V3I-STAL: Visual Vehicle-to-Vehicle Interaction via Simultaneous Tracking and Localization

Xiaobai Liu
Department of Computer Science
San Diego State University (SDSU), San Diego, CA
xiaobai.liu@mail.sdsu.edu

ABSTRACT

This paper investigates a visual interaction system for vehicle-to-vehicle (V2V) platform, called V3I. Our system employs common visual cameras that are mounted on connected vehicles to perceive the existence of isolated vehicles in the same roadway, and provides human drivers with imagery situational awareness. This allows effective interactions between vehicles even with a low permeation rate of V2V devices. The underlying research problem for V3I includes two aspects: i) tracking isolated vehicles of interest over time through local cameras; ii) at each time-step fusing the results of local visual perceptions to obtain a global location map that involves both isolated and connected vehicles. In this paper, we introduce a unified probabilistic approach to solve the above two problems, i.e., tracking and localization, in a joint fashion. Our approach will explore both the visual features of individual vehicles in images and the pair-wise spatial relationships between vehicles. We develop a fast Markov Chain Monte Carlo (MCMC) algorithm to search the joint solution space efficiently, which enables real-time application. To evaluate the performance of the proposed approach, we collect and annotate a set of video sequences captured with a group of vehicle-resident cameras. Extensive experiments with comparisons clearly demonstrate that the proposed V3I approach can precisely recover the dynamic location map of the surrounding and thus enable direct visual interactions between vehicles .

1. INTRODUCTION

Motivation Vehicle-to-vehicle (V2V) [53] communication platform can exchange GPS positions, speed, intents and other traffic data between nearby connected vehicles and has shown great promises in avoiding traffic crashes, preventing injuries, and reducing traffic congestions. In the last decade, V2V reached a level of maturity, insofar as vehicle-resident wireless hard-wares have undergone multiple generations to achieve their implementations [50]. However, it is still challenging to deploy V2V system in practice, because V2V-equipped vehicles need to share roads with isolated (non-connected) vehicles which will be dominant in amount for a long period.

MM '16, October 15-19, 2016, Amsterdam, Netherlands
© 2016 ACM. ISBN 978-1-4503-3603-1/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2964284.2964285

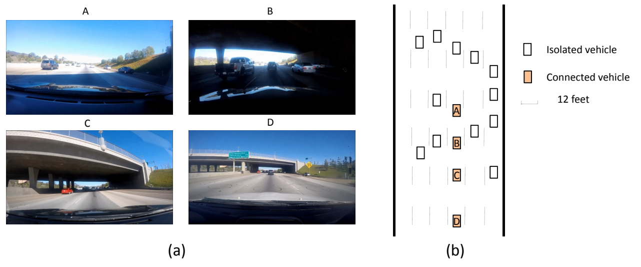

Figure 1: Visual Vehicle-to-Vehicle Interaction (V3I). (a) Inputs: four video frames captured by cameras on the connected vehicles; (b) the fused map of both isolated and connected vehicles. The relative positions of isolated vehicles are automatically estimated. (The road map is used for illustration purpose).

To obtain the traffic data of the isolated vehicles that share the same roadway, an effective and affordable method is to enhance connected vehicles with cameras. Then the isolated vehicles can be perceived and networked with the connected vehicles so as to form a dynamic vehicle network. The human drivers of connected vehicles, for example, are able to access extended range of traffic situations and are notified of potential traffic collisions. This visual system is called *Visual Vehicle-to-Vehicle Interaction* or *V3I*. Our system can work with low penetration rate of connected vehicles and thus allow the smooth transitioning to the next intelligent transportation system.

The key research question of V3I is how to detect, track and localize vehicles that appear in the field of views (FOVs) of the connected vehicles (through cameras). It involves both local visual perceptions and global visual data fusion across the networked cameras. Figure 1 illustrates four FOVs in (a) and the fused map in (b). The main technical difficulties are sourced from the fact that all cameras are moving at relatively high speeds (60-85 miles per hour). In particular, the following issues are ubiquitous in a V3I system.i) While the V3I involves a series of visual perceptions, e.g. detection, tracking, localization, it is crucial to to process all visual inputs with real-time efficiency to allow time-critical warnings or reactions; ii) The localization component is highly ambiguous since it is actually solving regression of high dimension data (video sequence); iii) Since the V2V network is dynamic, there are frequent changes over the spatial configurations of vehicles.

Objectives In this work, we are interested in developing an efficient and scalable V3I system that can extract time-critical, precise, and reliable traffic data with extended range from monocular video sequences. At each time-step, our goal is to construct a dynamic map that includes accurate 3D positions of the nearby vehicles that are either connected or isolated. We will update the map adaptively

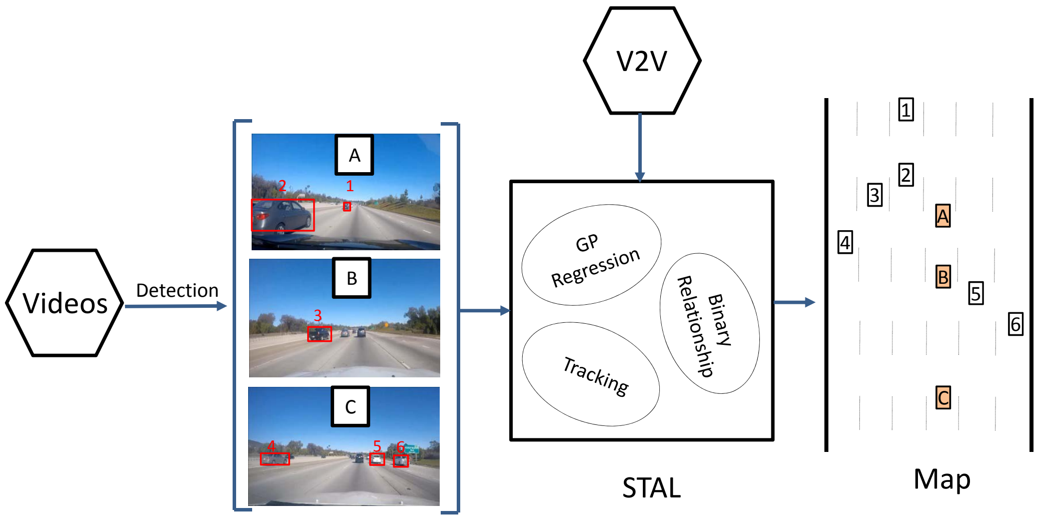

Figure 2: Flowchart of the proposed approach. *Videos*: monocular video sequences captured from multiple connected vehicles; *V2V*: vehicle to vehicle traffic data; *STAL*: the proposed simultaneous tracking and localization method that jointly formulates three components: tracking, regression of vehicle locations and consistencies of a set of between-vehicle spatial relationships; *Map*: location map that includes vehicle positions at every time-step.

over time and use it as an imagery digitalization of the surrounding traffic conditions. The map would be propagated to connected vehicles in the proximity and used as an interaction medium.

The V3I system involves two basic vision tasks: tracking vehicles over time in individual cameras and localizing them in a global 3D map. These two sub-tasks are coupled and mutually beneficial, and should be formulated and solved jointly.

Overview We introduce a unified probabilistic approach for simultaneous tracking and localization (STAL) of vehicles in V3I system. Our approach will exploit both visual information extracted in individual cameras, including lanes, vehicles, etc., and V2V traffic data, e.g. GPS positions. Since none of these observations are reliable nor accurate, it is crucial to perform reasoning while keeping uncertainties to avoid pre-mature decision making.

