FusionEye: Perception Sharing for Connected Vehicles and its Bandwidth-Accuracy Trade-offs

Hansi Liu*, Pengfei Ren†, Shubham Jain‡, Mohannad Murad†, Marco Gruteser*, Fan Bai†
*Rutgers University, North Brunswick, NJ, US
†General Motor Inc, Warren, MI, US
‡Old Dominion University, VA, US
*{hansiiii,gruteser}@winlab.rutgers.edu,
†{pengfei.ren,fan.bai,mohannad.murad}@gm.com,
‡jain@cs.odu.edu

***Abstract*—Automated driving and advanced driver assistance systems benefit from complete understandings of traffic scenes around vehicles. Existing systems gather such data through cameras and other sensors in vehicles but scene understanding can be limited due to the sensing range of sensors or occlusion from other objects. To gather information beyond the view of one vehicle, we propose and explore FusionEye - a connected vehicle system that allows multiple vehicles to share perception data over vehicle-to-vehicle communications and collaboratively merge this data into a more complete traffic scene. FusionEye uses a self-adaptive topology merging algorithm based on bipartite graph. We explore its network bandwidth requirements and the trade-off with merging accuracy. Experimental results show that FusionEye creates more complete scenes and achieves a merging accuracy of 88% with 5% packet drop rate and transmission latency around 200ms. We show that richer vehicle descriptors offer only marginal accuracy improvements compared to lower communication overhead options.**

***Index Terms*—Connected Vehicles, ADAS, Vehicle Verification**

I. Introduction

The robustness of automated driving and advanced driver assistance systems (ADAS) depends on accurate awareness of the traffic scene around the vehicle. Such scene information includes position, velocity, and types of nearby traffic participants. Vehicles capture this information with the help of various on-board sensors, which largely require line-of-sight. Radars and lidars primarily measure distances to objects in specific directions while cameras capture richer visual information about objects and their surrounding environment albeit with less accurate distance measurements. Thanks to recent advances of deep learning in computer vision, vehicles can recognize objects and traffic participants from the captured images and create semantically labeled traffic scenes.

However, occlusion and limited range of these line-of-sight sensors pose restrictions on how completely a car can sense its surrounding traffic. Blind spots can create hazards or cause existing automated driving systems to move slowly due to the resulting uncertainties [1]. While increasing the quantity of sensors or adding specialized non-line-of-sight sensors [2] [3] may yield improvements, it increases the complexity and cost associated with each vehicle.

Therefore, we consider an approach where vehicles share their individual perceptions over wireless vehicle networks and fuse different perspectives into a more complete traffic topology. Existing work on vehicle-to-vehicle communication networks also proposed information sharing but was limited to traffic participants' announcing their own positions [4] [5] [6]. An advantage of perception sharing is that vehicles can also share information about other vehicles or traffic participants that have been detected with sensors but are not equipped with their own transponders. The few projects that have begun to study this approach have focused largely on the extremes of sharing dense lidar point cloud data [7] and sharing basic position estimates of detected traffic participants [8]. Although Meng et al. [9] provide a framework for map sharing between vehicles, its evaluation is limited to simulation and little is known about how to merge such information efficiently and how much information about detected objects should be shared to accurately fuse the views in real world traffic scenarios.

This paper explores the design space of perception sharing and fusing systems by developing and studying a prototype system that allows multiple vehicles to share and fuse surrounding traffic topologies over a wireless network connection.

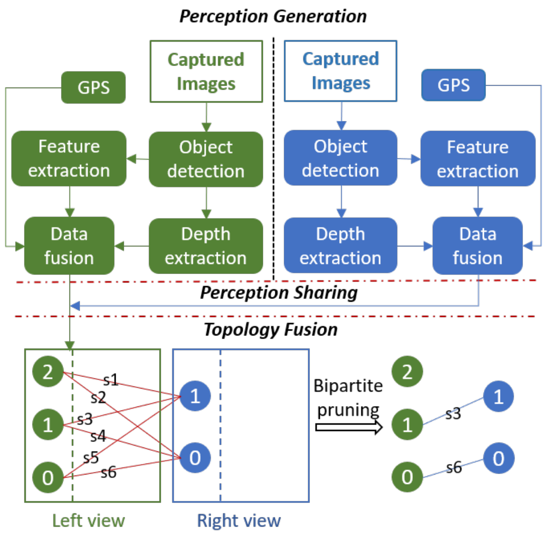


Fig. 1. FusionEye system overview

We focus on sharing information of vehicles, because vehicles are the most common traffic participants and a challenging case due to many vehicles looking similar. For each observing vehicle, we aim to create a merged topology that contains all nearby vehicles, both those detected by its own sensors as well as those detected by other observing vehicles. To achieve a complete and accurate merged topology, it is essential to correctly match the same vehicles from different views. In particular, we design a self-adaptive bipartite merging algorithm that determines which vehicles in two perspectives are the same and need to be merged when constructing the merged topology. At one vehicle node, the algorithm merges two topology maps (one locally created, another received from a nearby vehicle) and treats each detected vehicle as a node in the bipartite graph and determines an optimal matching based on scores of feature representations.

We analyze several feature representations with increasing complexity to explore the network bandwidth and accuracy trade-offs in sharing and merging perceptions of vehicle positions. In our system, each car is equipped with a camera that detects nearby vehicles using a deep neural network, and collects data on multi-lane suburban roads. We first associate vehicles using longitudinal and lateral lane position features determined by our proposed lateral lane position determination algorithm. Then we adopt other features with varying sizes and overhead, including SIFT [10], SURF [11], and color histograms of bounding box patches of the detected vehicles in a captured image to further associate vehicles from two separate views and analyze how these different features affect our merging accuracy as well as network communications.

