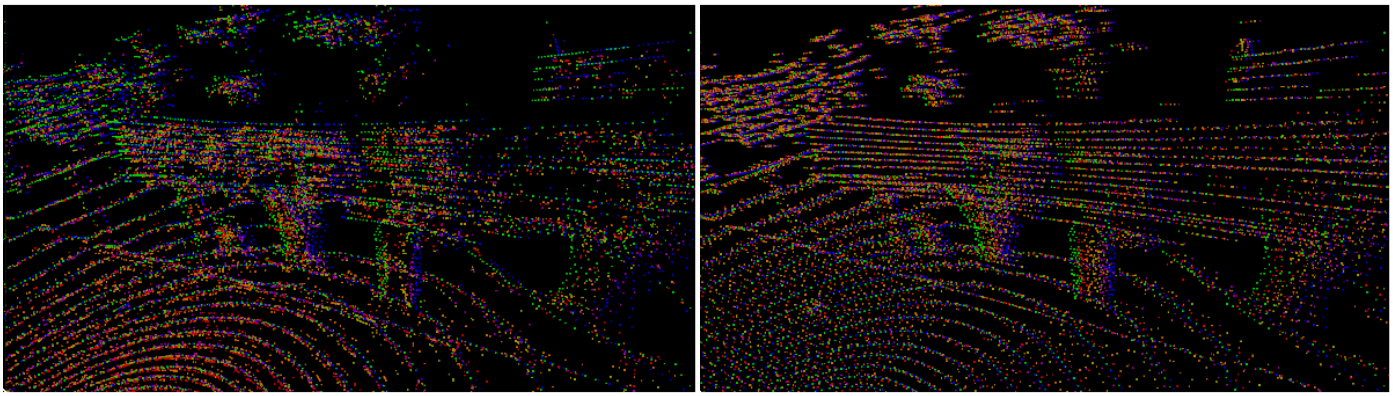
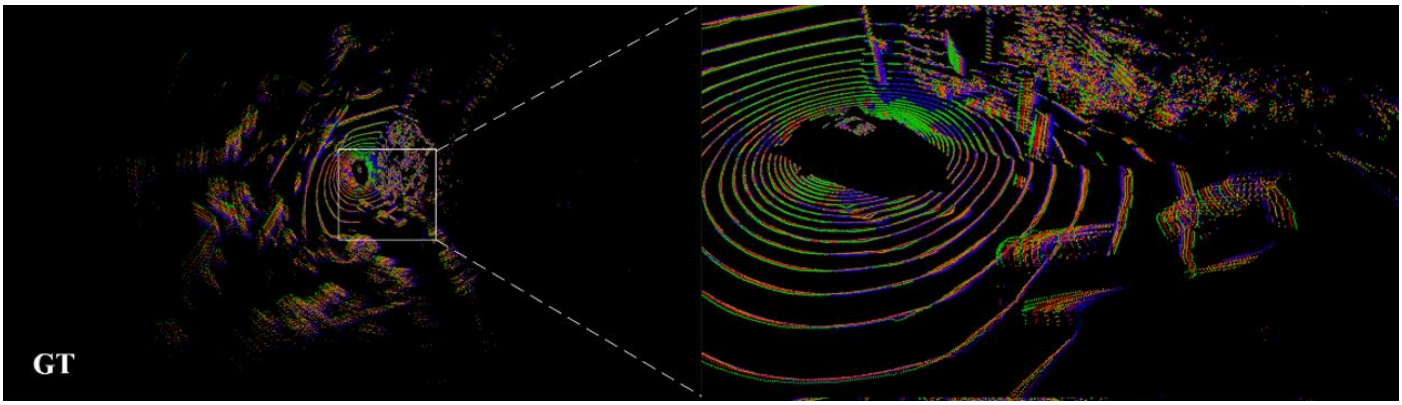