One of the major components of our approach is to predict the 3D relative locations of isolated vehicles detected in monocular videos. A traditional solution [49] [20] is to calibrate camera parameters and reconstruct 3d position of vehicles based on camera geometry. However, such analytic methods will not work well for V3I system where both cameras and vehicles of interests are moving at relatively high speeds. In this work, we study a calibration-free solution, motivated with the fact that an experienced human driver can roughly estimate the relative distance between a nearby vehicle and the host vehicle. We cast the localization task as a supervised regression problem, which aims to learn to map the visual features of the vehicle images to the 3D locations. In particular, we first collect a set of monocular video sequences and manually annotate the locations of individual isolated vehicles with an interactive annotation toolkit. The annotations are not given in absolute 3D coordinates but only reflect how close the isolated vehicles locates to the host vehicles. At each time-point, we collect the short sequence of video frames in prior as a training sample. Every sample is provided with a continuous label indicative of relative distance. With these training data, we develop a Gaussian Process model [26] [32] with a novel time-series kernel to solve the regression problem. In comparisons to the other discriminative methods, e.g., support vector regression [41], Gaussian Process (GP) models are probabilistic and enjoy the benefits common to all kernel-based models.

Another major component of our approach is a general Bayesian formulation of a set of pair-wise spatial relationships between vehicles. It includes, e.g., left-right (i.e., a vehicle locates on the left to the another one), front-back (i.e., a vehicle locates in front of the other one), or further-closer (depth to the cameras). These spatial relationships are used to test the current solution hypothesis: whether the relationships hold in the estimated location map. The proposed formula is characterized with two folds. First, expressing spatial relationships between vehicles in binary form, instead of in relative coordinates, enables robust measurement from noisy data. Second, although a single relationship carries weak information, there are multiple (hundreds of) relationships even for a short sequence of video frames.

The proposed probabilistic approach will optimize both **discrete variables**, e.g. trajectory identity (ID) of vehicle boxes, and **continuous variables**, e.g., 3D locations, and the optimal solution lies on a joint solution space. While it is in general intractable [46], Markov Chain Monte Carlo (MCMC) method can be used to search the solution space by simulating a Markov Chain walking toward the target distribution. It is a true approximation scheme for the optimal: when run with unlimited resources, it converges to the optimal solution eventually. However, the conventional MCMC algorithms are restricted to their slow mixing rate, which prevents their applications in real-time systems. In this work, we develop a fast MCMC algorithm that can efficiently sample the joint solution space with rapid mixing rate. The key technical contributions are two dynamics. i) Dynamic-I: in the continuous space, we utilize the gradient information of the proposal energy functions to make proposals that are distant from the current solution but are still accepted with high probabilities. ii) Dynamic-II: in the discrete space, we construct multiple adjacent graphs to represent the trajectory solutions and cast tracking as graph coloring problem. At each step we select a cluster of graph nodes, instead of a single node, and change its color to reconfigure the present solution. The two dynamics are alternatively performed following the Metropolis Hasting algorithm [46] to guarantee convergence.

Contributions The three contributions of this paper are (i) a novel V3I problem that have potentials in enhancing the next generation transportation system; (ii) a probabilistic formulation for V3I that integrates both the outputs of a supervised regression model and a large collection of pair-wise binary spatial relationships between vehicles, and (3) a hybrid MCMC optimization algorithm that makes distant proposals in the joint solution space. The proposed approach can generate accurate location maps for both isolated and connected vehicles through common visual sensors, and provide an affordable way for vehicle to vehicle interaction.

2. RELATIONSHIPS TO PREVIOUS WORKS

While V2V techniques have been studied extensively [53, 50], to our best knowledge, V3I is the first system that aims to provide human drivers with imagery situational awareness. The proposed approach is motivated with four research directions in multimedia understanding and computer vision.

Vehicle Detection in images has been extensively studied in the last decade [57]. These detectors can be roughly divided into two categories. The first category tries to combine hand-crafted features, e.g. SIFT [29], HOG [10] [5], and various machine learning algorithms, e.g., boosting [47], SVM [10], decision trees [8]. The other category [23] [18] aims to automatically learn features from images using deep neural network and achieves impressive results fueled by the availability of large-scale image datasets [9] [23]. A recent seminal work is the region-based Convolution Neural Network (R-CNN) [14]. A large variants of R-CNN are proposed to improve convergences [13] or enable end-to-end learning (i.e., getting rid of the post-progressing step) [35]. In this work, we will utilize the R-CNN technique to our real-time system for detecting vehicles of interest in video sequences.

Visual object tracking is a long-standing problem [40] in computer vision. Recently the tracking-by-detection framework [17] has achieved impressive results mostly due to the advances in object detections [11, 35], as aforementioned. Another major stream is to cast visual tracking as data association problem in the Bayesian framework [7]. In particular, Oh et al. proposed a Markov Chain Monte Carlo Data Association (MCMCDA) algorithm, and showed that it clearly outperforms the alternative Bayesian treatments, e.g., joint probabilistic data association (JPDA) [36, 37], multiple hypothesis tracking (MHT) [34], etc. Khan et al. [22] integrated the MCMC sampling within the particle filer tracking framework. Yu et al. [54] utilized the single site sampler for associating foreground blobs to trajectories. Liu et al. [30] introduced a spatial-temporal graph to cast tracking as a graph coloring problem and solve it using MCMC algorithm. These algorithms enjoy the advantages of the Monte Carlo search, e.g., avoiding pre-mature decisions while solving the optimal solution. However, they are restricted to the high computational complexity and can not be applied over real-time systems. In this work, we introduce a few improvements over the MCMC algorithm, including making distant proposals in both continuous and discrete spaces, exploiting the deepest gradient directions and moment information, and casting tracking in the cluster sampling framework. We will demonstrate that, with these modifications, it is feasible for MCMC algorithm to achieve real-time performance.

Joint Recognition and Reconstruction has been investigated for various computer vision problems, including simultaneous localization and mapping (SLAM) [49, 20], image labeling and scene reconstruction [25] [31], object recognition and reconstruction [15], room layout partition and object modeling [16, 39], tracking and camera calibration [24] [55], video reconstruction [45]. In this work, we follow the same methodology but focus on simultaneous tracking and mapping of vehicles of interest, which has not been studied. Fusing visual data on the V2V platform is a new research direction that enables interactions between connected vehicles.

Gaussian Process or GP provides a non-parametric framework for deriving regression or classification with explicit uncertainty models. As a probabilistic generative method, GP requires much less training samples to achieve the same level of performance as the other discriminative models [47, 56]. Although the standard GP suffers from high computational complexity $O(N^3)$ where N is the number of training samples, it is possible to largely reduce the computational cost by imposing sparsity over the training samples [42, 32], e.g., learning a small number of pseudo-input samples [43] [44], or using a sparse subset of the training samples [26, 42]. In computer vision, Kapoor et al. [19] applied GP method for image categorization with human-in-the-loop and achieved impressive results. For all these GP algorithms, the central modeling choice is the specification of a kernel. In this work, we use GP to predict the 3D locations of vehicles and integrate the probabilistic outputs of GP with a unified probabilistic formula. As a technical contribution, we introduce a time-series kernel to explore both appearance and temporal cues in the GP formula, which is a novel concept in this direction.

3. OUR APPROACH

In this section, we introduce our V3I-STAL approach, starting with an introduction of the whole pipeline. Then, we will specify a probabilistic formula.

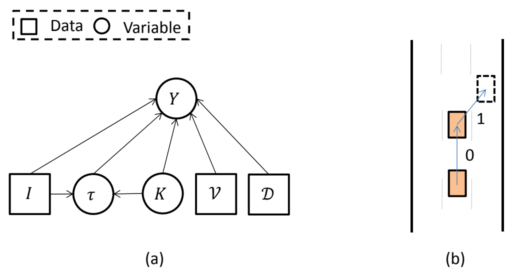

Figure 3: (a) Graphical model of the proposed probabilistic method. (b) state change of a vehicle: staying on the same lane, 0, or changing lane, 1.

3.1 Overview of Our Approach

We consider a set of time-synchronized video sequences that are captured by multiple cameras on a group of connected vehicles in the proximity. Our goal is to track and localize isolated vehicles in videos and generate a global location map at each time-step. We refer this task to *simultaneous tracking and localization* or *STAL*. The cameras on connected vehicles are dynamically networked through V2V protocols and thus form a dynamic ad-hoc camera network. The field of views (FOVs) of cameras might overlap with each other at a certain time and might become disjoint over time.