The contributions of this work are summarized as follows:

- Explore the design space of wireless perception sharing systems that allow a car to benefit from other car' views.
- Design a bipartite merging algorithm, which determines if detected vehicles from two views are the same, to create an accurate merged topology map.
- Develop a lane determination algorithm for fine-grained determination of each detected vehicle's lateral lane position.
- Show through field tests that the algorithm adapts well to outdoor traffic scenes and achieves high merging accuracy.
- Analyze trade-offs between network bandwidth and merging accuracy for vehicle representations of different sizes.

II. FusionEye Design

Each FusionEye vehicle logically has two main roles: (i) it can share its own perceptions with its neighboring vehicles to expand their visions; (ii) it can receive and fuse perceptions of neighboring vehicles with its own perceptions to expand its own vision. The FusionEye design consists of three main functional components: Perception Generation, Perception Sharing, and Topology Fusion. Figure 1 shows these components along with major steps of the FusionEye algorithm.

Perception Generation module acquires and processes sensor (cameras in our implementation) data captured at each vehicular node. The important steps here involve lane awareness, vehicle detection, distance estimation, and feature extraction. From each captured frame, FusionEye identifies all the vehicles in its view using YOLO [12], then determines which lane each car is in, and its distance to the observing vehicle as well as other visual features. *Perception Sharing* module gathers the aforementioned features and its location to form feature representations which are crucial components of the local views, since they allow the system to share critical attributes from the scene without transmitting raw frames. The feature selection process is discussed in detail in Section 4. Features are then assembled into a UDP packet which is broadcast to all neighboring vehicles within the transmitting range. Finally, in *Topology Fusion* module, each vehicle receives perception packets from one neighbor to match and associate same vehicles across different views to create an enhanced and complete topology of the scene. Here we address the problem of association using a proposed bipartite merging algorithm for images of detected vehicles captured by two separate vehicles with different perspectives, illumination, and image sizes.

Sharing full visual information alone would require a large amount of bandwidth. Instead, FusionEye shares light-weight feature information of detected vehicles. We carefully identify the key information with different transmitting costs and explore the effects of sharing them. Specifically, each vehicle shares it's own depth of the detected vehicles, color histogram, SIFT/SURF descriptor of detected vehicles in 2D images. Together with kinematic information (GPS coordinates), this light-weighted visual information is shared with neighboring vehicles for topology fusion.

III. Perception Generation

The Perception Generation module obtains relative positions of observed vehicles to the observing vehicle in both lateral space (i.e., lane position) and longitudinal space (i.e., depth). This is a key step in creating a topology map of the surrounding vehicles and forms an important feature for matching. We develop vision-based solutions for accurate lateral lane matching as well as depth estimation in the following subsections.

A. Lane-level Position Determination

Besides detecting vehicles in cameras' views using YOLO, it is of great significance to identify which lane they're traveling in since understanding the specific lane is crucial for traffic topology generation and fusion. Typical lane detection algorithms are mainly used to identify the lane in which the observing vehicle is traveling. Here we extend lane detection to identify the lane positions of all detected vehicles in a camera view. We first employ Hough Transform on the extracted frames, which aids in identifying if a detected car is to the left or right side of the observing vehicle. To further determine the lane-level position of each detected vehicle (e.g., left1, left2, middle, right1, right2), we propose a heuristic based on the distance from the left (right) bottom corner of a bounding box to the right (left) lane marker and the bounding box's height.

Assuming the camera mounted vehicle is in the "middle" lane, we first determine if the bounding box of a detected car is in the left or right portion of the image. According

Algorithm 1: Get left lane level position

```
Input1: $box$ - Bounding box coordinates
Input2: $lane$ - Lane marker $y = kx + c$
Output: $pos$ - specific lane position for a bounding box
1  $[k, c] \leftarrow$ getParam($lane$)
2  $[l, r, t, b] \leftarrow$ extractCoord($box$)
3  $boxHeight = b - t$
4  $boxCenter = [(l + r)/2, (t + b)/2]$
5  if $boxCenter_y < k \times boxCenter_x + c$ then
6      $boxCorner = [r, b]$
7      $distToLane \leftarrow$ dist($boxCorner$, $lane$)
8      if $boxCorner_y < k \times boxCorner_x + c$ then
9          if $distToLane < boxHeight/2$ then
10             $pos$ = "left1"
11         else
12             $pos$ = "left2"
13     else
14         if $distToLane > boxHeight/3$ then
15             $pos$ = "middle"
16         else
17             $pos$ = "left1"
18 return $pos$
```

to the center coordinates of the bounding box $(\bar{x}, \bar{y})$ and the estimated left/right lanes $y = kx + b$; if $\bar{y} < k_l\bar{x} + b_l$, the bounding box is on the left portion, if $\bar{y} < k_r\bar{x} + b_r$, then the bounding box is on the right portion. (The origin of the pixel coordinate system is at top left corner of the image.) According to perspective projection, as a detected vehicle gets further away from the middle lane, its bounding box will be further away from the detected lane markers, and proportionally smaller, which can be reflected by the bounding box size and its distance to the lane marker. Based on this observation, we compare the bounding box's height h with the distance d from the left (right) corner of the box to the right (left) lane marker for vehicles on one side to determine if it is driving in the first or second left(right) lane. Algorithm 1 presents the detail of the algorithm that determines the specific position of a left bounding box. Similarly, we use the same logic to specify lane positions for the right bounding boxes. Figure 2a and 2b show a typical pair of frames in which lane markers as well as multiple vehicles with lane level positions are determined.

