2024 IEEE/ACM Ninth International Conference on Internet-of-Things Design and Implementation (IoTDI) | 979-8-3503-7025-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/IoTDI61053.2024.00010

Cooperative Infrastructure Perception

Fawad Ahmad*[1], Christina Suyong Shin*[2], Weiwu Pang*[2], Branden Leong[2], Pradipta Ghosh[3], and Ramesh Govindan[2]

[1]Rochester Institute of Technology, Rochester, NY, USA
[2]University of Southern California, Los Angeles, CA, USA
[3]Meta, Menlo Park, CA, USA
fawad@cs.rit.edu, {cshin956, weiwupan, branden, ramesh}@usc.edu, iampradipta@meta.com

***Abstract*—Recent works have considered two qualitatively different approaches to overcome line-of-sight limitations of 3D sensors used for perception: cooperative perception and infrastructure-augmented perception. In this paper, motivated by increasing deployments of infrastructure LiDARs, we explore a third approach – *cooperative infrastructure perception*. This approach generates perception outputs by fusing outputs of multiple infrastructure sensors, but, to be useful, must do so quickly and accurately. We describe the design, implementation and evaluation of Cooperative Infrastructure Perception (CIP), which uses a combination of novel algorithms and systems optimizations. It produces perception outputs within 100 ms using modest computing resources and with accuracy comparable to the state-of-the-art. CIP, when used to augment vehicle perception, can improve safety. When used in conjunction with offloaded planning, CIP can increase traffic throughput at intersections.**

***Index Terms*—Cooperative Perception, Infrastructure-assisted Perception, Autonomous Vehicle Systems**

I. Introduction

Machine perception extracts higher-level representations of a scene from low-level sensor signals in real-time. Perception is essential for autonomy and is now a crucial component of every autonomous driving stack [1], [2]. The perception component of an autonomous driving stack extracts bounding boxes and tracks of dynamic objects in a scene such as vehicles, pedestrians and bicyclists. It may also extract compact representations for static scene elements such as lane markers and drivable space.

Perception has long suffered from sensor range and line-of-sight limitations. For example, a LiDAR on the vehicle A in Fig. 1 may not have enough range to see the bicyclist behind the vehicle B. Even if it did, the LiDAR's view would be occluded by the vehicle B. To address this, prior work has considered two qualitatively different approaches. In *cooperative perception*, vehicles share sensor or perception outputs between themselves [3] to effectively extend visual range and address line-of-sight limitations. For example, the vehicle A (Fig. 1) could "see" the bicyclist using vehicle B's perception outputs. In *infrastructure-assisted perception* [4], a vehicle augments its own perception using sensors in the infrastructure [5]. In Fig. 1, the vehicle A could "see" the occluded bicyclist using the LiDAR at the top of the figure.

In this work, we consider a complementary capability, *cooperative infrastructure perception* (or CIP). This produces bounding boxes and tracks of dynamic objects in a scene, such as vehicles and pedestrians, by combining and processing outputs of multiple infrastructure sensors. Fig. 1 shows an example in which CIP can use four LiDARs placed at an intersection to cooperatively detect and track vehicles, pedestrians and bicyclists. Compared to cooperative perception, CIP's LiDARs, when mounted well above vehicle height, will likely have a better view of objects in the scene. For example, the LiDAR on the left might be able to view part of the bicyclist behind the vehicle B, while the vehicle A's LiDAR may not. Compared to infrastructure-assisted perception, CIP's LiDARs can collectively obtain more complete views of an object and so can estimate better bounding boxes. For example, each LiDAR adjacent to the vehicle B can only view half of the vehicle; together, they can view the entire car.

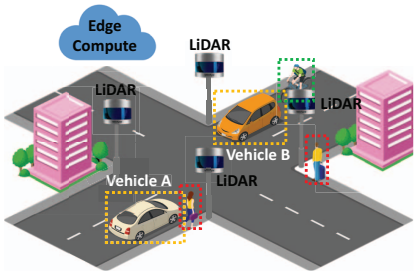


Figure 1: CIP deployment at an intersection with multiple LiDARs and nearby edge compute.

CIP enables several novel capabilities:

Perception Augmentation. CIP can deliver bounding boxes and tracks of all traffic participants to all vehicles. An autonomous vehicle can augment its own perception with these and plan better paths, thereby increasing overall safety.

Planning Offload. CIP's outputs can be used to plan trajectories for all vehicles on edge compute (Fig. 1), and deliver each vehicle's trajectory wirelessly. Offloading planning can be useful in restricted settings such as ports, parking lots, factories, and so on. Indeed, industry is developing such a capability to autonomously guiding a car without on-board perception and planning into a parking spot [6].

Pedestrian Situational Awareness. CIP's outputs can be processed to deliver situational awareness to pedestrians such as an audio cue to a visually-impaired pedestrian or rendering by a real-time outdoor augmented reality system.

* Equal contribution to this work.

979-8-3503-7025-6/24/$31.00 ©2024 IEEE
DOI 10.1109/IoTDI61053.2024.00010

More generally, CIP can be deployed not just at intersections, but in plazas, shopping malls, college and enterprise campuses and tourist attractions, and can be used to guide pedestrians, bicyclists and vehicles in various ways.

Three technology trends enable CIP. LiDARs are becoming cheaper, especially with the development of solid-state LiDARs [7]. Large cloud providers are rolling out edge compute deployments [8]. Finally, cellular carriers have made substantial investments in 5G deployments [9].

Given these trends, we expect that CIP software can be architected as a software pipeline running on commodity edge compute (Fig. 1). CIP will process and combine LiDAR outputs to produce objects and tracks, then deliver them wirelessly to traffic participants. The use cases presented above motivate two challenging requirements that CIP must meet.

- Autonomous vehicles must perceive the world and make driving decisions at a frequency of 10 Hz with a tail latency less than 100 ms [10]. To be applicable to autonomous driving, CIP must also adhere to the same latency constraints. LiDARs generate data at 10 frames per second. At 30 MB per frame, with four LiDARs at an intersection, this translates to CIP having to process data at a raw rate of nearly 10 Gbps. With commodity compute, this is not straightforward.
- CIP generates a scene description that consists of dynamic objects along with their positions, bounding boxes, heading vectors and motion vectors. The accuracy of these must match or exceed the state-of-the-art computer vision algorithms.

It is not immediately obvious that CIP can meet these requirements; for example, the most accurate 2D and 3D object detectors on a popular autonomous driving benchmark [11] incur a processing latency of 60-300 ms.

To address these challenges, our work makes the following contributions:

- To fuse 3D frames from multiple LiDARs, CIP introduces a novel alignment algorithm whose accuracy is significantly higher than prior work (§II-A).
- With an accurate 3D fused view, CIP introduces fast and cheap implementations of algorithms for dynamic object extraction, tracking, and speed estimation. These are centered around a single bounding box abstraction which CIP computes early on in the pipeline (§II-B). This design choice is crucial for ensuring speed without sacrificing accuracy.
- Cheap algorithms for heading estimation are inaccurate, so CIP develops a more accurate GPU-offload heading estimator to meet the latency constraint (§II-C).
- Its outputs can be used to augment vehicle perception, or enable offloaded planning, capabilities that can increase vehicular safety and throughput.

Using real-world datasets and simulations, we show that CIP can generate perception outputs with a 99th percentile latency of less than 90 ms in scenes with 30-50 vehicles and pedestrians, just using a 16 core desktop with a single GPU. Its object extraction and tracking accuracy compare well with the state-of-the-art. When used to augment perception, it can ensure safety in 3×-5× more scenarios than standard autonomous driving. When used with offloaded planning, it can reduce traffic wait time by up to 5×.

II. CIP Design

In this section, we describe CIP's design, beginning with an overview of its approach. We do this using data from our deployment of four LiDARs at the four corners of a busy intersection of a major metropolitan area as shown in Fig. 2(a).[1]

Inputs and Outputs. The input to CIP is a continuous sequence of LiDAR frames from each LiDAR in a set of *overlapping* LiDARs deployed roadside. The output of perception is a compact *abstract scene description*: a list of bounding boxes of moving objects together with their motion vectors (Fig. 2(e)).

Approach. To address the challenges described in §I, CIP uses a three stage pipeline, each with three sub-stages (Fig. 3). *Fusion* combines multiple LiDAR views into fused frames (Fig. 2(c)), and then subtracts the static background to reduce data for subsequent processing (Fig. 2(d)). *Participant extraction* identifies traffic participants by clustering and, estimates a tight 3D bounding box around each object. *Tracking* associates objects across frames, and estimates their heading and motion vectors (Fig. 2(e)).