Figure 2 illustrates the flowchart of the proposed V3I-STAL approach. We utilize the sliding window strategy [30] to deal with streaming video sequences. For the present time window, we start with detecting vehicles and lanes for every monocular video. The detection results along with the status of connected vehicles (e.g., speed, locations) are used as inputs for a probabilistic approach, which aims to solve tracking and localization jointly. The outcome of our STAL approach is a scene map that includes the relative coordinates of both isolated and connected vehicles.

Relative World Coordinate We define a relative coordinate system to facilitate the derivation of our approach. We take one of the connected vehicles as reference and consider its location as the origin, i.e., (0, 0). For any other vehicle in the proximity, its coordinate (y_1, y_2) is defined as: i) $y_1 \in [..., -2, -1, 0, 1, 2, ...]$ indicates the relative lane to the origin where 0 means the same lane with the reference vehicle, -1 means the left lane, and +1 the right lane); ii) $y_2\ in[1, 2, 3, 4, ...]$ which indicates the relative distance projected on the lane direction (larger number indicates being further). In the rest of this paper, 3D vehicle locations are defined in this relative coordinate system unless otherwise specified.

3.2 Probabilistic Formula

We first introduce the notations used in this paper. Let T denote the size of the time-window, or the number of video frames. Let M denote the number of cameras, I^m the m-th video sequence. Let $I = [I^{[1..M]}]$. Let $\mathcal{C}$ denote the pair-wise relationships between vehicles (introduced later), let $\mathcal{D}$ denote the extra training data, $\mathcal{V}$ denote the traffic status of connected vehicles. Let $\mathcal{O} = [\mathcal{I}, \mathcal{C}, \mathcal{D}, \mathcal{V}]$ pool over all the inputs of our approach.

For each camera m, let x^m_{kt} denote the 2D bounding box of the k-th isolated vehicle at the time t. The corresponding 2D trajectory is denoted as $\tau^m_k = [x^m_{kt}]$.Let $\tau = [\tau^m_k]$ pool all the 2D trajectories, $k \in [0..K^m], m \in [1..M]$, where K^m indicates the total number of trajectories to solve in the mth video. τ^m_0 denotes the collection of bounding boxes not assigned to any trajectory. We denote the 3D location of the k-th isolated vehicle at time t by Q^m_{kt}. Let $Q^m_k = [Q^m_{kt}]$ denote the k-th trajectory, $Q = [Q^m_k]$ pool all the 3D trajectories of isolated vehicles. Let R^m_t denote the 3D location of the m-th connected vehicle and $R = [R^m_t]$. Let $Y = \{Q, R\}$. Note that the 2D locations of connected vehicles are fixed (the bottom-central point) throughout the video sequences. Thus, we can describe the solution of STAL as the following representation,

$$W = (K^m, \tau, Y) \tag{1}$$

where

$$\tau = [\tau^m_0, \tau^m_k], \tau^m_k = [x^m_{kt}], \tag{2}$$

$$Y = \{Q, R\}, Q = [Q^m_k], Q^m_k = [Q^m_{kt}] \tag{3}$$

$$R = [R^m], R^m = [R^m_t] \tag{4}$$

$$m \in [1..M], k \in [0..K^m] \tag{5}$$

According to Bayes' rule, we can solve the optimal representation as maximizing a posterior, that is,

$$W^* = \arg\max_W p(W|\mathcal{O}) \tag{6}$$

where the posterior model $p(W|\mathcal{O})$ can be factorized as the product of three probability distributions

$$p(W|\mathcal{O}) \propto \prod_m \Big[p(K^m) \prod_k p(\tau^m_k|K^m, \mathcal{O})\Big] p(Y|\tau, K, \mathcal{O}) \tag{7}$$

The first term $p(K^m) \propto \exp(-K^m)$ is used to encourage the number of trajectories to be small, resulting in a compact representation. The second probability $p(\tau^m_k|K^m, \mathcal{O})$ defines the distribution over 2D trajectories conditioning on the inputs. The third term $p(Y|\tau, K, \mathcal{O})$ defines the distribution over 3D trajectories, which can be further factorized to be the product of three probabilistic models,

$$p(Y|\tau, K, \mathcal{O}) \propto p(Q|\mathcal{D}, \tau, K, \mathcal{I}) p(R|\mathcal{V}, \tau, K, \mathcal{I}) p(Y|\mathcal{C}, \tau, K, \mathcal{I}) \tag{8}$$

The first two terms are distributions over 3D locations of isolated and connected vehicles, respectively. The third term is defined over the pair-wise spatial relationships between vehicles.

Figure 3 (a) illustrates the graphical dependences of the above variables. In the rest of this section, we will elaborate the above probabilistic models .

3.2.1 Probabilistic Model for 2D trajectories

The distribution $p(\tau^m_k|K^m, \mathcal{O})$ is defined over the 2D trajectories, which can be further factorized as

$$p(\tau^m_k|K^m, \mathcal{O}) \propto p(\tau^m_k|K^m) p(I^m|\tau^m_k) \tag{9}$$

where $p(\tau^m_k|K^m)$ is the prior term and $p(I|\tau^m_k)$ is the likelihood term.

Prior In this work, we identify two states of a moving vehicle at each time-step: staying on the same lane as the previous time or not. We will introduce a simple method in Section 4 to detect the state of changing lane. Let $\omega(\tau^m_k, t)$ returns 0 if the vehicle stays on the same lane at time t; otherwise 1. Fig. 3 (b) illustrates the state change. Then, we employ a simple Ising/Potts model [6] to penalize discontinuity of states, i.e.,

$$p(\tau^m_k) \propto \exp\left[-\beta \sum_t \mathbf{1}\Big(\omega(\tau^m_k, t) \neq \omega(\tau^m_k, t-1)\Big)\right] \tag{10}$$

where β is a constant and $\mathbf{1}(\cdot)$ is an indicator function. Eq. (10) is used to constraint the number of times a vehicle change its lane during driving.

Likelihood The likelihood term $p(I|\tau^m_k)$ defines the probability of the observations given the 2D trajectories, which can be factorized as

$$p(I^m|\tau^m_k) = \prod_{x^m_{kt} \in \tau^m_k} \frac{p^f(x^m_{kt})}{p^b(x^m_{kt})} \cdot p^a(x^m_{kt}, x^m_{k,t-1}) \tag{11}$$

where $p^f(\cdot)$ and $p^b(\cdot)$ are foreground and background probabilities, respectively. The first term measures the ratio of the foreground distribution over the background distribution, which can be analogous to a probabilistic foreground/background classifier. We use the detection scores to approximate this ratio, as introduced in Section 4. The second term is defined over the appearance divergence between bounding boxes of the same trajectory. We have, $p^a(x^m_{kt}, x^m_{k,t-1}) \propto \exp\big(-\mathcal{L}(f^m_{kt}, f^m_{k,t-1})\big)$ where f are the visual features extracted from the vehicle boxes, and $\mathcal{L}(\cdot)$ returns the Euclidean distance of two feature vectors. We will introduce the visual features to use in Section 4.

The probability for 2D trajectories are separately defined for individual cameras. While our models follow the previous stochastic data association framework, e.g. [33] [30], we specify a new type of prior model and will solve the discrete variables τ^m_k by maximizing a unified posterior $p(W|I)$, instead of $p(\tau^m_k|K^m, \mathcal{O})$.

3.2.2 Probabilistic Model for Localizing Isolated Vehicles

The probabilistic model $p(Y|\mathcal{D}, \tau, K, \mathcal{I})$ is used to specify the mappings between visual features and 3D locations of vehicles. In other words, given visual features of a vehicle, we aim to predict its relative distance from the camera. This is feasible since the visual features of a vehicle vary while locating at different distances and angles w.r.t. the camera. Such coherences have been modeled by Wu et al. [51] with an image scaling theory, e.g., a distant vehicle will appear with less sketch/boundary features in comparisons to a nearby vehicle.