B. Distance Estimation

We consider two approaches for distance estimation to estimate relative distances of the detected vehicles using bounding box coordinates: Mono-camera and RGB-D camera distance estimation. To understand the first method, assume that the world's coordinate frame is coincident with the camera's coordinate frame. Then perspective projection (with no rotation and translation of the camera coordinate frame) formulates the relationship between a 2D image pixel and a 3D real-world point as follows:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{1}$$

where $[x, y, w]$ are 2D homogeneous pixel coordinates, $[X, Y, Z]$ are 3D real-world coordinates. f_x, f_y, c_x, c_y in the matrix denote the intrinsic parameters of the camera. Converting the homogeneous coordinates x into pixel coordinates x_p, we obtain:

$$x_p = f_x \cdot \frac{X}{Z} + c_x \tag{2}$$

The above equation shows that the depth of a point can be estimated by its 2D pixel coordinates with known lane width $(L = X)$ and camera intrinsic parameters. Thus the estimated depth is:

$$d = Z = f_x \times \frac{L}{|x - c_x|} \tag{3}$$

The second approach to obtain a detected car's distance is referring to the depth map captured by a RGB-D stereo camera, which is usually generated for each frame pair. The depth map stores the estimated distance of the corresponding pixel in the image frame, for every pixel (x, y). Since the vehicle detector component outputs the image coordinates where a vehicle was detected, the system can simply retrieve the distance from the depth map at these coordinates. Our implementation focuses on the first approach since we observed that the range of the RGB-D camera we had access to was too limited for the driving scenarios in our experiments.

Finally, we use the estimated distances and lane level positions of the detected vehicles to create a 2D topology map with respect to each observing vehicle, as shown in Figure 2f.

IV. Topology Fusion

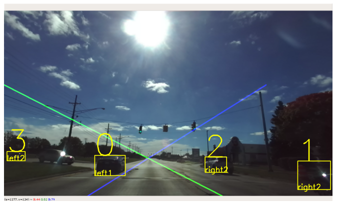
Associating vehicles from different views is essential to merging vehicle perceptions. FusionEye proposes an N-to-M bipartite graph matching algorithm which is flexible in that it can accommodate a range of possible vehicle descriptors. We first briefly introduce the concept of bipartite graph, then discuss how we utilize bipartite graph matching to merge separate views obtained from different vehicles.

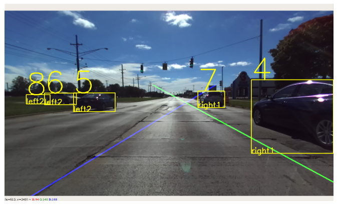
A. Topology Fusion via Bipartite Graph

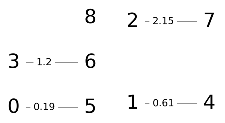
A bipartite graph $G = (V, E)$ contains nodes $v \in V$ that can be separated into two independent sets L and R, and each edge $e \in E$ of the bipartite graph connects a node from set L to a node in set R. In a fully-connected or complete bipartite graph, every pair of nodes in the two sets are connected and no two nodes from the same set are connected. A matching M of a bipartite graph is a subset of edges such that each node in V appears in at most one edge in M. A maximal matching, by definition, is a matching to which no more edges can be added without increasing the degree of a node to two.

B. Vehicle Verification as Bipartite Matching

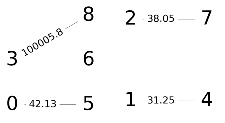
We interpret groups of detected vehicles from car A and car B as two sets in the bipartite graph. As shown in Figure 1, we represent each detected left-sided vehicle viewed from car A using a node in set L and denote each detected left-sided vehicle viewed from car B as a node in set R. Then we connect all the nodes from L and R to construct a fully-connected bipartite graph G in which the scores of the edges describe the similarity between one vehicle to another under certain metrics. The more similar the two vehicles are, the smaller


(a) Left view

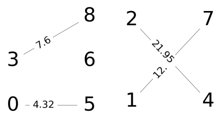

(b) Right view



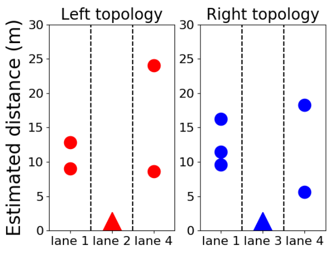
(c) Merging results using depth information only



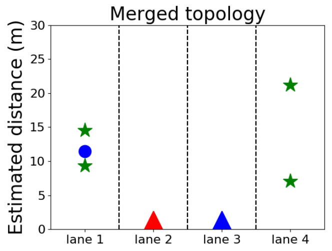
(d) Merging results using depth, color histogram and SURF features



(e) Merging results using depth, color histogram, and SIFT features



(f) Individual topology maps



(g) Merged topology maps

Fig. 2. Merging results of three different merging policies of a pair of synchronized frames. (a)(b): left view and right view, where estimated lane markers are shown in blue (left) and green (right), YOLO [12] detected vehicles are highlighted by bounding boxes with IDs and lane positions. (c)(d)(e): different merging results after bipartite pruning. Left/right bipartite shows merging in left/right side for each view. Each vehicle is represented by red nodes with an ID corresponding to the bounding box ID. The scores of each edge is also shown. (f): Individual topology maps from two views. Two observing vehicles are shown in red and blue triangles and their detected vehicles are represented by red and blue circles respectively. (g): Merged topology via transmitting features of depth, color histogram and SURF descriptors. Green stars indicate the same vehicles observed in both views.

their edge score is. In this manner, we transfer the merging task to a problem of finding the optimal maximal matching of G that has the minimum sum of scores, which indicates that each pair of nodes are most likely to be the same. To assign each edge a discriminative score, we focus on three categories of visual features that comprehensively and uniquely describe each vehicle node, and use them to calculate the scores of the edges that quantify the differences between a pair of nodes in the bipartite graph.