This design achieves CIP's goals using three ideas:

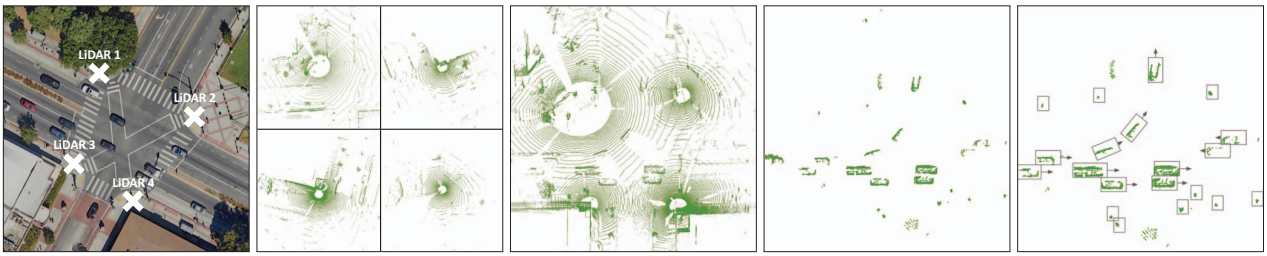
- CIP exploits the fact that LiDARs are static to cheaply *fuse* point clouds from multiple overlapping LiDARs into a single *fused frame*. Such a frame may have more complete representations of objects than individual LiDAR frames. Fig. 2(b) shows the view from each of the four LiDARs; in many of them, only parts of a vehicle are visible. Fig. 2(c) shows a *fused frame* which combines the four LiDAR frames into one; in this, all vehicles are completely visible.[2]
- CIP builds most algorithms around a single abstraction, the 3D bounding box of an object (Fig. 2(e)). Its tracking, speed estimation, and motion vector estimation rely on the observation that the centroid of the bounding box is a convenient consistent point for estimating these quantities, especially when fused LiDAR frames provide comprehensive views of an object.
- CIP uses, when possible, cheap algorithms rather than expensive deep neural networks (DNNs). Only when higher accuracy is required does CIP resort to expensive algorithms, but employs hardware acceleration to meet the latency budget; its use of a specialized heading vector estimation is an example.

To appreciate the novelty of this approach, consider Table I which compares CIP's design to that of open-source autonomous driving perception designs [1], [2]:

- These approaches rely on a pre-built 3D map (called HD map), and localize the ego-vehicle (the one on which the autonomous driving stack runs) by matching its LiDAR scans against the HD map. In contrast, CIP does not require a map for positioning: all vehicle positions can be directly estimated from the fused frame.

[1] Practical deployments of roadside LiDARs need to consider coverage redundancy and other placement geometry issues, which are beyond the scope of this paper.

[2] Not immediately obvious from the figure, but a LiDAR frame, or a fused frame is a 3D object, of which Fig. 2(c) is a 2D projection.



(a) (b) (c) (d) (e)

FIGURE 2: CIP deployment at a busy intersection with heavy vehicular and pedestrian traffic in a large metropolitan city. (a) A top down view of the intersection (taken from Google Maps [12]). We mounted four LiDARs near each of traffic light poles situated at the four corners of the intersection. (b) An individual frame from each one of the four LiDARs. (c) A fused frame. (d) Point clouds of traffic participants (dynamic objects) at the intersection. (e) Bounding boxes and motion vectors for traffic participants, calculated over successive frames.

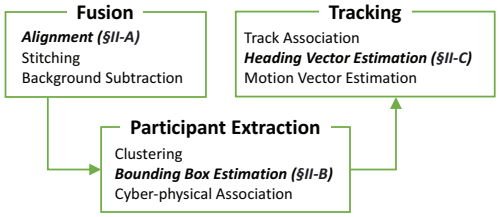


FIGURE 3: Perception stages. Bold sub-stages described in detail.

	CIP	**AV Perception** [1], [2]
Mapping	***None***	Pre-built HD map
Localization	***Live Fused Point Cloud*** (§II-A)	LiDAR scan matching, Fusion with GPS-RTK or IMU
Object Detection	***3D Object Detection*** (§II-B)	Obstacle detection in 2D BEV projection, 2D object detection in camera, projected to 3D
Tracking/Motion Estimation	Kalman filter with matching, ***Heading Estimation*** (§II-C)	Kalman filter with matching

TABLE I: CIP and autonomous vehicle perception.

- Autonomous driving stacks use different ways to identify traffic participants. They extract obstacles from a 2D birds-eye-view or use a 2D object detector on images, then back-project these to the 3D LiDAR view. CIP, on the other hand, directly extracts the 3D point cloud associated with each participant. It does this cheaply using background subtraction because its LiDARs are relatively static.
- Autonomous driving stacks use Kalman filters to estimate motion properties[3] (*e.g.,* speed and heading) of other vehicles. For heading, CIP uses a more sophisticated algorithm to ensure higher tail accuracy.

Below, we describe parts of CIP's novel perception relative to autonomous driving stacks (bold text in Table I).

A. Accurate Alignment for Fast Fusion

Point Cloud Alignment. Each frame of a LiDAR contains a *point cloud*, a collection of points with 3D coordinates. These coordinates are in the LiDAR's own frame of reference. Fusing frames from two different LiDARs is the process of converting all 3D coordinates of both point clouds into a common frame of reference. *Alignment* computes the transformation matrix for this conversion.

Prior work has developed Iterative Closest Point (ICP) [13] techniques that *search* for the lowest error alignment. The effectiveness of these approaches depends upon the initial guess for LiDARs' poses. Poor initial guesses can result in local minima. SAC-IA [14] is a well-known algorithm to quickly obtain an initial guess for ICP. As we demonstrate in §IV, with SAC-IA's initial guesses, ICP generates poor alignments on full LiDAR frames.

Initial Guess using Minimal Information. Besides the point clouds, SAC-IA requires no additional input. CIP uses an algorithm to obtain good initial guesses using minimal additional input. Specifically, CIP only needs the distances on the ground between a reference LiDAR and all others to get good initial guesses.[4] We can measure these distances using, for example, an off-the-shelf laser rangefinder [15].

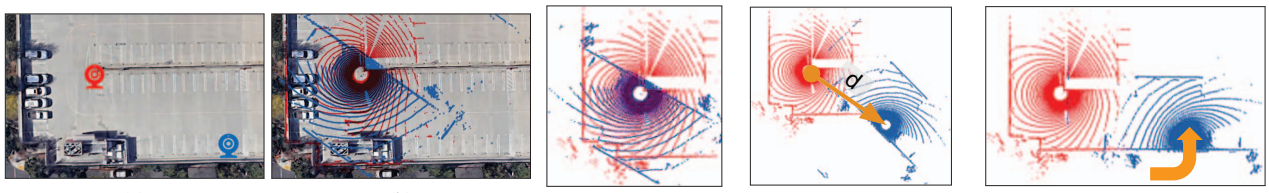
We now describe the algorithm for two LiDARs L_1 and L_2 (as shown in Fig. 4(a)); the technique generalizes to multiple LiDARs as described in Alg. 1. The inputs are two point clouds C_1 and C_2 (more generally, N point clouds captured from the corresponding LiDARs at the same instant (Fig. 4(b) and Fig. 4(c)), and the distance d on the ground between the LiDARs. The output is an initial guess for the pose of each LiDAR. We feed these guesses into ICP to obtain good alignments. The algorithm conceptually consists of three steps:

Fix base coordinates. Set L_1's x and y coordinates to be $(0, 0)$ *i.e.,* base at origin (Alg. 1 line 3). Then, assume that L_2's base is at $(d, 0)$ (Fig. 4(d)).

Estimate height, roll and pitch. In this step, we determine: the height of each LiDAR z_i, the roll (angle around the x axis), and pitch (angle around the y axis). For these, CIP relies on fast *plane-finding* algorithms [16] that extract planes (Alg. 1 lines 1 and 4) from a collection of points. These techniques output the equations of the planes. Assuming that the largest plane is the *ground-plane* (a reasonable assumption for roadside LiDARs), CIP aligns the z axis of two LiDARs with the normal to the ground plane (Alg. 1 line 5). In this way, it implicitly fixes the roll and pitch of the LiDAR. Moreover, after the alignment, the height of the LiDARs z_i is also known (because the z axis

[3]Vehicles use SLAM to estimate their own motion and heading.

[4]LiDAR GPS locations as input result in poor alignment (§IV).