We formalize the localization problem in the Gaussian process framework. We collect a training dataset where every sample consists of a short sequence of vehicle boxes, e.g. 30-50 frames. For every vehicle in videos, we annotate its location in the relative world coordinate (as aforementioned) considering the camera position as the origin. We use multiple vehicle boxes, instead of a single box, to construct one training sample for two reasons. First, it is technically critical to obtain motion features to estimate depth from a single camera [38]. Second, extracting multiple observations enables robust regression against noises and other variances (lighting changes etc.).

In the following we first introduce our Gaussian Process model and then define a time-series kernel.

Gaussian Regression Process Suppose we have a training dataset $\mathcal{D} = \{(x_i, y_i), i = 1, ..., n\}$, where x denotes a training sample, i.e., a sequence of detected boxes, and y denotes 3D locations of vehicles, with slightly notation abused. The column vectors for all n samples are aggregated in the design matrix X, and the outputs are collected in the output matrix Y, so we can write $\mathcal{D} = (X, Y)$.

GP model is to learn a latent function $h(x_i)$ that maps the input x_i to the output value. The observed value y_i is different from the function value $h(x_i)$ by additive noises, which is assumed to follows an I.I.D. Gaussian distribution with zero mean and variance σ^2, $\epsilon \sim \mathcal{N}(0, \sigma^2)$. GP model will put a zero mean Gaussian process prior on the latent function $h(\cdot)$:

$$h|X \sim \mathcal{N}(0, K_N) \tag{12}$$

where the covariance matrix K_N is constructed from a kernel function, i.e., $K_N(i,j) = K(x_i, x_j)$. The kernel function $K()$ depends on a small number of parameters θ, e.g., kernel width, to control the smoothness properties. In GP, we also assume a Gaussian likelihood model:

$$Y|h \sim \mathcal{N}(f, \sigma^2 \mathbf{I}) \tag{13}$$

where **I** is an integer matrix. Integrating out h in Eq. (13), we can obtain the marginal likelihood,

$$Y|X, \theta \sim \mathcal{N}(0, K_N + \sigma^2 \mathbf{I}) \tag{14}$$

which is used to train the GP model by finding a maximum w.r.t. the kernel parameters θ, e.g.,σ^2.

With a new input point x_*, prediction is made by conditioning on the observed data y_* and the learned hyper-parameter θ. The distribution of the output value y_* at the new point is then:

$$y_*|x_*, \mathcal{D}, \theta \sim \mathcal{N}(\mu, \Sigma) \tag{15}$$

with,

$$\begin{aligned} \mu &= K_*^T (K_N + \sigma^2 \mathbf{I})^{-1} y^*; & (16) \\ \Sigma &= K_{**} - K_*^T (K_N + \sigma^2 \mathbf{I})^{-1} K_* + \sigma^2; & (17) \end{aligned}$$

where $K_*(i) = K(x_*, x_i)$ is a column vector and $K_{**} = K(x_*, x_*)$.

To define $p(Q|\mathcal{D}, \tau, K, I)$, for every time t of a trajectory, we extract the trajectory segment from the time $t - \delta t$ to $t + \delta t$ as a testing sample, where δt is a small number. Then, we can apply the Eq. (15) to define the probabilistic model for 3D localization,

$$p(Q|\mathcal{D}, \tau, K, \mathcal{I}) \propto \prod_m \left[\frac{1}{|\tau_k^m|} \prod_{x_* \in \tau_k^m} p(Q_{kt}^m | x_*, \mathcal{D}, \theta) \right] \tag{18}$$

where $|\tau_k^m|$ denote the length of the trajectory τ_k^m.

As is clear from the definition in Eq (15), the standard GP is a non-parametric model and all the training samples are explicitly required at test time in order to construct the predictive distribution. It is infeasible to apply GP while the number of training samples is large. In this work, we use the method by Snelson et al. [43] [44], which tried to learn a small number (10-20) of pseudo training samples by minimizing the likelihood function Eq. (14).

Time-series Kernel The key problem left so far is how to define the kernel function $K(x_i, x_j)$ where x_i and x_j are two sequences of vehicle boxes in images.

We define the kernel function $K(x_i, x_j)$ as the product of two kernel distances in terms of both appearance and time. For a training sample x_i, let t_i denote a time-stamp , and f_{it} denote the corresponding visual feature of the vehicle box at time t_i. The time-series kernel is defined as,

$$K(x_i, x_j) = \frac{1}{|x_i| \cdot |x_j|} \sum_{(f_{it}, t_i) \in x_i} \sum_{(f_{jt}, t_j) \in x_j} \phi(f_{it}, f_{jt}) \varphi(t_i, t_j) \tag{19}$$

where $\phi()$ and $\varphi()$ are two squared exponential kernels

$$\varphi(t_i, t_j) = \kappa \exp(-\frac{(t_i - t_j)^2}{2\sigma_t^2}) \tag{20}$$

κ and σ_t are two kernel parameters in θ which can be learned by maximizing the marginal likelihood (14). The above time series kernel acts like a sequential convolution filter and takes both appearance and time proximities into consideration. We will specify the visual features in Section 4.

3.2.3 *Probabilistic Model For Localizing Connected Vehicles*

With V2V traffic data, we can obtain the absolute position of every connected vehicle. However, V2V locations are noisy and only sparsely available (per a few seconds). Therefore, instead of using the raw V2V positions, we aim to solve the location of connected vehicles in the relative world coordinate. In particular, we use the V2V data to measure the spatial displacement between two connected vehicles and enforce the same displacements between the relative locations to solve. We define the probabilistic model over V2V data as ,

$$p(R|\mathcal{V}, \tau, K, \mathcal{I}) \propto \exp\{-\frac{2}{M(M-1)} \sum_{m \neq n} \mathcal{L}(R_t^m - R_t^n, V_{mnt})\} \tag{21}$$

where V_{mnt} is the displacement between the vehicles m and n at time t, $\mathcal{L}(\cdot, \cdot)$returns the Euclidean distance between two vectors. It is worthy noting that, although the relative world coordinates can be scaled up to the world coordinate after calibrating cameras, in V3I, we only care about the relative 3D positions between vehicles.

3.2.4 *Probabilistic Model For Spatial Relationships*

We use a set of spatial relationships to describe the relative locations of vehicles at the time t:

- Left-right, denoted as $(m, i, j, t) \in \mathcal{L}$ where the vehicle i locates on the left of the vehicle j, both of which appear in the camera m;
- front-back, denoted as $(m, i, j, t) \in \mathcal{F}$ where the vehicle i locates in front of the vehicle j ;
- further-closer, denoted as $(m, i, j, t) \in \mathcal{A}$ where the isolated vehicle i is closer to the camera than the vehicle j, and
- equal position, denoted as $(m, n, i, j, t) \in \mathcal{E}$ where the vehicle i in camera m appear in the camera n with index j, and the two vehicles should have the same 3D locations.

Among these relationships, the former three are extracted from a single camera while the last one is from across cameras. These binary information, although not accurate, can provide discriminative constraints over the desired 3D vehicle locations.

We collect these pair-wise relationships from the results of vehicle detection and cross-view matching. Taking further-closer for instance, we can simply compare the size of two detected boxes to determine which one is further to the camera. Expressing the relationships in binary form enable the feasibility of a simple threshold based method. We denote all the relationships by $\mathcal{C} = \mathcal{L} \cup \mathcal{F} \cup \mathcal{A} \cup \mathcal{E}$. For each relationship $C_k \in \mathcal{C}$, we check if it still holds by the current estimation of 3D trajectories Y, which is straightforward since the moving direction is available as well. We denote the validity by $\mathcal{H}(Y, C_k)$ which returns 0 if the relationship holds; 1 otherwise. Thus, the likelihood model conditioning on C is defined as,

$$p(Y|\mathcal{C}, \tau, K, \mathcal{I}) \propto \exp\{-\gamma \sum_{C_k \in C} \mathcal{H}(Y, C_k)\} \tag{22}$$

where γ is a constant parameter. This model $p(Y|C, \tau, K, I)$ is maximized while all the relationships in C hold in Y.