Category 1: Difference based on spatial information. First, lane level positions can help us distinguish two vehicles. Recall that we assign each bounding box of a detected vehicle with a position tag, it is certainly impossible to predict a left-sided vehicle is the same as a right-sided vehicle. Second, the estimated depth of each detected vehicle gives us information of relative position, which provides a certain level of uniqueness since vehicles in moderate traffic conditions tend to keep safety distances with others. Thus each detected vehicle's estimated depth is more likely to be unique from one observer's perspective. Suppose we are constructing a bipartite graph based on the left-sided detected vehicles from two views, for a node u in left view (car A) and node v in right view (car B), their edge score is computed as

$$s_{dist} = ||u.dist - v.dist| - D_{AB}| \tag{4}$$

where D_{AB} represents the distance between car A and car B computed from the Euclidean distance of their GPS readings. Notice that s essentially denotes the depth difference of two detected vehicles under the same coordinate system. Thus we expect s to be small if the vehicles in two views are the same. Although GPS readings in reality contain errors in urban areas, which makes s a noisy measurement, the error will affect all edge scores equally and therefore have little effect on the association decisions. More significant are camera distance estimation errors to observed cars.

Category 2: Representation difference based on Color Histogram. Matching vehicles based on spatial information alone is insufficient and may lead to errors. For example, a detected vehicle u in the left view is compared with two detected vehicles v_1, v_2 in the right view where they are driving side by side. So they have similar estimated depth and the score of edge $u{-}v_1$ is close to the score of $u{-}v_2$. We therefore also use visual information. Considering that we are observing the same scene at the same moment during our experiment, we assume the images of a vehicle observed under two perspectives do not have much illumination variance. Moreover, each image patch cropped based on the vehicle detector bounding box mostly contains a vehicle body with uniform color. Thus we adopt color histogram as our second visual feature. While relatively straightforward we expect them to be effective in this environment based on the aforementioned considerations. For each detected vehicle, we compute its color histogram $\boldsymbol{H}$ where $\boldsymbol{H}$ is a 24×1 vector where the first, second, and third set of 8 elements represents the color histogram for R, G and B channel, respectively. Then, the color histogram score for two nodes u and v in the bipartite graph is computed as

$$s_{hist} = ||\frac{\boldsymbol{H_u}}{||\boldsymbol{H_u}||} - \frac{\boldsymbol{H_v}}{||\boldsymbol{H_v}||}|| \tag{5}$$

Category 3: Representation difference based on SIFT/SURF descriptors. Considering that there still exist cases where color histograms of vehicle patches vary too much due to strong reflections or different backgrounds resulting from perspective changes, we also adopt SIFT and SURF features, which are widely used to find correspondences between scenes or objects because they are invariant to color as well as minor shifting and scaling. For each detected vehicle, we compute SIFT and SURF features within the bounding box image patch and the SIFT/SURF score for two nodes u and v in the bipartite graph is computed as

$$s_{desc} = \frac{\sum_{i}^{N} ||\boldsymbol{desc[i]_u} - \boldsymbol{desc[i]_v}||}{N} \tag{6}$$

where $\boldsymbol{desc}[i]$ represents the 128×1 SIFT descriptor or 64×1 SURF descriptor for key point i and N represents the total

Algorithm 2: Bipartite pruning

```
Input: $G$ - A fully connected bipartite graph
Output: $G_p$ - The pruned bipartite graph
1  $lNodes, rNodes \leftarrow$ getLeftRightLaneNodes($G$)
2  if $len(lNodes) \leq len(rNodes)$ then
3      $side1 = lNodes, side2 = rNodes$
4  else
5      $side1 = rNodes, side2 = lNodes$
6  foreach $index \in indexList$ do
7      $cnt = 0$ ;                        // counter for matches
8      $minScore = +\inf$
9      foreach $s2Node \in side2$ do
10         if $s2Node.isMatched$ == $True$ then
11             $G_c$.removeEdge($side1[i]$, side2Node)
12         else
13             $cnt++$
14             $curScore \leftarrow G_c$.score($side1[i], s2Node$)
15             if $cnt == 1$ then
16                 $minScore \leftarrow curScore$
17                 $msNode \leftarrow s2Node$
18             else if $curScore \geq minScore$ then
19                 $G_c$.removeEdge($side1[i]$, side2Node)
20             else
21                 $minScore \leftarrow curScore$
22                 $G_c$.removeEdge($side1[i]$, $msNode$)
23                 $msNode \leftarrow s2Node$
24         $(msScore, side1[i])$.isMatched $\leftarrow$ Ture
25         $s \leftarrow s + minScore$
26 $G_p \leftarrow min_s(G_c)$
27 return $G_p$
```

number of matched key points from the two bounding box patches. Notice that there are situations in which $N = 0$, which means there are not enough matched key points between the two bounding boxes. Although tweaking the implementations of SIFT/SURF may create more features, it also increases erroneous key point correspondences which negatively affects the similarity scores. For these special cases of insufficient key points, we assign very large scores to indicate that these two image patches likely show different vehicles.

The complexity and robustness of scoring system increases from category 1 to category 3. In our experiments, we explore how these different combinations of scoring policy will affect our merging accuracy as well as transmission qualities since it is indeed our intention to understand the impact of Bipartite graph design options on the end-to-end performance of FusionEye system. Once the scores in the bipartite graphs for all the given frame pairs have been calculated, we propose Algorithm 2 to prune the fully connected bipartite graph into an optimal maximal bipartite graph that has the minimum sum of scores.

$$G_{optimal} = \arg\min_{G} \sum_{u \in L, v \in R} s_{uv} \tag{7}$$

C. Perception Sharing

FusionEye focuses on several different options for sharing visual information. Certain information is much more critical than others during the topology fusion process. At the same time, these visual features have different sizes in terms of communication transmission. For instance, depth information and color histogram are small, taking only tens of bytes; on the other hand, the size of SURF descriptors and SIFT descriptors could vary from 1KB to 100 KB. According to their contributions and data sizes, we put these pieces of visual information into three layers: 1) Kinematic information such as GPS coordinates, is the most important part for topology fusion. 2) Light-weight visual information including object's bounding box, depth, and color histogram. Although it only takes tens of bytes, it plays an important role in topology fusion. 3) Heavy-weight visual information including SURF descriptors and SIFT descriptors, which also contributes to topology fusion. Its size, however, is much larger than the light-weight visual information. Following this information hierarchy, we design our own customized perception sharing protocol. Three information layers are assembled into two types of packets, as seen in Table I. Kinematic information is the most critical one while it takes only a few bytes. So it is put in the header of both packets. Light-weight visual information are assembled into a single packet. This packet is usually small but important. It could be transmitted multiple times accordingly, based on the wireless technology and environment. Heavy-weight visual information is usually too large to fit in a single UDP datagram, so we split them into multiple batches.