(a) (b) (c) (d) (e)

FIGURE 4: An illustration of CIP's point cloud alignment algorithm. (a) A top down view of a parking lot with two LiDARs shown by red (L_1) and blue (L_2) icons. (b) The inputs to initial guess algorithm are point clouds (C_1 and C_2) in the respective LiDAR's coordinate system along with the ground distance d between them. (c) Figure (b) with background removed. (d) To fix the base coordinates, CIP displaces C_2 by the ground distance d. (e) CIP rotates both C_1 and C_2 by small yaw increments to find the combination with the least distance between the point clouds.

Input : 1. $\mathbf{C} = \{C_1, \cdots, C_N\}$; C_i is LiDAR L_i's point cloud.
2. $D = \{d_2, \cdots, d_N\}$; d_i is the distance from L_i to the reference LiDAR L_1.
Output: $T = \{T_2, T_3, \cdots, T_N\}$; T_i is the transformation matrix from C_i to L_1's coordinate system.

```
1 SegmentGroundPlane ( C_1 );
2 for i ← 2 to N by 1 do
3 |   AlignPosition ( C_i, C_1, d_i );
4 |   SegmentGroundPlane ( C_i );
5 |   AlignGroundPlane ( C_i, C_1 );
6 |   EstimateYaw ( C_i );
7 end
```

Algorithm 1. Estimating an initial guess for alignment.

is perpendicular to the ground plane).

Estimate yaw. Finally, to determine yaw (angle around the z axis), we use the technique illustrated in Fig. 4(e) (Alg. 1 line 6). In this technique, CIP rotates both point clouds C_1 and C_2 with different yaw settings until it finds a combination that results in the smallest 3D distance[5] between the two point clouds. We have found that ICP is robust to initial guesses for yaw that are within about 15-20° of the actual yaw, so CIP discretizes the search space by this amount. In case of Fig. 4(e), CIP calculates the initial guess for the yaw by rotating the blue point cloud (C_2).

Obtaining Alignment. CIP repeats this procedure for every other LiDAR L_i with respect to L_1, to obtain initial guesses for the poses of every LiDAR (Alg. 1 line 2). It feeds these into ICP to obtain an alignment.

Alignment is run only once when installing the LiDARs. Re-alignment may be necessary if a LiDAR is replaced or re-positioned. Alignment is performed infrequently but is crucial to CIP's accuracy (§IV-C).

Per-frame Stitching. LiDARs generate frames at 10 fps (or more). In Fig. 2, when each LiDAR generates a frame, the fusion stage performs *stitching*. Stitching applies the coordinate transformation for each LiDAR generated by the alignment resulting in a fused frame (Fig. 2(c)).

B. Reusing 3D Bounding Boxes

CIP's efficiency results from reusing the *3D bounding box* of a participant (Fig. 2(e)) in processing steps. After stitching,

[5]The 3D distance between two point clouds is the average distance between every point in the first point cloud to its nearest neighboring point in the second point cloud.

Centroid			Axes	Dimension
Tracking	Speed Estimation	Cyber-phy Association	Heading Estimation	Planning

TABLE II: CIP reuses properties of the bounding boxes in multiple modules to ensure low latency.

CIP extracts the bounding box by performing *background subtraction* on the fused point cloud to extract points belonging to dynamic objects. On these points, it applies *clustering* to determine points belonging to individual objects. Finally, it runs a *bounding box estimation* algorithm on these points. These use well-known algorithms, albeit with some optimizations; we describe these in §II-D. CIP uses the bounding box for many of its algorithms (Table II); we describe these below.

Tracking. To associate objects across frames (*tracking*), CIP uses a Kalman filter to predict the position of the *centroid* of the 3D bounding box. Then, it finds the best match (in a least-squares sense) between predicted positions and the actual positions of the centroids in the frame. Although tracking in point clouds is a challenging problem [17] for which research is exploring deep learning, a Kalman filter works *exceedingly well* in our setting. The biggest challenge in tracking is occlusions: when one object occludes another in a frame, it may be mistaken for the other in subsequent frames (*ID-switch*). Because our fused frame includes perspectives from multiple LiDARs, ID-switches occur rarely (§IV).

Speed Estimation. To estimate *speed* of a dynamic object, CIP measures the distance between the centroid of the bounding box in one frame and the centroid w frames in the past (w is a configurable window size). It then estimates speed by dividing the distance by the time to generate w frames.

Cyber-physical Association. CIP needs to associate an object seen in the LiDAR with a cyber endpoint (*e.g.,* an IP address). This is important so that CIP can send that object customized results *i.e.,* perception results relevant to that object or a customized trajectory planned for that vehicle. For this step, CIP uses a calibration step performed once. Given a vehicle for which we know the cyber-physical association (*e.g.,* LiDAR installer's vehicle), we estimate the transformation between the trajectory of the vehicle seen in the LiDAR view with the GPS trajectory (details omitted for brevity). CIP uses this to transform a vehicle's GPS trajectory to its expected trajectory in the scene, then matches actual scene trajectory to expected trajectory in a least-squares sense. To define the scene trajectory,

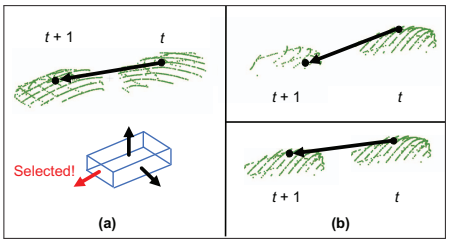


FIGURE 5: The figure shows the points belonging to a vehicle in two successive frames t and $t+1$. (a) Strawman approach for heading determination. (b) CIP's approach.

we use the centroid of the vehicle's bounding box.

The centroid of the bounding box is an easily computed and consistent point within the vehicle that simplifies these tasks. Because we have multiple LiDARs that capture a vehicle from multiple directions, the centroid of the bounding box is generally a good estimate of the actual centroid of the vehicle.

Besides these, CIP (a) estimates heading direction from the axes of the bounding box (discussed in §II-C) and (b) uses the dimensions of the box to represent spatial constraints for planning (discussed in §III).

C. Fast, Accurate Heading Vectors

To compute the *motion vector* of a vehicle, CIP first determines, for each object, its instantaneous *heading* (direction of motion), which is one of the three surface normals of the bounding box of the vehicle. It estimates the motion vector as the average of the heading vectors in a sliding window of w frames. Most autonomous driving stacks can obtain heading from SLAM or visual odometry (§V), so little prior work has explored extracting heading from infrastructure LiDAR frames.

A Strawman Approach. Consider an object A at time t and time $t+1$. Fig. 5(a) shows the points belonging to that object. Since those points are already in the same frame of reference, a strawman algorithm finds the vector between the centroid of A at time t and centroid of A at time $t+1$. Then, the heading direction is the surface normal from the bounding box that is most closely aligned with this vector (Fig. 5(a)). We have found that the error distribution of this approach can have a long tail (although average error is reasonable). If A has fewer points in $t+1$ than in t (Fig. 5(b), upper), the computed centroid will be different from the true centroid, which can induce significant error.

CIP's Approach. To overcome this, CIP uses ICP to find the transformation matrix between A's point cloud in t and in $t+1$. Then, it places A's point cloud from t in frame $t+1$ (Fig. 5(b), lower). Finally, it computes the vector between the centroids of these two (so that the centroid calculations are based on the same set of points). As before, the heading direction is the surface normal that is most closely aligned with this vector.

GPU Acceleration. ICP is compute-intensive even for small object point clouds. If there are multiple objects in the frame, CIP must run ICP for each of them. We have experimentally found this step to be the bottleneck. Thus, we developed a fast GPU-based implementation of heading vector estimation, which reduces the overhead of this stage (§IV-D).

A typical ICP implementation has four steps: (1) estimating correspondence between the two input point clouds, (2) estimating transformation between the two point clouds, (3) applying the transformation to the source point cloud and (4) checking for ICP convergence. The first step requires a nearest neighbor search; instead of using octrees, we adapt a parallelizable version described in [18] but use CUDA's parallel scanning to find the nearest neighbor. The second step shuffles points in the source point cloud then applies the Umeyama algorithm [19] for the transformation matrix. We re-implemented this algorithm using CUDA's demean kernel and a fast SVD implementation [20]. For the third and fourth steps, we developed custom CUDA kernels. This step scales linearly with the number of vehicles but parallelizes easily to multiple GPUs; at intersections with many vehicles, CIP can use edge computing resources with multiple GPUs.

D. Optimizations

Background Subtraction. CIP removes points belonging to static parts of the scene[6]. This is: (a) especially crucial for voluminous LiDAR data, and (b) feasible in our setting because LiDARs are static. It requires a calibration step to extract a *background point cloud* from each LiDAR [21], then creates a *background fused frame* using the results from alignment. To extract the background point cloud, CIP takes the intersection of a few successive point clouds and the aggregating intersections taken at a few different time intervals.