4. INFERENCE

In this section, we first introduce the proposed inference algorithm and then elaborate the implementations of our V3I system.

4.1 Hybrid Monte Carlo for V3I-STAL

Given multiple monocular video sequences and V2V traffic data, our inference algorithm aims to find the optimal solution W by maximizing a posterior $p(W|I)$. This is an intractable problem [46]. As an approximate optimization technique, Markov Chain Monte Carlo (MCMC) algorithm [46, 33, 30] can search the solution space by simulating a Marko Chain. However, existing MCMC methods are restricted to two aspects: i) the mixing rate is slow particularly for the joint solution space which is the case in this work; and ii) the search usually got stuck in local minimal without proper parameter setting in practice.

To address the above issues, we develop a hybrid Monte Carlo algorithm that can search over the joint solution space efficiently. Our algorithm starts with an initial solution of W that could be greedily or randomly obtained. Then, we design a set of dynamics to reconfigure the current solution which simulates a Markov Chain walking toward the target posterior distribution $p(W|\mathcal{O})$. The dynamics are either jump moves, e.g. creating a new trajectory, or diffusion moves, e.g. adding a bounding box into an existing trajectory. These stochastic dynamics are paired with each other to make the solution changes reversible to guarantee convergence to $p(W|I)$. Formally, a dynamic is proposed to drive the solution state from W to W', which is accepted with the probability in the Metropolis-hasting form [46]: $\min(1, \frac{p(W'|\mathcal{O})q(W \to W')}{p(W|\mathcal{O})q(W' \to W)})$, where $q(W \to W')$ is the proposal probability.

Algorithm 1 summarizes the proposed inference algorithm. In this work, we use two dynamics, including a dynamic in the discrete space and a dynamic in the continuous space.

Dynamic-I: Cluster Sampling for 2D tracking This dynamic is used to add, remove or change existing 2D trajectories observed by individual cameras, i.e., τ. We unify all these operations through a cluster sampling procedure with a adjacent graph. Figure 4 illustrates four such graphs where the y-direction indicates the time-step. We detect a set of vehicle boxes, illustrated as blobs, in every frame and use these boxes as graph nodes to construct the adjacent graph. A graph node is only linked to the spatially coherent nodes in the consecutive frames. With these adjacent graphs, the goal of this dynamic is to color these graph nodes such that nodes belonging to the same trajectory have the same color.

We develop a cluster sampling method following the previous Sweden-Wang Cut algorithm [46, 6], including three major steps. First, for every edge $e =< i, j >$, we compute a local probability, denoted as q_e, to indicate how likely the two linked nodes belong to the same trajectory and thus should be assigned with the same color. We set the edge probability using the feature similarity,

$$q_e = \exp\{-\mathcal{L}(f_i, f_j)\} \tag{23}$$

where f_i and f_j are feature vectors of the two bounding boxes. We normalize q_e so that $\sum_{e=<i,\cdot>} q_e = 1$ where $< i, \cdot >$ represents all edges that links the node i to the next frame. Second, we turn off edges at random according to q_e which leads to a set of connected components (CCPs) of graph nodes. Last, we choose one of the CCPs and assign its nodes to a new color, which results in three atomic changes, as Fig. 4 illustrates, i) birth of a new trajectory: $A \to C$; ii) Merge, $C \to A$; iii) Switch: $A \to B$ or $B \to D$.

The proposal probability for the Dynamic I is calculated as $q(W \to W') = \prod_{e \in Cut} q_e$ where Cut denotes the edges that are turned off around the selected CCP.

At each iteration step, the Dynamic I will flip the colors of a cluster of nodes, instead of a single node, like the traditional single-site sampler, e.g. Gibbs [46]. Similar idea has been used for image segmentation [46] and 3D scene reconstruction [30].

Dynamic-II: Hamiltonian dynamics For 3D localization We develop a proposal-making scheme for the continuous parts of W, i.e., the 3D trajectories Y, with the Hamiltonian Dynamics [4] [1].

Hamiltonian Dynamics can be used to draw random samples from a probability distribution with continuous variables for which direct sampling is difficult. The key is to define a Hamiltonian function in terms of the target distribution from and a kinetic energy term parameterized with auxiliary Momentum variables. Then, we can alternate simple updates for those momentum variables with Metropolis updates. A state proposed in this way can be distant from the current state but have a high probability of acceptance. In comparisons to the random-walk proposal schemes [46] that explore the state space slowly, Hamiltonian Dynamics can converge a lot faster to the target distribution.

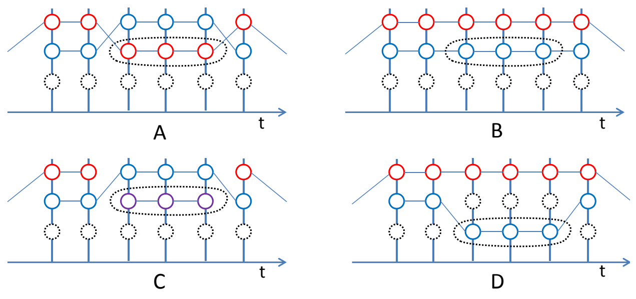

Figure 4: Four atomic states of the discrete tracking solution. There are three atomic changes, i) birth of a new trajectory: $A \to C$; ii) Merge, $C \to A$; iii) Switch: $A \to B$ or $B \to D$. Dotted circles indicate the selected CCPs.

In this work, we aim to sample a joint distribution and use the Hamiltonian dynamics to make proposals in the continuous space. Formally, let $E(Y) = -\log P(W|\mathcal{O})$ denote the target energy function, i.e., the negative log-probability. Consider Y as a position at the energy landscape, g as the momentum at a time t. Sampling Y is equal to moving Y through the energy landscape with a varying moment g. The energy $H(Y, g)$ at a time-step is a combination of the energy $E(Y)$ and the kinetic energy $K(g)$, i.e. $H(Y, g) = E(Y) + K(g)$. We set $K(g) = g^T g/2$ as conventional.

Algorithm 1 Inference .

```
1: Input: $M$ monocular video sequences; V2V data
   Output: 2D and 3D trajectories of vehicles;
2: Initialize the solution representation $W$;
3: Iterate until convergence,
       - Randomly select one of the two dynamics;
       - Make proposals accordingly to reconfigure the current
         solution;
       - Accept the change with a probability
```

The partial derivatives of $H(Y, g)$ determine how Y and g change over time, according to the Hamilton's equations [1]:

$$\frac{\partial Y}{\partial t} = \frac{\partial K(g)}{\partial g} = g, \tag{24}$$

$$\frac{\partial g}{\partial t} = -\frac{\partial E(Y)}{\partial Y} \tag{25}$$

Starting with initial state at time t=0, we can iteratively compute the states of Y and the moment g at each time t, following the Euler's method:

$$Y^{t+1} = Y^t + \alpha g^t \tag{26}$$

$$g^{t+1} = g^t - \alpha \frac{\partial E(Y)}{\partial Y} \tag{27}$$

where α is a constant.

The proposal probability for the Hamiltonian Dynamic is defined over the energy changes after L times updates. Let (Y, g) denote the initial state, (Y^*, g^*) the updated states after L times, we have

$$q(W \to W') = \frac{1}{\mathcal{Z}} \exp\left[H(Y, g) - H(Y^*, g^*)\right] \tag{28}$$

where $\mathcal{Z}$ is a normalization constant . It is worthy noting that we set the evolution times L to be a small number, which allows the other dynamics in our inference algorithm to be done more often to help preserve the detail-balance of the sampling process.