TABLE I
PACKET FORMATS

Timestamp	Frame ID	Vehicle ID	Kinematic Info	Light-weight visual Info

Timestamp	Frame ID	Vehicle ID	Kinematic Info	Heavy-weight visual Info	Batch ID

V. FusionEye Implementation

To validate FusionEye, we conduct real-time transmission field tests to evaluate wireless channel bandwidth, latency, and packet drop rate in real-world environments. Figure 3 shows our system setup. We adopt two vehicles, each equipped with a laptop (Intel Xeon E3-1505M v5 CPU, 32 GB of DDR4 RAM, NVIDIA Quadro M2000M GPU). Each laptop is connected to a camera, a UBlox GPS, and a TP-Link Talon AD7200 wireless router. cameras capture frames and compute depth with a resolution of 1920×1080 at 30fps. Ublox GPS samples at a rate of 5Hz. For our implementation only the observing vehicles are equipped with GPS modules. Considering that the exchange of complex visual information requires large bandwidth, we adopt wireless routers that operate in 802.11 b/g/n mixed mode.

Data Collection. Our data collection effort was conducted in Warren, MI, with normal city driving speed of 40 Mph - 50 Mph. Several measurement campaigns had been conducted, with each lasting about 2 hours. During each measurement campaign, the two vehicles were driving side by side on adjacent lanes and cameras were set to record the front view with GPS information associated with each captured frame. We down-sample the video frames every 300ms. Overall, our system captured 30,000 frames and exchanged 3.3 GB information.

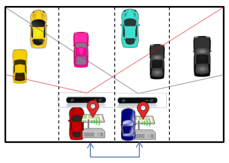


Fig. 3. Experiment setups. Red and blue lines represent perception ranges of red and blue car respectively

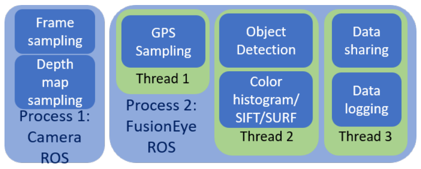


Fig. 4. Parallelized system pipeline. Camera ROS sends captured frame and depth map to FusionEye ROS. The three threads runs concurrently to perform real-time processing and transmission

Implementation. During real-time wireless transmissions field tests, each captured frame and the corresponding depth map are first fed to YOLO's pipeline to generate bounding boxes for detected vehicles. Based on bounding box coordinates, 3 kinds of visual features are extracted: (1) Depth value of the bounding box center. (2) Color histogram. (3) SIFT/SURF features. Once features are extracted for all the bounding boxes in each frame, vehicle A assembles both kinematic and visual information into UDP datagrams, and broadcasts these datagrams to its neighboring vehicles over a WiFi channel. It also stores a copy of the information locally. We then apply our Bipartite graph fusion algorithms to the information that vehicle A stores locally and the information it received from vehicle B. The same procedure is repeated at vehicle B. Finally, we evaluate merging accuracy and transmission qualities under four merging policies that use different combinations of the above 4 information: depth only, depth with color histogram, depth with color histogram and SURF or SIFT features.

Both depth map extraction and object detection need to be performed on GPU since they require significant amount of computational resources. We build customized C++ wrappers for capturing the RGB-D map and YOLO real-time object detection. These two tasks are configured to run in two ROS [13] nodes implemented as two different processes, so they share GPU's resources most efficiently without conflicting with each other. In addition, we design our efficient system pipeline to fully utilize parallel processing, thus enabling faster data processing. As summarized in Figure 4. We assign three threads in the second ROS node for concurrent execution of GPS sampling, distance estimation, feature extraction, data transmission, and logging.

Another critical component of FusionEye is time synchronization. All information shared among vehicles must be synchronized in order to be appropriately fused together. We use NTP for time synchronization. One vehicle is configured as a NTP server of the WLAN, while the other vehicle is configured to synchronize with this NTP server periodically over WiFi.

VI. Evaluation & Analysis

A. Merging Accuracy

To evaluate the performance of our bipartite merging algorithm, we compute true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR) using predictions provided by pruned bipartite graph and ground truth that we manually labeled offline. While a seemingly reasonable baseline comparison might be to use only GPS readings to fuse the topology, recall that we assume only the observing vehicles to share GPS readings. This allows for a scenario where not all vehicles are equipped with the same technology. We therefore focus on comparing performace with different feature combinations. Figure 5 presents precision, recall, and F-scores for 4 merging policies in which different combinations of features are used to compute the edge scores in the bipartite graph. We notice that in general the performance of the merging algorithm improves as we incorporate more fine-grained descriptors to compute edge scores. Naturally, the merging accuracy also improves as the description of a vehicle becomes more comprehensive and robust as more information is taken into account. However, the merging performance using depth information, color histogram, and SIFT features is worse than using depth information only. It is counter-intuitive that SIFT features did not contribute as well as SURF features in distinguishing two vehicles. The reason behind this phenomenon is that the number of SIFT features is much less than the number of SURF features for the same image patch. As a result, there exist situations in which no SIFT key point match exists for two vehicle nodes and the score of their edge is set to be a large number. This uniform score assignment when no SIFT match exists leads to verification errors, since it is possible for two views of the same vehicle to have few SIFT features and yet no key point matches especially when they are observed from a larger distance and the bounding box sizes are small.