Subtraction before Stitching. CIP can subtract the background fused frame from each fused frame generated by stitching. We have found that removing the background from each LiDAR frame (using its background point cloud), and then stitching points in the residual point clouds can significantly reduce latency. Stitching scales with the number of points, which this optimization reduces significantly.

Leveraging LiDAR Characteristics. Many LiDAR devices only output *returns* from reflected laser beams. Generic background subtraction algorithm requires a nearest-neighbor search to match a return with the corresponding return on the background point cloud. Some LiDARs (like Ouster [22]), however, indicate *non-returns* as well, so that the point cloud contains the output of every beam of the LiDAR. For these, it is possible to achieve fast background subtraction by comparing corresponding beam outputs in a point cloud and the background point cloud.

Computing 3D Bounding Boxes. On the points in the fused frame remaining after background subtraction, CIP uses a standard clustering algorithm (DBSCAN) [23] to extract multiple clusters of points where each cluster represents one traffic participant. Then, it uses an off-the-shelf algorithm [24], which determines a minimum oriented bounding box of a cluster using principal component analysis (PCA). From these, it extracts the three surface normals of the object: the vertical

[6]Static parts of the scene (e.g., an object on the drivable surface) might be important for path planning. CIP uses the static background point cloud to determine the drivable surface; this is an input to the planner (§III-B). We omit this for brevity.

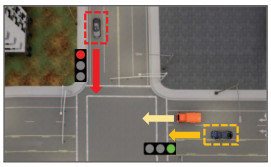


FIGURE 6: The orange truck obstructs ego-vehicle's (yellow box) view of the red-light violator.

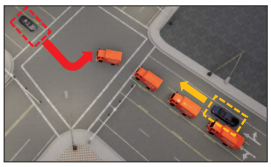


FIGURE 7: The orange trucks obstruct the ego-vehicle's (yellow box) view of the left-turning car (red box).

axis, the axis in the direction of motion, and the lateral axis.

III. USE CASES

Beyond describing CIP, it is important to demonstrate its utility. To this end, we describe its use in (a) augmenting vehicle perception and (b) offloading planning to the edge. These improve traffic safety and throughput respectively (§IV).

A. Augmenting a Vehicle's Perception

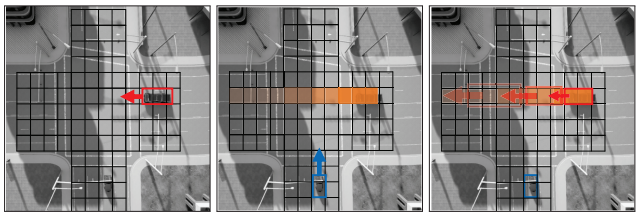
3D sensors mounted on autonomous vehicles are prone to line-of-sight-limitations and occlusions. NHTSA [25] has highlighted a number of scenarios where line-of-sight limitations can cause potential traffic accidents [26]. For instance, in Fig. 6 the ego-vehicle (bounded in yellow) has the right-of-way and attempts to cross the intersection. However, it is unaware of an oncoming red-light violating vehicle (bounded in red) which is occluded by the orange truck. As a result, the ego-vehicle will crash into the red-light violating vehicle. The same is true for Fig. 7 where the ego-vehicle (bounded in yellow) cannot see the vehicle taking the unprotected left turn (bounded in red).

In such traffic scenarios, if the vehicle had access to other 3D views that could sense the oncoming vehicle, it could avoid the traffic accident. With LiDARs mounted at the intersection, CIP can wirelessly transmit its perception outputs to all vehicles. Vehicles, after positioning these bounding boxes and motion vectors in their own coordinate system [27], can fuse them with results from their on-board perception stack.

Finally, they feed the fused results to their on-board planning module. In an autonomous vehicle, the planner [28] generates, every LiDAR frame, short-term *trajectories* (at the timescale of 100s of milliseconds) that the vehicle must follow. A trajectory is a sequence of way-points, together with the precise times at which the vehicle must arrive at those way-points. Because the fused perception results contain the obstructed vehicles, the planner is aware of their presence and their motion. As a result, it can devise collision-free trajectories for vehicles to enable safer driving.

B. Offloading Planning to the Edge

In §III-A, each vehicle plans its trajectory independently. However, because CIP has a comprehensive view of the intersection, it can actually plan trajectories for *all* vehicles on edge compute. While this may seem far-fetched, there already exists at least one company [6] exploring this capability in limited settings. In malls and airports, this capability uses infrastructure sensors and edge-based planning to guide vehicles to and out of their parking spaces.



(a) (b) (c)

FIGURE 8: (a) The first planned vehicle (red) can use the entire drivable space. (b) The second vehicle (blue) treats the first as an obstacle (orange). Different shades represent different times at which grids are occupied. (c) The motion-adaptive buffer around a vehicle is proportional to its speed.

FIGURE 9: With autonomous driving's decentralized planning, in the absence of a traffic light controller, vehicles come to a deadlock at the intersection where they are unable to cross it.

Offloading planning to the edge can improve traffic throughput at intersections. Instead of traffic lights, the planner can regulate the speed of each car so that all cars can traverse the intersection safely, possibly without stopping. Traffic-light free intersections [29] are a long sought after goal in the transportation literature.

To demonstrate this, we have adapted a fast *single-robot* motion planner, SIPP [30]. The input to SIPP is a goal for a robot, the positions over time of the dynamic obstacles and an occupancy grid of the environment (Fig. 8(a)). The output is a provably collision-free shortest path for the robot. At each frame, the planner must plan trajectories for every CIP-capable vehicle. Without loss of generality, assume that vehicles are sorted in some order. The edge-offloaded planner iteratively plans trajectories for vehicles in that order: when running SIPP on the i-th vehicle, it represents all $i-1$ previously planned vehicles as dynamic obstacles in SIPP. Fig. 8(b) illustrates this, in which the trajectory of a previously planned vehicle is represented as an obstacle when planning a trajectory for the blue vehicle.

This approach has two important properties:

1) All trajectories are collision-free. Two cars i and j cannot collide, since, if $j > i$, j's trajectory would have used i's as an obstacle, and SIPP generates a collision-free trajectory for each vehicle.

2) This planner cannot result in a traffic deadlock. A deadlock occurs when there is a cycle of cars in which each car's forward progress is hampered by another. Fig. 9 shows examples of deadlocked traffic with two, three, and four cars (these scenarios under which these deadlocks occurred are described in §IV-G). However, our edge-offloaded planner cannot deadlock. If there exists a cycle, there must be at least one pair of cars i, j where $i < j$ and i blocks j. But this is not possible, because, when planning for j, the planner represents i's trajectory as an obstacle.

Because it was designed for robots, SIPP makes some

idealized assumptions: all robots use the planned trajectories, all have the same dimensions, no trajectories are lost and robots can start and stop instantaneously. In our implementation, we adapted SIPP to relax these but omit details for brevity.

IV. Evaluation

Our evaluations demonstrate CIP's performance and accuracy, and its potential for improving traffic safety and throughput.

A. Methodology

Implementation. We implemented CIP and the two use cases discussed in §III on the Robot Operating System (ROS [31]). ROS provides inter-node (ROS modules are called *nodes*) communication using publish-subscribe, and natively supports points clouds and other data types used in perception-based systems. CIP runs as a ROS node that subscribes to point clouds, processes them as described in §II, and publishes the results. The offloaded planner (§III-B) builds on top of an open-source SIPP implementation [32], runs as a ROS node, subscribes to the CIP results, and publishes trajectories for each vehicle. CIP requires 6909 lines of C++ code, and the use cases 3800.

Real-world Traces. We evaluated CIP on a large open source multi-LiDAR intersection dataset (LUMPI [33]). This dataset contains approximately 2.5 hours of 3D data collected across several days from upto five LiDARs mounted at a busy intersection in Hanover, Germany. These LiDARs include two 16-beam LiDARs (Velodnye VLP-16) and three 64-beam LiDARs (Velodyne HDL-64, Hesai Pandar64, and Hesai PandarQT). CIP is designed to handle such sensor heterogeneity.

For all evaluations in this paper, we ran CIP on an "edge compute" device, an AMD 5950x CPU (16 cores, 3.4 GHz) and a GeForce RTX 3080 GPU. This device has significantly less compute than a commercial edge offering; for example, a server on Google's distributed cloud edge has 96 vCPUs and 4 GPUs [8]. In that sense, our evaluation is conservative relative to what one might expect from a real deployment.