**Figure 2. Performance (Left: SOTA method; Right: our method)**



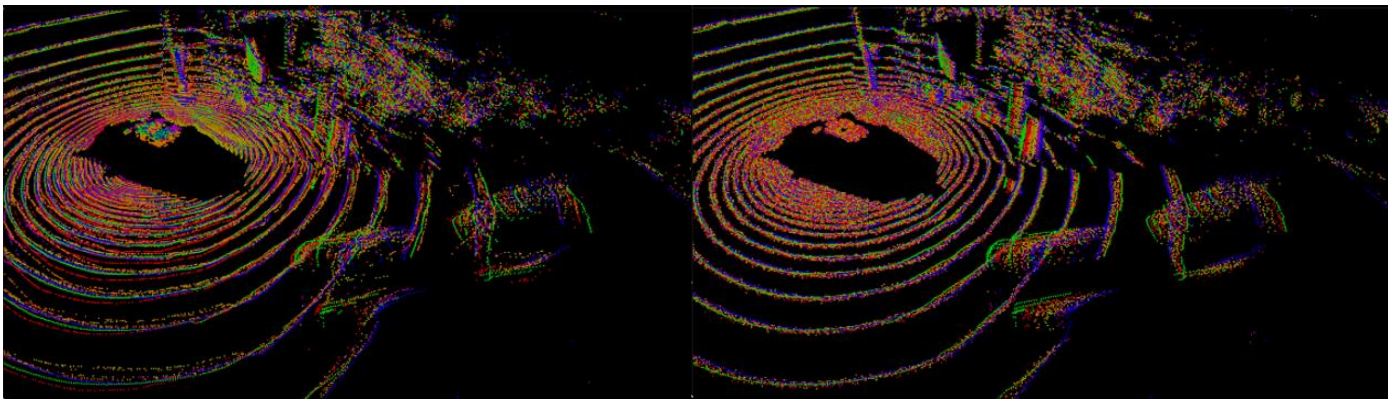



**Figure 3. Performance (Left: SOTA method; Right: our method)**

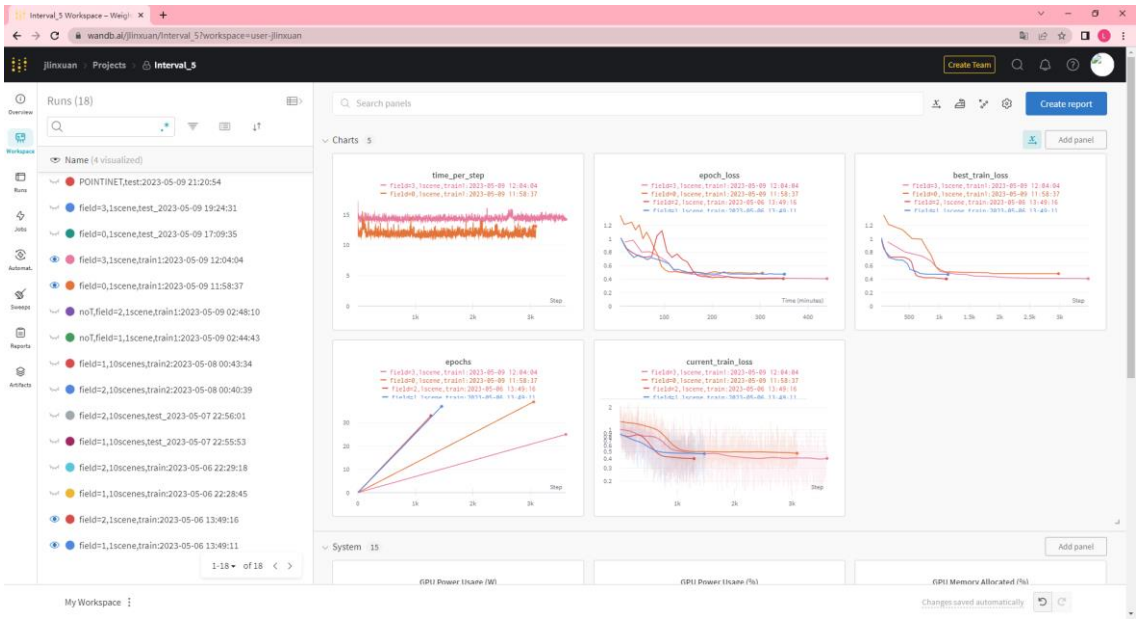



**Figure 4. Training Records (WandB)**