Another observation is that the precision is less than recall, which indicates that false negative rate is lower than false positive rate. This can be explained by the process of our bipartite merging algorithm. As we are pruning the complete bipartite graph to be a maximal match, we inevitably preserve some of the false matching since a matching of a complete bipartite graph has edges for all the nodes. Thus, even if nodes from two sets represent different vehicles, they can still be matched if they are the only two nodes in the bipartite graph.

As an example, we present the predictions of different merging algorithm applied to the same pair of image frames shown in Figure 2a and Figure 2b. Figure 2c shows the merging results of using depth as the only information to compute the bipartite scores. Here depth information did not successfully predict similarity between car3 in left view and car6 in right view, since car3 and car6 have almost same depth with respect to the observers. However, once the information of color histogram and SURF descriptors of the bounding box is taken into account, as shown in Figure 2d, the merging decision is correct as car3 from left view is now successfully matched to car8 from right view. Notice that the same car in two views has a similar color distribution as well as similar appearances over the bounding box, which provides color histogram and SURF descriptors that exhibit small Euclidean distance. Figure 2e, on the other hand, is a counter-example

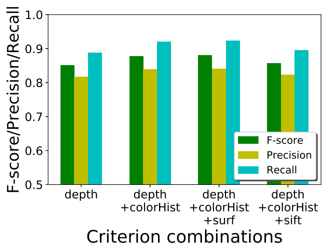


Fig. 5. F-score, precision and recall for different merging policies

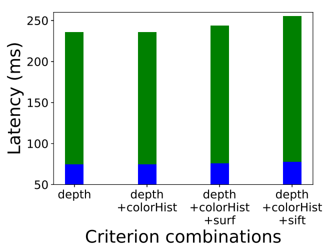


Fig. 6. Different time duration before merging algorithm begins

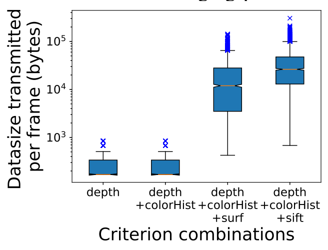


Fig. 7. Distribution of the number of bytes transmitted per frame

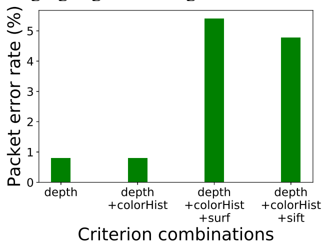


Fig. 8. Packet error rates for different merging policies

showing that SIFT might not be a good criterion for our merging problem, as it failed to distinguish the similarity of $\{$car2, car7$\}$ and $\{$car1, car4$\}$. Regarding perception expansion results, we qualitatively validated that our system can alleviate partial perception situations. As shown in Figure 2f, vehicle 6 is not detected in the left view, thus not plotted in the left observer's topology. However, since the right car successfully detects vehicle 6, its estimated position is plotted in the merged topology as shown by the blue circle in Figure 2g. Besides, FusionEye successfully associates same vehicles ($\{$vehicle 3, vehicle 8$\}$, $\{$vehicle 0, vehicle 5$\}$, $\{$vehicle 2, vehicle 7$\}$ and $\{$vehicle 1, vehicle 4$\}$) that occur in both views in the merged map. Now that we have shown qualitatively that bipartite merging alleviates partial perceptions, it is more valuable to focus on the bandwidth trade-off studies to understand whether partial perception can be alleviated with acceptable resource consumption and under what configurations the system achieves optimal performance.

Each merging accuracy has a different cost. Here we focus on the latency involved with each scheme. Latency is defined by the duration from the time a frame is grabbed by the camera till all necessary information is received by the receiver side so that the merging process can begin for that frame. As shown in Figure 6, we observe that latency increases as we transfer more information through the network. Notice that the latency for merging using depth information is the same as merging using both depth and color histogram information since we transmit both depth and histogram in one packet due to their smaller size compared to SIFT/SURF features. The baseline latency is shown in blue for each bar, which consists of image processing time including vehicle detection using YOLO, estimating depth value according to the bounding box coordinates as well as computing SIFT/SURF descriptors for that bounding box patch. The green part of each bar indicates how much time is needed before all the necessary information is received at the receiver side so that merging algorithm can start. This includes the overhead of constructing each depth packet by combining timestamp, depth value, along with color histogram as a whole, as well as the processing time to split the SIFT/SURF descriptor matrices into a sequence of sub-packets.

B. Transmission Analysis

We compute two categories of error rate during data transmission: incomplete bounding box rate and packet drop rate.

Incomplete Bounding Box. When transmitting a sequence of packets of SIFT/SURF descriptors for a bounding box patch, it is likely to lose packets in transmission and packet receive rate at the receiver end will be affected. In such cases, the received information only describes part of the bounding box. We define the incomplete bounding box rate as the percentage of the bounding boxes whose received SIFT/SURF descriptors are less than transmitted SIFT/SURF descriptors.

The comparison of incomplete bounding box rate under different transmission policy is shown in Figure 9. Notice that when we only use depth compared to when we use depth and color histogram to compute the edge scores, the incomplete bounding box rate is the same as packet drop rate. This is because the packet we transmit is at "bounding box level". In other words, each packet contains depth information and color histogram of the whole bounding box. As long as the packet is successfully received, the corresponding bounding box information reaches the receiver side successfully. Thus, the incomplete bounding box rate is the same as the drop rate for depth packet. SIFT/SURF packets on the other hand, need to be split into multiple packets which renders their data to be at a "sub-box" level. If the number of received feature points is less than the number of transmitted feature points, the corresponding bounding box's information is not completely received. We observe that SIFT packets cause a higher count of incomplete bounding boxes than SURF packets.