Real-world Testbed. Besides evaluating CIP on real-world traces, we also built our own real-world testbed consisting of four Ouster LiDARs [22] (an OS-1 64, an OS-0 128, and two OS-0 64). Three of these have a field of view of 90° while the last one has a 45° field of view.

Simulation. The LUMPI dataset does not contain ground-truth, so we complement our evaluations with a simulator, which helps us evaluate CIP accuracy and scaling. We use CarLA [26], an industry-standard photo-realistic simulator for autonomous driving perception and planning. It contains descriptions of virtual urban and suburban streets, and, using a game engine, can (a) simulate the control of vehicles in these virtual worlds, and (b) produce LiDAR point clouds of time-varying scenes. Unless otherwise noted, our simulation based evaluations focus on intersections; several challenge scenarios for autonomous driving focus on intersections [34].

Metrics. We quantify end-to-end performance in terms of the 99th percentile of the latency (***p99 latency***) between when a LiDAR generates a frame and when CIP produces its outputs for that frame. To quantify accuracy of individual components, we use metrics described in prior works (defined later).

B. CIP Performance

We ran CIP on 2.25 hours of point clouds traces from the LUMPI dataset. In these experiments, we measured the average, 99th percentile and end-to-end latency of CIP. Our results show that CIP is able to achieve the 99th percentile latency less than 100 ms.

The LUMPI dataset provides two traces (Fig. 10), one with three infrastructure LiDARs and another with five. In the 3-LiDAR setup, the LiDARs are deployed on one side of a 4-way intersection. The 5-LiDAR setup adds two more lidars to cover the other side of the intersection. The latter covers more of the intersection, so CIP detects more traffic participants.

Fig. 11 and Fig. 12 show the perception latency and number of participants for each frame. In the 3-LiDAR setup, CIP's median latency is 38.7ms and p99th latency is 71.6ms. In the 5-LiDAR setup, CIP's median latency is 56.1ms and p99th is 88.8ms. CIP's latency roughly scale with the number of participants in the intersection. In the trace with the 5-LiDAR setup, a small number of frames exceed the 100ms target; these frames have more than 50 participants and more than 30 vehicles. The number of participants in the 5-LiDAR setup is about $1.5\times$ more than the 3-LiDAR setup. Overall, it shows that CIP can comfortably support, with modest computing resources, such an busy intersection with more than 40 participants and can still maintain the tail latency within 100 ms.

C. CIP Accuracy

Because the LUMPI dataset does not have ground-truth, we evaluated CIP's accuracy using real-world data that we collected using our own testbed.[7] On this dataset, we manually labeled ground-truth positions. We also evaluated CIP's accuracy using CarLA. We show that CIP's positioning, heading, speed, and tracking accuracies are comparable to that reported in other works.

Metrics. We report positioning error (in m), which chiefly depends upon the accuracy of alignment. We measure heading accuracy using the average deviation of the heading vector, in degrees, from ground truth. For the accuracy of velocity estimation, we report both absolute and relative errors. Finally, we use two measures to capture tracking performance [35]: multi-object tracking accuracy (MOTA) and precision (MOTP). The former measures false positives and negatives as well as ID switches (§II); the latter measures average distance error from the ground-truth track.

Results. Table III summarizes our findings. Positioning error is about 8-10 cm in CIP, both in simulation and in the real world; the state-of-the-art LiDAR SLAM [36] reports about 15 cm error. Our heading estimates are comparable to prior work that uses a neural network to estimate heading. Speed estimates are highly accurate, both in an absolute sense (error of a few cm/s) and in a relative sense (over 97%).

[7]We also use this dataset to quantify latency with offloaded planning; §IV-D.

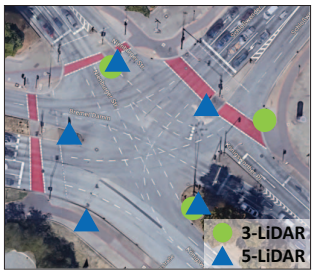


FIGURE 10: LiDARs placement in the LUMPI dataset.

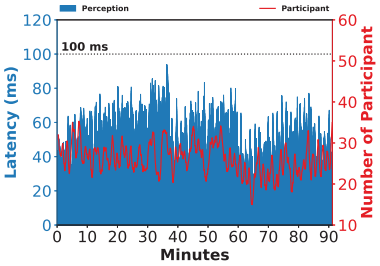


FIGURE 11: CIP performance with a 3-LiDAR setup in the LUMPI dataset.

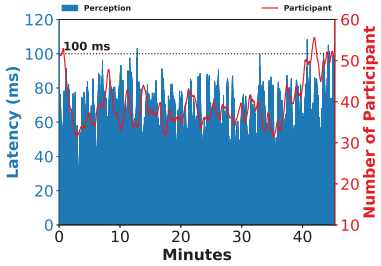


FIGURE 12: CIP performance with a 5-LiDAR setup in the LUMPI dataset.

Metric	Real	Sim	Prior
Positioning Error (m)	0.10	0.08	0.15 [36], [38]
Heading Error (°)	8.17	6.45	5.10 [39]
Speed Error (m/s)	0.04	0.06	—
Speed Accuracy (%)	98.80	97.49	—
MOTA (%)	100	99.54	84.52 [37]
MOTP (m)	0.12	0.08	—

TABLE III: Perception accuracies from real-world data and simulation compared to prior works.

		RMSE (m)	
		Average	Std Dev
Simulation	**CIP**	**0.03**	**0.02**
	SAC-IA	39.2	13.4
	GPS	23.8	11.1
Real-world	**CIP**	**0.09**	**0.04**
	SAC-IA	11.8	13.9
	GPS	13.0	1.2

TABLE IV: CIP's novel alignment algorithm outperforms existing state-of-the-art initial alignment algorithms.

Finally, tracking is also highly accurate. In the real-world experiment, tracking was perfect. In simulation, with 10 vehicles concurrently visible, MOTA is over 99%, and CIP outperforms the state-of-the-art neural network in 3D tracking [37], for two reasons. The neural network solves a harder problem, tracking from a moving LiDAR. Our fused LiDAR views increase tracking accuracy; when using a single LiDAR to track, MOTA falls to 93%. Finally, MOTP is largely a function of positioning error, so it is comparable to that value.

Alignment Accuracy. Although alignment is performed only once, its accuracy is crucial for CIP; without accurate alignment, CIP's perception components could not have matched the state-of-the-art (Table III).

Comparison alternatives. To contextualize CIP's alignment performance, we compared it against two other ways of obtaining an initial guess for ICP: a standard feature-based approach, SAC-IA [14]; and using GPS. In this experiment, we use point clouds from our simulations and real-world experiments. The three approaches estimate initial guesses for the pose of three LiDARs, and feed those poses to ICP for building a stitched point cloud. In our evaluations, we report the average root mean square error (RMSE) between the stitched point cloud (*after* running ICP on the initial guesses) for every approach against a ground-truth.

Results. CIP's alignment results in errors of a few cm (Table IV), almost 2-3 orders of magnitude lower error than the competing approaches, which explains why we chose this approach. SAC-IA [14] does not take any inputs other than the point clouds and estimates the transformation between two point clouds using 3D feature matching. However, this works well only when point clouds have a large amount of overlap. In our setting, LiDARs are deployed relatively far from each other resulting in less overlap, and SAC-IA is unable to extract matching features from multiple LiDAR point clouds. Using GPS for alignment provides a good initial guess for the relative *translation* between the LiDARs. However, GPS cannot estimate the relative *rotation* between LiDARs, so results in poor accuracy.

D. Latency Breakdown and Scaling

Setup. To explore the total latency with more vehicles and to understand the breakdown of latency by component, we designed several scenarios in CarLA with increasing numbers of vehicles traversing a 4-way intersection. At this intersection, both streets have two lanes in each direction. We varied the number of vehicles from 2 to 14, to understand how CIP's components scale. To justify this range of the number of vehicles, we use the following data: the average car length is 15 ft [40] and the width of a lane is 12 ft [41]. Allowing for lane markers, medians, and sidewalks, let us conservatively assume that the intersection is 60 ft across. Suppose the intersection has traffic lights. Then, if traffic is completely stalled or moving very slowly, at most 4 cars can be inside the intersection per lane, resulting in a maximum of 16 cars (*i.e.,* the maximum capacity of the intersection is 16). If cars are stalled, CIP does not incur much latency since it does not have to estimate motion, heading, or plan for these, so we limit our simulations to 14 vehicles.[8] On the other hand, if traffic is moving at 45 mph (or 66 fps) and cars maintain a 3-second [42] safe following distance, then at most one car can be within the intersection per lane, for a total of 4 cars.