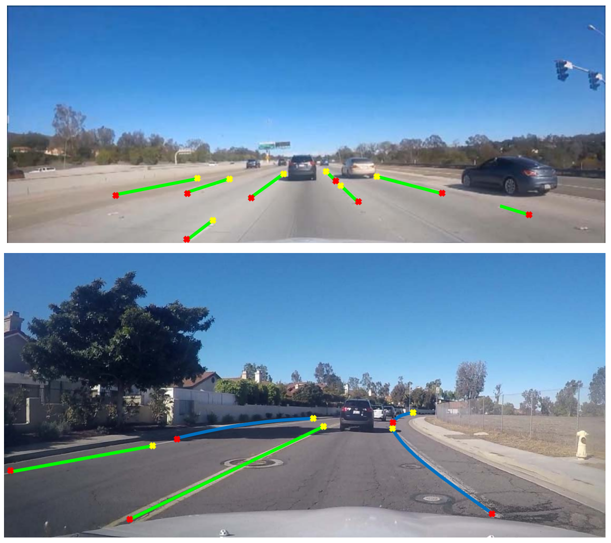

Figure 5: Exemplar results of lane detections. We detected both straight lanes (red) and curved lanes (blue).

4.2 Implementation

Visual Features Instead of using traditional descriptors (e.g. SIFT, HOG) to represent the visual appearances of vehicle images, we employ the powerful Deep Convolution Neural Network (CNN) [23]. With a set of training images of vehicles, we fine tune the CaffeNet [18], which consists of 5 convolutional layers, 2 max-pooling layers, 3 fully-connected layers and a final 1000-dimensional output. The negative training samples are randomly cropped from the video frames in our dataset .

Given an image of vehicle as input, we forward-pass it through the learned DCNN and take the 1000-dimensional output as our appearance descriptor, i.e. f, which is used to define the kernel function $\phi()$ and the edge probability q_e. These descriptors have been proved to function like the Texton filter responses [18] yet with more capabilities of dealing with variances.

Detection of Vehicles and Lanes To detect vehicles in video sequences, we utilize the Region-based Convolutional Neural Network (R-CNN) [14], which includes two steps: making vehicle proposals and classifying proposals. In implementation, we use the Fast R-CNN framework by Girshick [13] that jointly performs proposal classification and bounding box regression by optimizing a multi-task loss function. We use the soft-max functions in the last layer. We use mini-batches of sample size 128. We follow the sampling strategy in the original article. To deal with scale issues, we compute image pyramids which basically augment the training data. This is feasible since our training dataset is relatively small (about 2000 vehicle images). We use the network parameters pre-trained on the VOC 2012 dataset [18] and fine tune it on our training dataset with the same multi-task loss function. The fine-tuning step is very important because that, in V3I system, the camera setting (viewpoints, heights) are completely different from that in the VOC 2012 dataset. Among all the steps of RCNN detector, the bottleneck is the number of region proposals generated for every video frame, which is 64 by default. To accelerate the process, for each vehicle tracked, we predict its position in the new video frames according to its current position and moving speed (in images). We sample a few proposal around the predicted position only. We found that this simple trick can largely reduce the number of proposals while keeping the same detection performance.

In addition, we develop a lane detection algorithm that can process video frames at real-time speed. It includes three major steps: filtering image with Derivative of Gaussian (DOG), Line/Spline Fitting with RANSAC method and Post-processing for lane linking. These algorithms are quite standard and have been discussed in previous efforts, e.g. [2]. We didn't use the top-view of the road [2] because it involves additional computations yet bring minimal gains in our case. Figure. 5 illustrates the two results of lane detection. We will use the estimated lane information to determine if a vehicle stays on the same way over time, as is used in Eq. (10).

Constructing Pair-wise Constraints Given two bounding boxes of vehicles detected in the same video frame, we construct the three pair-wise relationship sets $\mathcal{L}, \mathcal{F}, \mathcal{A}$ as follows. First, we simple compare their horizontal image coordinates to determine which one is on the left hand side. Second, the front-back relationships are derived from the comparisons of their vertical image coordinates. This is feasible because of the cameras are facing forward and moving in parallel directions. Third, we assume the vehicle with larger size box is closer to the camera. Since the above three relationships are all expressed in binary form, instead of relative values (like [10]), they can be robustly extracted even with a simple threshold method.

To obtain the relationships set $\mathcal{E}$, we need to identify the vehicles across cameras. This correspondence problem is traditionally solved with the standard stereo algorithms, e.g. extracting interest points first and applying RANSAC to estimate transform parameters [29]. These algorithms, however, do not work well in V3I, because the viewpoint changes are too big due to the wide baselines between cameras, as is illustrated in Fig. 1. Instead, we utilize a two-step cascade method for detecting such relationships across two cameras. In the first step, given the position of the two connected vehicles, we match the detected lanes across cameras and consider vehicles in different views sharing the same lane as a candidate match. In the second step, we extract the CNN features of each box, calculate their Euclidean distance and prune the candidate matches if their visual distance is beyond than a threshold. We compute pyramid representations for each vehicle image to deal with the viewpoint variances. This simple threshold-based method works well in practice because: i) the lane information from both V2V data and lane detectors can largely narrow the search space; ii) the relationships are used as a soft constraints which means that error predictions will not dominate the final results.

The above relationships, once detected, are used to define the posterior distribution $p(W|I)$. In particular, given 3D trajectories W in the present iteration, we need to verify if every single relationship in C holds or not, i.e. the value of $\mathcal{H}(Y, C_k)$ in Eq. (22). For the relationships $\mathcal{L}, \mathcal{F}, \mathcal{A}$, we can obtain the moving directions of the involved vehicles from the 3D trajectories which make the testing straightforward. For the relationships $\mathcal{L}$, we simply check if the two vehicles have the same 3D locations in Y. Note that a short period of video frames will have hundreds of such relationships which makes this term fairly robust against errors and noises.

Stream Video Parsing The proposed STAL algorithm works on a batch of video sequences and we extend it to deal with streaming videos with the sliding window strategy [30]. At each time t, we take as inputs the video frames during $t - \delta w$ to $t + \delta w$ where $2\delta w$ is the window size, and slide the window with a step (e.g. 25 frames) to get the another window. For each window, we run the Algorithm 1 and initialize the solution using the results over the previous window. In particular, we assume the number of trajectories K^m remains the same, and match every existing trajectory to the frames of the new window with the aforementioned cascade matching algorithm. The matched boxes are added into the existing trajectories. In this way, we obtain a good initialization that is close to the optimal solution. Afterwards, we refine the current solution with the two dynamics. This sliding window method can largely reduce the computational cost while achieving the same level of performance.

5. EXPERIMENTS

Dataset To evaluate the proposed V3I approach, we collect a video dataset that includes four groups of video sequences. Each group consists of 4 time-synchronized video sequences captured by 4 cameras on connected vehicles. Figure 6 illustrates the driving route while shooting the group-1, where the star indicates the starting point. Figures 1 and 2 show video frames from the group-1 and group-2, respectively. These vehicles were running on a freeway at about 60 miles per hour. Each video lasts 10-20 minutes. Since the videos are captured during the peak traffic hours in the morning, there are continuous stream of isolated vehicles appearing in the videos. These video sequences are with many challenges, e.g., lighting changes, motion blur, and work as a challenging benchmark for evaluating V3I algorithms.

We develop an interactive toolkit to manually annotate the 2D and 3D vehicle trajectories for all the 4 groups of video sequences. We use the relative world coordinates. A major feature of the toolkit is that it can interpolate trajectories between the annotated frames in both 2D images and 3D map so that the user does not need to annotate every single frame. A similar system was developed by Vondrick et al. [48], which focused on obtaining 2D trajectories. We also annotate the locations of connected vehicles in videos and use the annotations as V2V data, i.e., $\mathcal{V}$. It takes about 3 hours for a user to annotate a video sequence of 20 minutes that includes hundreds of vehicles.