Average packet drop rate. We compute the average packet drop rate for each type of packet by calculating the ratio of the number of dropped packets to the total number of transmitted packets of a certain type. As shown in Figure 8, although SIFT sub-packets contribute to more incomplete bounding boxes than SURF sub-packets, the drop rate of SIFT sub-packet is lower than the SURF sub-packet. This counter-intuitive observation gives us an insight that SURF sub-packets have a "clustered" dropping manner in which a large number of SURF sub-packets drop can happen within one bounding box's feature transmission sequence, while SIFT sub-packets are dropped in a "uniform-distributed" manner where more bounding boxes are incomplete but each bounding box only loses less feature points compared to SURF scenarios. The explanation can be further substantiated by checking the transmitted data size distribution per frame at receiver side. As shown in Figure 7, the transmitted data per frame reaches the peak when merging two views using information of depth,

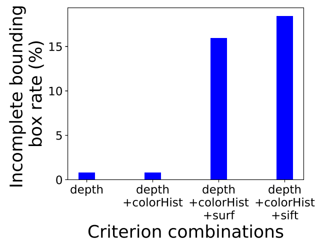


Fig. 9. Incomplete bounding box rate for different merging policies

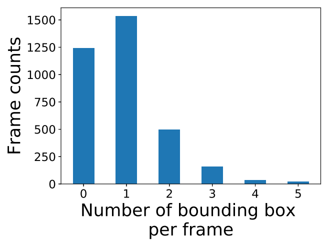


Fig. 10. Bounding box numbers in every frame

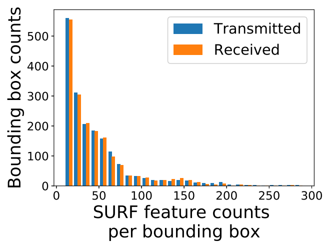


Fig. 11. Transmitted/Received SURF sub-packets for every bounding box

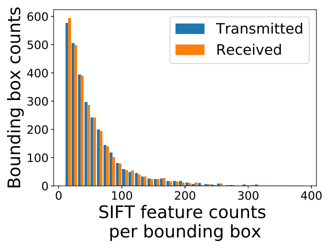


Fig. 12. Transmitted/Received SIFT sub-packets for every bounding box

color histogram, and SIFT packets. Although size of each SIFT sub-packet and SURF sub-packet is the same, we observe more SIFT data is transmitted to the receiver side due to lower drop rate of SIFT sub-packets.

We finally focus on characteristics of the collected data itself. We are interested in how many bounding boxes exist in each frame as well as how many SIFT/SURF features are detected in each bounding box. Figure 10, Figure 11, and Figure 12 show the distribution of bounding box per frame and histogram of SURF/SIFT sub-packet numbers respectively. We observe that for moderate traffic condition, in most cases we have 1 to 2 bounding boxes per frame, and in each bounding box there tend to be less than 100 feature points. Notice that the value of each bin of the histogram of SIFT sub-packet is always greater than the corresponding bin in histogram of SURF sub-packet, which reflects the relationship shown in Figure 7 where SIFT packets contribute to larger size of transmitted data.

VII. Related Work

Cooperative camera systems and connected vehicles. Recent studies of cooperative camera systems including SLAM and visual odometry [14] [15] explored methods leveraging multiple camera's view to improve the objects' location estimation as well as situation awareness. More recently, with the substantiate feasibility studies of DSRC and other connected vehicles systems [16] [17] ,there has been more works that focus on sharing the camera perceptions through V2V networks not just for autonomous vehicles but also for other Advanced Driving Assistance Systems (ADAS). The work of See Through Systems [18] [19] aim to improve driving safety by expanding the front view of a vehicle when it is occluded by the front vehicle in the same lane. The system communicates through DSRC and warps the front car's view to adapt the back car's viewing angle such that the blocking vehicle ahead looks transparent in the back car's augmented view. More recently, OmniView [9] proposed a framework that allows multiple vehicles to share and fuse their local maps but only evaluated the system in terms of reception rate and latency through simulation. In contrast, FusionEye systematically evaluates the topology fusion in a real-world fashion and explores relationship with merging accuracy with communication quality. Inspired by SLAM's philosophy, Hang et al. [7] proposed Augmented Vehicular Reality (AVR) that stitches raw 3D point clouds reconstructed by leading following vehicles into a expanded view. Instead of sharing raw pixel data to simply expand perception range, our system aims to explore the bandwidth requirements and achievable merging accuracy for vehicles that extract and share semantic information such as vehicle features and positions from their sensor data. Although AVR shares additional information in terms of image pixels, topologies generated by FusionEye contain more semantic information such as categories of traffic participants and their distances Besides, FusionEye's network structure saves bandwidth significantly and supports more than a few participating vehicles.

Vehicle verification and association. Recently there also have been active research efforts in the area of vehicle verification and matching via computer vision [20] [21] [22]. In these works, it is common to treat the verification problem as a binary classification problem and apply supervised learning on the collected data set. Feature representation is one of the critical factors to achieve good verification accuracy, as the work of Arrspide and Salgado [20] and Guo et al. [21] proposed to improve Gabor Features in different manners. Furthermore, Hsu et al. [22] proposed a sparse representation that adapted well to non-overlapped views of vehicles. More recently, Deep learning methods also thrives in vehicle verification domain. Liu et al. [23] adopted Triplet Loss [24] with deep neural network to learn robust representations of the on road vehicles and determine whether they are the same based on their Euclidean distances. However, these methods' success rely on the systematic data collecting and time-consuming training processes. Other vehicle association methods in V2V communication systems such as [6] [8] use GPS relative distance as spatial features or explicit vehicle type (SUV, sedan, etc) as visual fingerprints to associate vehicles detected by different observers. These methods either need all participating vehicles in the experiment to be equipped with GPS sensors or require their prior information such as vehicle type. FusionEye differs from these works in a way that only a limited number of equipped observing vehicles are exchanging messages while any other vehicle on the road can be a participant in the merged topology.