Breakdown for CIP. Table V depicts the breakdown of 99-th percentile (p99) latency by component for CIP as well as the total p99 latency, as a function of the number of vehicles in the scene. In all our experiments, CIP processed frames at the full frame rate (10 fps).

The total p99 latency for CIP increases steadily up to 82 ms

[8]We have verified that, above 14 vehicles, end-to-end latencies actually drop.

	Number of Vehicles					
Component	**2**	**4**	**7**	**10**	**12**	**14**
BG Subtraction	7.5	8.9	9.9	9.9	11.0	18.7
Stitching	0.08	0.11	0.13	0.13	0.17	0.19
Clustering	4.3	5.8	11.2	13.0	22.2	20.5
Bounding Box	0.06	0.07	0.09	0.15	0.17	0.16
Tracking	0.1	0.15	0.21	0.28	0.37	0.38
Heading Vector	8.8	12.9	24.0	28.8	41.9	52.6
Total	**18.5**	**26.4**	**43.7**	**47.1**	**69.9**	**81.5**

TABLE V: p99 per-frame latency (in milliseconds) for perception. We exclude latency numbers for motion vector estimation which are on the order of a few microseconds.

for 14 vehicles[9] from 19 ms for 2 vehicles. This highlights perception's data dependency (§II); performance of some components depends on the number of participants. These numbers suggest that modest off-the-shelf compute hardware that we have used in our experiments might be sufficient for traffic management at moderately busy intersections. This data dependency also suggests that deployments of CIP will need to carefully provision their infrastructures based on historical traffic (similar to network planning and provisioning).

The three most expensive components are background subtraction, clustering and heading vector estimation. Background subtraction accounts for about 10 ms, but depends slightly on the number of vehicles; to be robust, it uses a filter (details omitted in §II-A) that is sensitive to the number of points (or vehicles). Clustering accounts for about 20 ms with 14 vehicles and is strongly dependent on the number of vehicles since each vehicle corresponds to a cluster.

Heading estimation accounts for nearly 65% of perception latency, even after GPU acceleration (§II-C). These results show that heading vector estimation not only depends on the number of vehicles, but on their dynamics as well. When we ran perception on 16 vehicles, p99 latency actually dropped; in this setting, 16 vehicles congested the intersection, so each vehicle moved very slowly. Heading vector estimation uses ICP between successive object point clouds; if a vehicle hasn't moved much, ICP converges faster, accounting for the drop.

Other components are lightweight. Stitching is fast because of the optimization described in §II-A. Bounding box estimation is inherently fast. Track association is cheap because it tracks a single point per vehicle, the centroid of the bounding box. Motion estimation takes a few microseconds and relies on positions computed during stitching. Thus, leveraging abstractions and reusing values from earlier in the pipeline help CIP meet latency targets (§II-C).

Benefits of optimizations. Table VI quantifies the benefits of our optimizations. Stitching before background subtraction requires nearly 70 ms in total; reversing the order reduces this time by 6.7×. By exploiting LiDAR characteristics (§II-A), CIP can perform background subtraction in 1.5 ms per frame.[10] A CPU-based heading vector estimation requires nearly 1 s which would have rendered CIP infeasible; GPU acceleration (§II-C) reduces latency by 35×.

[9]For many of our experiments, including this one, we have generated videos to complement our textual descriptions. These are available at an ***anonymous YouTube channel***: https://www.youtube.com/@cip-iotdi24.

[10]Table V does not include this optimization, since it can only be applied to some LiDARs

Optimization	Before	After	Ratio
BG subt before stitching	67.7	10.0	6.7
Exploiting LiDAR characteristics	9.9	1.5	6.6
Heading vector GPU acceleration	1057.7	28.8	36.6

TABLE VI: Impact of optimizations on p99 latency

Calibration Steps. Finally, alignment (§II-A) of 4 LiDARs takes about 4 minutes. This includes not just the time to guess initial positions, but to run the ICP (on a CPU). Because it is invoked infrequently, we have not optimized it.

E. Perception Augmentation: Safety

In this and subsequent sections, we quantify the feasibility and benefits of the use cases in §III. We begin by demonstrating the increased safety resulting from augmenting an autonomous vehicle's perception with CIP outputs (§III-A). CIP has a comprehensive view of an intersection, so it can lead to increased safety. To demonstrate this, we implemented two scenarios in CarLA from the US National Highway Transportation Safety Administration (NHTSA) precrash typology [34]; these are challenging scenarios for autonomous driving [26].

Red-light violation. A orange truck and the ego-vehicle (yellow box) approach an intersection (Fig. 6). An oncoming vehicle (red box) on the other road violates the red traffic light. The orange truck can see the violator and hence avoid collision, but the ego-vehicle cannot.

Unprotected left-turn. The ego-vehicle (yellow box) heads towards the intersection (Fig. 7). A vehicle (red box) on the opposite side of the intersection makes an unprotected left-turn. The ego-vehicle's view is blocked by the orange trucks.

Methodology and Metrics. In each scenario, CIP augments the ego-vehicle's on-board perception. When comparing against (un-augmented) autonomous driving, to ensure a more-than-fair comparison, we (a) equip autonomous driving with ground-truth (perfect) perception and (b) use an on-board SIPP planner in single-mode (plan only for the ego-vehicle) for autonomous driving. The alternative would have been to use an open-source stacks like Autoware [2] which has its own perception and planning modules. However, in these experiments, we are trying to understand the impact of augmenting a vehicle's perception with CIP, so we chose a simpler approach that equalizes implementations.

For both scenarios, we vary speeds and positions of the ego-vehicle and oncoming vehicle to generate 16 different experiments. We then compare for what fraction of experiments each approach can guarantee safe passage.

Results. In both scenarios (Table VII), autonomous driving ensures safe passage in fewer than 20-40% of the cases. CIP ensures safety in all cases because it senses the oncoming traffic that is occluded from the vehicle's on-board sensors[11]. This gives the planner enough time to react and plan a collision avoidance maneuver — in this case, stop the vehicle. Of the two cases, the unprotected left-turn was the more difficult one for CIP as it is for autonomous driving, which fails

[11]Please see YouTube channel for videos. https://www.youtube.com/@cip-iotdi24

NHTSA Scenario	Safe Passage (%)	
	CIP	*Autonomous Driving*
Red-light Violation	100	37
Unprotected Left-turn	100	18

TABLE VII: With more comprehensive perception, CIP can provide safe passage to vehicles in both scenarios.

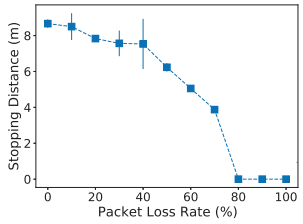


FIGURE 13: CIP can ensure collision-free trajectories with up to 70% packet-loss rates.

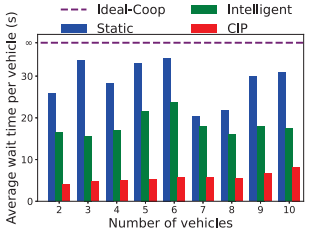


FIGURE 14: CIP minimizes average wait times (seconds) for vehicles with traffic light free intersections.

more often in this case. Yet, CIP is able to guarantee safe passage in all 16 cases. In the red-light violation scenario, CIP senses the oncoming traffic early on and has enough time to react. However, in the unprotected left-turn, the ego-vehicle is traveling relatively fast and the oncoming traffic takes the left-turn at the last moment. Even in this case, CIP gives the vehicle enough time to react. Though CIP has a smaller time to react, its motion-adaptive bounding box and stopping distance estimation ensure that the vehicle stops on time.

Robustness to Packet Loss. In our implementation, the centralized planner transmits trajectories wirelessly to vehicles. It generates trajectories over a longer time horizon (§III-B) to be robust to packet loss. To quantify its robustness, we simulated packet losses ranging from 0 to 100% for in the red-light violation scenario (Fig. 6). For each loss rate, we measure the stopping distance between the ego-vehicle and the oncoming traffic which violates the red-light. Higher stopping distances are good (Fig. 13). As we increased the packet-loss, the stopping distance decreased because the ego-vehicle was operating on increasingly stale information. Even so, CIP ensures collision-free passage for the ego-vehicle through the intersection till 70% loss, with minimal degradation in stopping distance till about 40% loss.