We divide each video sequence into two even parts and use them for training and testing purposes, respectively.

Learning and Parameters To obtain the hyper-parameters of the time-series kernel $K(\cdot)$, we optimize the marginal likelihood function (14) using the alternate gradient descent algorithm. Following [44], we revise Eq. (14) to learn a few pseudo-inputs to reduce complexity. We set the number of pseudo-inputs to be 20.

The implementation of Algorithm 1 can be found in Section. 4. We use the maximal likelihood method (MLE) [46] to estimate the model parameters in $p(W|I)$, including λ, β, and $\mathcal{Z}$, and fix them throughout the experiments. We set the size of the sliding window to be 60 frames, and sliding step to be 10 frames. The parameters for the vehicle detectors and lane detectors remains the same as in their respective articles.

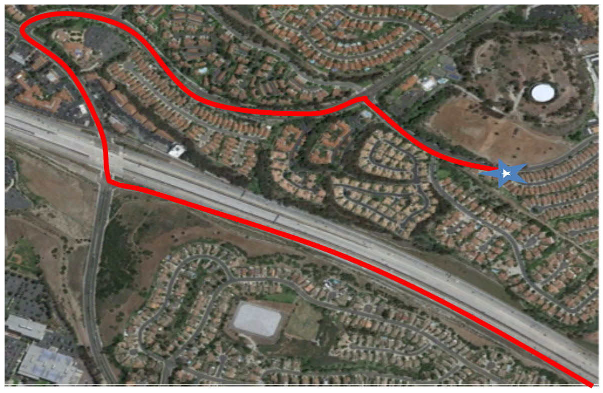

Figure 6: Driving routine of the video sequences in group-1. Blue star indicates the starting point.

Variants of Our Approach and Baselines We implement three variants of the proposed approach to analyze the benefits of individual components. i) STAL-1, that optimizes the probability models for 2D trajectories and uses Dynamic-I for inference. This is an advanced version of the MCMCDA [33] algorithm. ii) STAL-2: that implements Algorithm 1 without the pair-wise spatial relationship $\mathcal{C}$. We simply set the the probability $p(Y|\mathcal{C}, \tau, K, I)$ to be a constant. iii) STAL-3, that is a full implementation of Algorithm 1.

We compare the proposed approach with 4 recent state-of-the-art trackers: 1) Geiger et al. [12], that follows the track-by-detection pipeline and uses the Kalman Filtering algorithm; 2) Lenz et al. [27], a min-cost flow tracking algorithm; 3) Xiang et al. [52] that tracks objects with multiple Gaussian Decision Processes. 4) Andriyenko et al. [3] that proposed a discrete-continuous optimization for 2D tracking. We use the source codes provided by the authors and properly train their models on our dataset according to their descriptions.

We implement baseline methods for localization as follows. First, we apply each of the above baseline trackers and STAL-1 on the input videos to get 2D trajectories, i.e., τ. Then, with fixed τ, we run the Algorithm 1 with the Dynamic-II only. This leads to five baseline algorithms that solve tracking and localization in turn.

Metrics For *tracking*, we employ the widely used CLEAR met-

Sequences (#vehicles)	Method	MOTA(%)	MOTP(%)	MT(%)	PT(%)	ML(%)	IDSW	FRAG
	STAL-3	**67.15**	**71.23**	**47.59**	**78.67**	**8.92**	**181**	**190**
	STAL-2	63.27	68.32	45.32	73.56	9.34	210	205
	STAL-1	56.15	60.94	39.29	67.21	11.52	235	316
Group 1 (335)	Geiger et al. [12]	59.34	63.34	42.36	69.34	13.46	432	275
	Lenz et al. [27]	60.21	61.12	39.15	68.98	11.97	614	243
	Xiang et al. [52]	58.76	63.27	36.27	64.32	19.24	543	259
	Andriyenko et al. [3]	48.13	56.32	35.37	45.61	20.34	781	345
	STAL-3	**62.94**	**67.83**	**46.27**	**68.31**	**10.21**	**276**	**215**
	STAL-2	58.17	61.09	38.49	62.75	13.45	324	229
	STAL-1	55.64	54.36	37.23	58.16	15.18	350	417
Group 2 (362)	Geiger et al. [12]	56.72	55.68	40.34	57.32	16.29	613	439
	Lenz et al. [27]	54.21	57.13	39.51	60.31	14.53	581	352
	Xiang et al. [52]	53.76	56.19	38.14	61.29	17.59	462	533
	Andriyenko et al. [3]	45.36	48.32	27.89	52.34	21.69	684	726
	STAL-3	**52.16**	**62.84**	**47.68**	**73.14**	**12.37**	**324**	**236**
	STAL-2	48.56	59.12	42.75	68.75	14.35	392	397
	STAL-1	43.29	53.87	40.50	64.21	18.57	421	531
Group 3 (274)	Geiger et al. [12]	47.19	55.12	41.29	63.59	15.66	382	689
	Lenz et al. [27]	46.36	57.27	43.57	65.82	16.14	475	725
	Xiang et al. [52]	42.19	58.34	45.38	61.32	16.34	419	963
	Andriyenko et al. [3]	37.35	49.25	36.21	58.35	21.34	531	1236
	STAL-3	**65.72**	**73.59**	**52.76**	**75.96**	**8.16**	**289**	**210**
	STAL-2	61.78	67.31	51.34	68.34	9.34	395	256
	STAL-1	53.29	62.15	46.12	66.72	11.63	441	443
Group 4 (281)	Geiger et al. [12]	59.42	64.37	48.52	62.36	9.47	476	369
	Lenz et al. [27]	58.51	65.92	50.36	67.82	12.24	497	479
	Xiang et al. [52]	54.97	59.46	49.21	63.49	13.43	531	412
	Andriyenko et al. [3]	49.23	45.29	45.97	52.75	16.53	724	567

Table 1: Results of tracking. STAL-1, STAL-2, STAL-3 are three implementations of the proposed approach. For every group of sequences, The total number of objects are listed in the first column. See text for detailed explanations.

Table 2: Results of 3D localizations (average errors in meters).

Methods	Group-1	Group-2	Group-3	Group-4
STAL-3	**1.24**	**1.72**	**2.03**	**1.16**
STAL-2	1.56	2.14	2.56	1.56
STAL-1	2.03	3.25	2.97	2.79
Geiger et al. [12]	2.34	2.76	2.67	2.53
Lenz et al. [27]	2.21	2.85	2.86	2.13
Xiang et al. [52]	2.17	3.02	3.10	2.24
Andriyenko et al. [3]	3.16	3.75	2.85	3.17
GP	3.67	3.46	3.21	3.24
SVR [41]	5.32	6.32	4.31	5.29

rics [21]: Tracking Accuracy (MOTA) and Tracking Precision (MOTP) to measure three kinds of errors in tracking: false positives, false negatives and identity switches. We also report the percentages of *mostly tracked* (MT), *partly tracked* (PT) and *mostly lost* (ML) groundtruth (referring to [28]), as well as the numbers of identity switches (IDSW) and fragments (FRAG). Hit/miss for the assignment of tracking output to groundtruth is set to a threshold of Intersection over Union (IoU) ratio 50%. For *localization*, we calculate the average localization errors: the Euclidean distances between the estimated coordinates and groundtruth coordinates. We calculate localization errors in meters facilitate interpretation. Note that we only count the mostly tracked vehicles, i.e. the ratio of groundtruth trajectories that are covered by a track hypothesis for at least 80% of their respective life space.

Efficiency Our approach is implemented in MATLAB and runs on a desktop with Intel I7 3.0GHz CPU, 32GB memory and Nvidia K40 GPU. Given 4 sequences of 1080P, the runtime on average is 15-20 fps for object detection, 1000-1500fps for pre-processing, and $10-20$ fps for running the hybrid Monte Carlo method. Overall, the proposed algorithm obtains $10-12$ fps, which is dependent on the vehicle density of the sequence. With proper code migration and optimization, e.g., batch processing, our algorithm can achieve real-time processing.