VIII. Discussion

Our system mainly focuses on camera-based detection and merging. It is also worth evaluating systems with lidar-based detection, since it may provide better distance estimation accuracy. Although it may not allow us to further extract rich visual features as did in this work, the different characteristics of lidar may allow using other features and it would be interesting to compare the resulting merging accuracy with that of camera-based merging.

The current bipartite merging algorithm algorithm and implementation is limited to cases where the vehicles in one view represent a subset of the vehicles in the other view. For example, if there are no common vehicles but vehicles in both views, the merging algorithm would still falsely match vehicles across those two views. This could be addressed with further pruning of the bipartite graph and more robust feature representation for vehicles.

The experiment in this paper is intended as a first exploration of the design space for feature representations that support topology merging across vehicles. While we started with vehicles, pedestrian, bicyclists and other traffic participants could likely be handled by incorporating respective visual classifiers and adding features for association. Since these traffic participants move more slowly and are often sparsely distributed on roadways (in the United States) they are likely easier to match. Generally, additional scenarios, such as curved lanes, roads with varying number of lanes, relative vehicle positions beyond side-by-side, pedestrians and other traffic participants, and different weather/environments are worth studying to further understand the generality of our findings.

IX. Conclusion

This paper introduces FusionEye, a novel approach in which vehicles share their individual perceptions over wireless vehicle networks and fuse their different views into a more complete traffic topology leveraging our fine-grained lane determination and bipartite merging algorithm. Through extensive empirical experiments, we show that with two vehicles on a roadway, the algorithm adapts well to the challenging real-world urban traffic scenarios and achieve high merging accuracy with low fidelity vehicle descriptors. More interestingly, we found that the richer vehicle descriptors offer only marginal accuracy improvements and that the features from our lane determination algorithm and estimated distances realize most of the achievable gains at much lower communication overhead. These more compact descriptors appear preferable from a bandwidth-accuracy trade-off perspective.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation under Grant No. CNS-1329939.

References

[1] D. Riley, "Report: Waymo self-driving cars are having problems turning around corners," https://siliconangle.com/2018/08/28/report-waymo-self-driving-cars-problems-turning-around-corners/, 2018.

[2] B. R. Chang, H. F. Tsai, and C.-P. Young, "Intelligent data fusion system for predicting vehicle collision warning using vision/gps sensing," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2439–2450, 2010.

[3] F. de Ponte Müller, E. M. Diaz, and I. Rashdan, "Cooperative positioning and radar sensor fusion for relative localization of vehicles," in *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE, 2016, pp. 1060–1065.

[4] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A survey of inter-vehicle communication protocols and their applications," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 2, 2009.

[5] L. Li and F.-Y. Wang, "Cooperative driving at blind crossings using intervehicle communication," *IEEE Transactions on Vehicular technology*, vol. 55, no. 6, pp. 1712–1724, 2006.

[6] S. Demetriou, P. Jain, and K.-H. Kim, "Codrive: Improving automobile positioning via collaborative driving," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 72–80.

[7] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan, "Avr: Augmented vehicular reality," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (Mobisys)*, ser. MobiSys '18. Munich, Germany: ACM, 2018, pp. 81–95. [Online]. Available: http://doi.acm.org/10.1145/3210240.3210319

[8] S. Fujii, A. Fujita, T. Umedu, S. Kaneda, H. Yamaguchi, T. Higashino, and M. Takai, "Cooperative vehicle positioning via v2v communications and onboard sensors," in *Vehicular Technology Conference (VTC Fall), 2011 IEEE*. IEEE, 2011, pp. 1–5.

[9] R. Meng, S. Nelakuditi, S. Wang, and R. R. Choudhury, "Omniview: A mobile collaborative system for assisting drivers with a map of surrounding traffic," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 760–765.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[13] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system."

[14] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *IEEE transactions on pattern analysis and machine intelligence*, 2012.

[15] M. Kaess and F. Dellaert, "Visual slam with a multi-camera rig," Georgia Institute of Technology, Tech. Rep., 2006.

[16] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of dsrc and cellular network technologies for v2x communications: A survey," *IEEE transactions on vehicular technology*, vol. 65, no. 12, pp. 9457–9470, 2016.

[17] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE internet of things journal*, vol. 1, no. 4, pp. 289–299, 2014.

[18] C. Olaverri-Monreal, P. Gomes, R. Fernandes, F. Vieira, and M. Ferreira, "The see-through system: A vanet-enabled assistant for overtaking maneuvers," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 123–128.

[19] P. Gomes, F. Vieira, and M. Ferreira, "The see-through system: From implementation to test-drive," in *Vehicular Networking Conference (VNC), 2012 IEEE*. IEEE, 2012, pp. 40–47.

[20] J. Arrospide and L. Salgado, "Log-gabor filters for image-based vehicle verification," *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2286–2295, 2013.

[21] J.-M. Guo, H. Prasetyo, and K. Wong, "Vehicle verification using gabor filter magnitude with gamma distribution modeling," *IEEE Signal Processing Letters*, vol. 21, no. 5, pp. 600–604, 2014.

[22] S.-C. Hsu, I.-C. Chang, and C.-L. Huang, "Vehicle verification between two nonoverlapped views using sparse representation," *Pattern Recognition*, vol. 81, pp. 131–146, 2018.

[23] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2167–2175.

[24] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.