F. Offloaded Planner: Latency

In this section, we measure the performance of a real-world deployment with CIP running an offloaded planner on the same device as the CIP stack. To do this, we deployed four LiDARs at the corners of a busy four-way intersection (Fig. 2(a)) with heavy pedestrian and vehicular traffic in a large metropolitan area. These LiDARs connected to the edge compute device using Ethernet cables. The edge compute connected to Raspberry Pis on the vehicles through Wi-Fi (to proxy 5G). We collected data for nearly 30 minutes; we measured and report the end-to-end latency for every frame.

Metrics. We measure the end-to-end latency of CIP and the centralized planner. This is the time from when CIP receives 3D point clouds to when a vehicle receives its trajectory from the centralized planner over the wireless network.

Results. Fig. 15 shows the end-to-end latency for each frame,

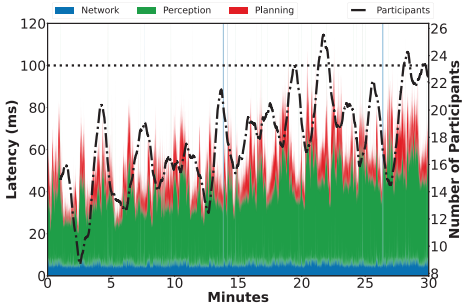


FIGURE 15: Per-component end-to-end latency (left y-axis) along with the number of traffic participants (right y-axis) from a deployment of CIP at a busy intersection in the real-world for over 30 minutes.

for over 30 minutes (approximately 18,000 frames), broken down by component. The average end-to-end latency is 57 ms and p99 latency is 91 ms. This shows that CIP can operates under the 100 ms latency budget for autonomous driving used by Mobileye [43]. Moreover, CIP processed LiDAR input at full frame rate. Lastly, unlike autonomous driving pipelines today which plan for only a single vehicle, CIP with offloaded perception can plan for 10 s of vehicles simultaneously within the 100 ms latency budget.

On average, CIP detects 18 traffic participants per frame at the intersection during our experiment. As our experiment progressed, the traffic at the intersection steadily increased, as shown by the dotted line in Fig. 15 (representing the running average of traffic participants for 600 frames or 60 seconds). Because CIP's latency depends on the number of traffic participants, this contributed to the slow increase in end-to-end latency towards the end of the experiment.

From this graph, we also observe that network latency is small *i.e.,* the average is 9 ms, whereas p99 is 17 ms. The same is true for planning latency for which the average is 11 ms and p99 is 26 ms. Planning latency scales with the number of vehicles. As the number of vehicles increase through the course of the experiment, planning latency also increases. However, in overall, perception latency (37 ms in average and p99 of 60 ms) dominates, which motivates the careful algorithmic and implementation choices in §II.

Scaling Offloaded Planning. To better understand how offloaded planning scales with the number of participants, Table VIII depicts the p99 planning latency from simulations. As expected, there is a dependency on the number of vehicles, since CIP individually plans for each vehicle. Planning latencies can be slightly non-monotonic – the planning cost for 12 vehicles is more than that for 14 – because the planner's graph search can depend upon the actual trajectories of the vehicles, not just their numbers.

G. Offloaded Planner: High-Throughput Traffic Management

In this section, we demonstrate CIP's use of offloaded planning to improving traffic throughput. Intersections contribute significantly to traffic congestion [44]; ***traffic-light free intersections*** can reduce congestion and wait times. An offloaded planner, enabled by CIP, because it centrally plans trajectories for all CIP-capable vehicles, can plan collision-free

Number of Vehicles	2	4	7	10	12	14
p99 Latency (ms)	2.4	5.1	11.9	17.1	28.7	24.2

TABLE VIII: p99 per-frame planning latency in milliseconds.

trajectories at an intersection without traffic lights. We have verified this in simulations as well.

Wait-time Comparison. A traffic-light free intersection can significantly reduce wait times at intersections, thereby enabling higher throughput. To demonstrate this, we compared CIP's average wait times (using a centralized planner) at an intersection against three other approaches:

- *Static.* An intersection with static traffic lights.
- *Intelligent.* Intelligent traffic light control [45] which prioritizes longer queues.
- *Ideal-Coop.* A traffic-light free intersection where autonomous vehicles use centralized perception but on-board planning (§III-A). This represents an idealized version of cooperative perception [3] because it assumes that every vehicle can see all other traffic participants.

For the first two approaches, we obtained policies from published best practices [46]. In all experiments, we placed four LiDARs at an intersection in CarLA.

Results. Compared to *Static* and *Intelligent*, CIP reduces average wait times for vehicles by up to 5× (Fig. 14). It performs better than even *Intelligent* because it can minimize the stop and start maneuvers at the intersection (by preemptively slowing down some vehicles), and hence can increase overall throughput, leading to lower wait times.[11]

Interestingly, beyond about 6 vehicles, wait times for strategies that use traffic lights (*Static* and *Intelligent*) drop. Recall that our intersection has two lanes in each direction. As the number of vehicles increases, the probability that a vehicle can pass the intersection without waiting increases. For example, with *Intelligent*, a vehicle waiting at an intersection can trigger a green light, so a vehicle arriving in the adjoining lane can freely pass. Even so, CIP has lower wait times than other alternatives up to 10 vehicles (wait time comparisons are similar beyond this number, results omitted for brevity).

Fig. 14 shows infinite wait times for *Ideal-Coop*. That is because, although one might expect that a decentralized planner might have comparable throughput to CIP, we found that it ***led to a deadlock*** (§III-B) with as few as two vehicles (each vehicle waited indefinitely for the other to make progress, resulting in zero throughput).[11] Fundamentally, this occurs because, with *Ideal-Coop*, vehicles lack global knowledge of planning decisions. Thus, deadlocks can happen at intersections without traffic lights even for more practical cooperative perception approaches which use on-board planners [3].

V. Related Work

Connected Autonomous Vehicles. Network connectivity in vehicles has opened up large avenues for research. A large body of work [47] has explored wireless technologies and standards (such as DSRC) for vehicle-to-vehicle and vehicle-to-infrastructure communication. Connected autonomous vehicles have also inspired proposals for cooperative perception [4], [48]–[51], collaborative map updates [52], and cooperative driving [53] in which autonomous vehicles share information with each other to improve safety and utilization. Some have proposed approaches to offload route planning (but not trajectory planning) to the cloud [54]. Others explore *platooning* [55] in which vehicles collaboratively and dynamically form platoons to enable smooth traffic flows. Beyond inter-vehicle collaboration, several proposals have explored infrastructure support for connected autonomous vehicles, with infrastructure augmenting perception [56], [57], or delivering traffic light status [55]. Other work focuses on infrastructure-assisted traffic management at intersections [44]. CIP goes beyond this body of work by demonstrating the feasibility of decoupling both perception and planning from vehicular control.

Infrastructure LiDAR-based Perception. Prior work has explored using infrastructure LiDAR to detect pedestrians [58] and road features such as lanes and drivable surfaces [59], [60], and to warn vehicles of impending collisions [61]. One work [62] proposes a genetic algorithm based LiDAR alignment, but unlike CIP, it has not explored the efficacy of an entire perception pipeline built on top of LiDAR fusion.

Point Cloud Alignment. CIP's alignment builds upon point cloud registration techniques [63]. Prior work has matched features [64]; these don't work well for CIP, where LiDARs capture the scene from very different perspectives.

Deep Neural Nets for 3D Detection and Tracking. For vehicle-mounted LiDARs, prior work has developed expensive neural nets for point cloud based detection [65], [66] and tracking [67], [68]. These are for vehicle-mounted LiDAR and are computationally expensive; CIP exploits static LiDARs and can use more efficient algorithms, §IV-C.

Motion Estimation. Heading and speed can be estimated using DNNs [68], SLAM [69], or visual odometry [70]. CIP uses a lightweight technique since it relies on static LiDARs.

VI. Conclusions

Fast cooperative infrastructure perception using multiple infrastructure sensors can enable novel automotive and outdoor mixed reality applications. CIP contains a suite of algorithms that generates cooperative perception outputs with a p99 latency of 100 ms, while still being as accurate as the state-of-the-art. It achieves this using careful alignment, reuse of visual abstractions, and systems optimizations including accelerator offload. When used to augment vehicle perception, it can improve safety. When used in conjunction with offloaded perception, it can increase traffic throughput at intersections.

References

[1] Baidu, "Apollo: Open source autonomous driving," 2017.

[2] K. Miura, S. Tokunaga, N. Ota *et al.*, "Autoware toolbox: Matlab/simulink benchmark suite for ros-based self-driving software platform," in *RSP*, 2019.

[3] H. Qiu, P. Huang, N. Asavisanu *et al.*, "Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving," in *MobiSys*, 2022.