Results Table 1 reports the tracking results of the various methods. Results over individual groups of videos are listed separately. From the results, we can obtain the following observations. First, the proposed joint tracking and localization methods, i.e., STAL-3 and STAL-2, outperformed the other baseline trackers, which demonstrates the superiority of the proposed joint framework. Second, the comparisons between STAL-3 and STAL-2 further suggest that it is beneficial to include the pair-wise spatial relationships between vehicles. In Figures 1 and 2, we plot the generated scene maps for the group-1 and group-2, respectively.

Table 2 reports the localization results over individual groups of videos. Among these methods, STAL-3 and STAL-2 can directly output 3D locations. The other methods solve tracking and localization in a subsequent fashion. All methods are able to access the V2V traffic data and integrate the outputs of Gaussian Regression Process (GP). We include the results of GP for comparisons as well. We also report the regression results by applying the SVR method over the data $\mathcal{D}$. From the table, we can observe that STAL-3 achieved the minimal localization error. While all the methods can certainly improve the GP results, it is evident that jointly formulating the tracking and localization can generate better localization result, which is consistent with our observations on tracking results.

6. CONCLUSIONS

This paper studies a novel data fusion problem, visual vehicle-to-vehicle interaction (V3I). We developed a unified probabilistic approach to simultaneously track and localize isolated vehicles through cameras on connected vehicles. Our formula integrates a Gaussian Regression Process with time-series kernel, and a set of pair-wise binary relationships in the Bayesian Framework. For inference, we introduced a fast Hybrid Monte Carlo algorithm to efficiently search the joint solution space. Extensive experiments with component analysis clearly demonstrated the effectiveness of the proposal approach.

7. ACKNOWLEDGMENT

The data used in this work was collected and annotated by the Computer Vision Lab at San Diego State University (SDSU). The author would like to thank Dr. Xianfeng Yang (SDSU) for his insightful discussions and suggestions. The original idea of this work was initialized and discussed when the author was traveling in Boston with his friends: Yibiao Zhao (MIT), Yixin Zhu (UCLA) and Tianmin Shu (UCLA).

8. REFERENCES

[1] A. Almeida. Hamiltonian systems: Chaos and quantization. *Cambridge monographs on mathematical physics*, 1992.

[2] M. Aly. Real time detection of lane markers in urban streets. In *IEEE Intelligent Vehicles Symposium*, 2008.

[3] A. Andriyenko and K. Schindler. Discrete-continuous optimization for multi-target tracking. In *Proc. CVPR*, 2012.

[4] M. Audin and D. Babbitt. Hamiltonian systems and their integrability. *American Mathmatical Society*, 2008.

[5] H. Azizpour and I. Laptev. Object detection using strongly supervised deformable part models. In *ECCV*, 2012.

[6] A. Barbu and S. Zhu. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *IEEE Trans. PAMI*, 27(8):1239–1253, 2007.

[7] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. multiple object tracking using k-shortest paths optimization. *IEEE Trans. PAMI*, 33(9):1806–1819, 2011.

[8] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2):81–227, 2012.

[9] M. Everingham, S. Eslami, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015.

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 32(9):1627–1645, 2010.

[11] P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. PAMI*, 32(9):1627–1645, 2010.

[12] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3d traffic scene understanding from movable platforms. *IEEE Trans. PAMI*, 2014.

[13] R. Girshick. Fast r-cnn. In *CVPR*, 2015.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[15] C. Haene, N. Savinov, and M. Pollefeys. Class specific 3d object shape priors using surface normals. In *CVPR*, 2014.

[16] M. Hejrati and D. Ramanan. Analysis by synthesis: Object recognition by object reconstruction. In *CVPR*, 2014.

[17] C. Huang, Y. Li, and R. Nevatia. Multiple target tracking by learning-based hierarchical association of detection responses. *IEEE Trans. PAMI*, 35(4):898–910, 2013.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding authors. In *Proc. ACM Multimedia*, 2014.

[19] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Gaussian processes for object categorization. *IJCV*, 2009.

[20] N. Karlsson, E. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich. The vslam algorithm for robust localization and mapping. In *ICRA*, 2005.

[21] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and j.. Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Trans. PAMI*, 31(2):319–336, 2009.

[22] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proc. ECCV*, 2006.

[23] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012.

[24] A. Kundu, Y. Li, F. Daellert, F. Li, and J. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *ECCV*, 2014.

[25] L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014.

[26] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: the informative vector machine. In *NIPS*, 2003.

[27] P. Lenz, A. Geiger, and R. Urtasun. Followme: Efficient online min-cost flow tracking with bounded memory and computation. In *ICCV*, 2015.

[28] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *Proc. CVPR*, 2009.

[29] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman. Sift flow: Dense correspondence across different scenes. In *Proc. ECCV*, 2008.

[30] X. Liu, L. Lin, and H. Jin. Contextualized trajectory parsing via spatio-temporal graph. *IEEE Trans. PAMI*, 35(12):3010–3024, 2013.

[31] X. Liu, Y. Zhao, and S. Zhu. Single-view 3d scene parsing by attributed grammar. In *CVPR*, 2014.

[32] L. C. o and M. Opper. Sparse online gaussian processes. *Neural Computation*, (14):641–668, 2002.

[33] S. Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Trans. PAMI*, 54(3):481–498, 2009.

[34] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automatic Control*, 24(6):843–854, 1979.

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. NIPS*, 2015.

[36] J. Roecker. A class of near optimal jpda algorithms. *IEEE Trans.Aerosp. Electron. Syst.*, 30(2):504–510, 1994.

[37] J. Roecker and G. Phillis. Suboptimal joint probabilistic data association. *IEEE Trans. Aerosp. Electron. Syst.*, 29(2):510–517, 1993.

[38] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proc. CVPR*, 2013.

[39] A. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. In *ICCV*, 2013.

[40] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Trans. PAMI*, 36(7):1442–1468, 2014.

[41] A. Smola and V. Vapnik. Support vector regression machines. In *NIPS*, 1997.

[42] A. J. Smola and P. Bartlett. Sparse greedy gaussian process regression. In *NIPS*, 2001.

[43] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *NIPS*, 2006.

[44] E. Snelson and Z. Ghahramani. Variable noise and dimensionality reduction for sparse gaussian processes. In *UAI*, 2006.

[45] S. Song and M. Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *CVPR*, 2014.

[46] Z. Tu and S. Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Trans. PAMI*, 24(5):657–673, 2002.

[47] P. A. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, 2006.

[48] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2012.

[49] C.-C. Wang. Simultaneous localization and mapping with detection and tracking of moving objects. In *ICRA*, 2002.

[50] T. Willke, P. Tientrakool, and N. Maxemchuk. A survey of inter-vehicle communication protocols and their applications. *IEEE Communications Surveys*, 2009.

[51] Y. Wu, S. Zhu, and C. Guo. From information scaling of natural images to regimes of statistical models. *Quarterly of Applied Mathematics*, 66(1):81–122, 2008.

[52] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi- object tracking by decision making. In *ICCV*, 2015.

[53] X. Yang, L. Liu, N. Vaidya, and F. Zhao. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *MOBIQUITOUS*, 2004.

[54] Q. Yu, G. Medioni, and I. Cohen. Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *Proc. CVPR*, 2007.

[55] H. Zhang, A. Geiger, and R. Urtasun. Understanding high-level semantics by modeling traffic patterns. In *ICCV*, 2013.

[56] S. Zhang, X. Yu, Y. Sui, S. Zhao, and L. Zhang. object tracking with multi-view support vector machines. *IEEE Trans. MM*, 17(3):265–278, 2015.

[57] X. Zhang, Y.-H. Yang, Z. Han, H. Wang, and C. Gao. Object class detection: A survey. *ACM Comput. Surv.*, 46(1), 2013.