[4] S. Shi, J. Cui, Z. Jiang *et al.*, "Vips: Real-time perception fusion for infrastructure-assisted autonomous driving," in *MobiCom*, 2022.

[5] "How Chattanooga is Achieving Vision Zero with Ouster Lidar," https://ouster.com/blog/how-chattanooga-is-achieving-vision-zero-with-ouster-lidar/.

[6] "Inside seoul robotics's contrarian approach to autonomous vehicle tech," https://techcrunch.com/2022/09/22/seoul-robotics-aims-to-automate-vehicles-movement-via-its-3d-sensor-platform-closes-25m-funding/.
[7] D. Trends, "A Self-Driving Car in Every Driveway? Solid-State Lidar is the Key," https://www.digitaltrends.com/cars/solid-state-lidar-for-self-driving-cars/, 2018.
[8] "Google distributed cloud edge," https://cloud.google.com/distributed-cloud-edge, 2022.
[9] "Qualcomm 5G," https://www.qualcomm.com/invention/5g, 2020.
[10] S. Lin, Y. Zhang, C. Hsu *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *ASPLOS*, 2018.
[11] A. Geiger, P. Lenz, C. Stiller *et al.*, "Vision meets robotics: The kitti dataset," *IJRR*, 2013.
[12] "Google Maps," www.google.com/maps.
[13] Y. Chen and G. G. Medioni, "Object modeling by registration of multiple range images," in *ICRA*, 1991.
[14] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3d registration," in *ICRA*, 2009.
[15] F. Duchon, M. Dekan, L. Jurisica, and A. Vitko, "Some applications of laser rangefinder in mobile robotics," *Journal of Control Engineering and applied informatics*, vol. 14, no. 2, pp. 50–57, 2012.
[16] K. G. Derpanis, "Overview of the ransac algorithm," *Image Rochester NY*, 2010.
[17] V. Vaquero, I. del Pino, F. Moreno-Noguer *et al.*, "Deconvolutional networks for point-cloud vehicle detection and tracking in driving scenarios," in *ECMR*, 2017.
[18] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using gpu," in *CVPR Workshops*, 2008.
[19] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Comput. Archit. Lett.*, 1991.
[20] M. Gao, X. Wang, K. Wu *et al.*, "Gpu optimization of material point methods," *TOG*, 2018.
[21] A. G. Kashani, M. J. Olsen, C. E. Parrish *et al.*, "A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration," *Sensors*, 2015.
[22] Ouster, "Ouster LiDAR," https://ouster.com/, 2020.
[23] M. Ester, H.-P. Kriegel, J. Sander *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, 1996.
[24] "Find minimum oriented bounding box of point cloud," http://codextechnicanum.blogspot.com/2015/04/find-minimum-oriented-bounding-box-of.html, 2015.
[25] W. G. Najm, R. Ranganathan, G. Srinivasan *et al.*, "Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications," US. NHTSA, Tech. Rep., 2013.
[26] CarLA, "Carla autonomous driving challenge," https://carlachallenge.org/.
[27] K. Nawaz Khan, A. Khalid, T. Yash, K. Dantu, and F. Ahmad, "VRF: Vehicle Road-side Point Cloud Fusion," in *MobiSys*, 2024.
[28] B. Paden, M. Cap, S. Z. Yong *et al.*, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE T-IV*, 2016.
[29] R. Tachet, P. Santi, S. Sobolevsky *et al.*, "Revisiting street intersections using slot-based systems," *PLOS ONE*, 2016.
[30] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *ICRA*, 2011.
[31] M. Quigley, K. Conley, B. Gerkey *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on OSS*, 2009.
[32] Whoenig, "Library with Search Algorithms for Task and Path Planning for Multi Robot/Agent Systems," https://github.com/whoenig/libMultiRobotPlanning.
[33] S. Busch, C. Koetsier, J. Axmann *et al.*, "Lumpi: The leibniz university multi-perspective intersection dataset," in *IV*. IEEE, 2022.
[34] W. G. Najm, R. Ranganathan, G. Srinivasan *et al.*, "Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications," US. NHTSA, Tech. Rep., 2013.
[35] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP*, 2008.
[36] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *ICRA*, 2015.
[37] H.-N. Hu, Q.-Z. Cai, D. Wang *et al.*, "Joint monocular 3d vehicle detection and tracking," in *ICCV*, 2019.
[38] I. Cvišić, I. Marković, and I. Petrović, "Soft2: Stereo visual odometry for road vehicles based on a point-to-epipolar-line metric," *IEEE Tran. on Robotics*, 2022.
[39] S. Casas, W. Luo, and R. Urtasun, "IntentNet: Learning to Predict Intention from Raw Sensor Data," in *CoRL*. PMLR, 2018.
[40] "Average car length," https://anewwayforward.org/average-car-length/.
[41] "Average lane width," https://safety.fhwa.dot.gov/geometric/pubs/mitigationstrategies/chapter3/3_lanewidth.cfm.
[42] "3-second rule for safe following distance," https://www.travelers.com/resources/auto/travel/3-second-rule-for-safe-following-distance.
[43] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv*, 2016.
[44] M. Khayatian, M. Mehrabian, E. Andert *et al.*, "A survey on intersection management of connected autonomous vehicles," *ACM Trans. Cyber-Phys. Syst.*, 2020.
[45] F. Ahmad, S. A. Mahmud, and F. Z. Yousaf, "Shortest processing time scheduling to reduce traffic congestion in dense urban areas," *IEEE Trans. Syst. Man Cybern. Syst.*, 2016.
[46] "Signal cycle lengths," https://nacto.org/publication/urban-street-design-guide/intersection-design-elements/traffic-signals/signal-cycle-lengths/.
[47] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas," *IEEE T-ITS*, 2018.
[48] H. Qiu, F. Ahmad, F. Bai *et al.*, "Avr: Augmented vehicular reality," in *MobiSys*, 2018.
[49] X. Zhang, A. Zhang, J. Sun *et al.*, "Emp: edge-assisted multi-vehicle perception," in *MobiCom*, 2021.
[50] Y. He, L. Ma, Z. Jiang *et al.*, "Vi-eye: Semantic-based 3d point cloud registration for infrastructure-assisted autonomous driving," in *MobiCom*. ACM, 2021.
[51] Y. He, C. Bian, J. Xia *et al.*, "Vi-map: Infrastructure-assisted real-time hd mapping for autonomous driving," in *MobiCom*, 2023.
[52] F. Ahmad, H. Qiu, R. Eells *et al.*, "Carmap: Fast 3d feature map updates for automobiles," in *NSDI*, 2020.
[53] R. Dariani and J. Schindler, "Cooperative strategical decision and trajectory planning for automated vehicle in urban areas," in *ICVES*, 2019.
[54] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annual Reviews in Control*, 2018.
[55] J. Schindler, R. Dariani, M. Rondinone *et al.*, "Dynamic and flexible platooning in urban areas," in *AAET*, 2018.
[56] D. Ravipati, K. Chour, A. Nayak *et al.*, "Vision based localization for infrastructure enabled autonomy," in *ITSC*, 2019.
[57] S. Gopalswamy and S. Rathinam, "Infrastructure enabled autonomy: A distributed intelligence architecture for autonomous vehicles," in *IEEE IV*, 2018.
[58] J. Zhao, H. Xu, H. Liu *et al.*, "Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors," *Transp. Res. Part C Emerg.*, 2019.
[59] J. Wu, H. Xu, and J. Zheng, "Automatic background filtering and lane identification with roadside LiDAR data," in *ITSC*, 2017.
[60] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Mach. Vis. Appl.*, 2014.
[61] O. Aycard, "Intersection Safety Using Lidar and Stereo Vision Sensors on a Demonstrator Vehicle," *Transportation*, 2011.
[62] R. Yue, H. Xu, J. Wu *et al.*, "Data registration with ground points for roadside LiDAR sensors," *Remote Sensing*, 2019.
[63] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *ICRA*, 1991.
[64] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *IROS*, 2008.
[65] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *CVPR*, 2018.
[66] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Std: Sparse-to-dense 3d object detector for point cloud," in *ICCV*, 2019.
[67] H. Qi, C. Feng, Z. Cao, F. Zhao, and Y. Xiao, "P2b: Point-to-box network for 3d object tracking in point clouds," in *CVPR*, 2020.
[68] W. Luo, B. Yang, and R. Urtasun, "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting With a Single Convolutional Net," in *CVPR*, 2018.
[69] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Trans. Robotics*, 2017.
[70] A. Rosinol, M. Abate, Y. Chang *et al.*, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *ICRA*, 2020.