Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features

Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, Jenq-Neng Hwang
Department of Electrical Engineering, University of Washington (UW)
Seattle, WA 98195, USA
{zhtang, gaoang, alexinuw, azheng, hwang}@uw.edu

Abstract

Tracking of vehicles across multiple cameras with non-overlapping views has been a challenging task for the intelligent transportation system (ITS). It is mainly because of high similarity among vehicle models, frequent occlusion, large variation in different viewing perspectives and low video resolution. In this work, we propose a fusion of visual and semantic features for both single-camera tracking (SCT) and inter-camera tracking (ICT). Specifically, a histogram-based adaptive appearance model is introduced to learn long-term history of visual features for each vehicle target. Besides, semantic features including trajectory smoothness, velocity change and temporal information are incorporated into a bottom-up clustering strategy for data association in each single camera view. Across different camera views, we also exploit other information, such as deep learning features, detected license plate features and detected car types, for vehicle re-identification. Additionally, evolutionary optimization is applied to camera calibration for reliable 3D speed estimation. Our algorithm achieves the top performance in both 3D speed estimation and vehicle re-identification at the NVIDIA AI City Challenge 2018.

1. Introduction

Among the studies in the *Intelligent Transportation System* (ITS), video analytics with data captured by multiple cameras have been of high significance for many applications, e.g., the estimation of traffic flow characteristics, anomaly detection, multi-camera tracking, etc. However, different from the majority of works in *Multiple Object Tracking* (MOT) that focus on human objects, tracking of vehicles in urban environments is much more challenging due to several reasons. First, because of the limited number of car models, the appearance similarity among vehicles is generally higher than humans. Second, in a busy flow of traffic especially at traffic intersections, many vehicle objects are occluded, which will cause severe identity switches. Last but not least, the viewpoints of the same car in two different cameras can vary largely.

Fig. 1. Object detection, SCT with 3D speed estimation and multi-camera tracking. The detected vehicles and their car types are shown on *top left*. The trajectories of objects are shown on *top right* with the estimated speed in mi/h plotted above each bounding box. At the *bottom*, a pair of identified vehicles in two camera views are shown in *red bounding boxes*.

In *Single-Camera Tracking* (SCT), the problem of vehicle tracking for 3D real world speed estimation (in terms of mi/h, not pix/sec) remains challenging. Some propose to utilize traditional approaches for MOT such as Bayesian inference methods [1]. Automatically generated 3D vehicle models are adopted in [2, 3] to address the problem of occlusion. But for long videos with busy traffic flow, a more efficient and reliable descriptor of appearance features is critically needed. Also, the semantic features for data association in vehicle tracking, which has been proven effective in many works of human tracking [4, 5], is of little attention. Finally, the accurate backprojection of vehicle positions into 3D world space is another critical issue to be addressed.

Inter-Camera Tracking (ICT) of vehicles, i.e., vehicle re-identification, can be considered as an instance-level object search task, which is different from vehicle detection, tracking, and categorization problems. Many works [6, 7] focus on the extraction of reliable appearance features, especially attributes learned by *Deep Convolutional Neural Networks* (DCNNs). Additionally, license plate recognition [8] and spatio-temporal optimization [6] are commonly used to resolve confusion between vehicles, as there are large number of similar cars in urban surveillance. But all the mentioned approaches can only perform well in fine-grained vehicle categorization and verification, which may not be suitable for vehicles captured in low resolution.

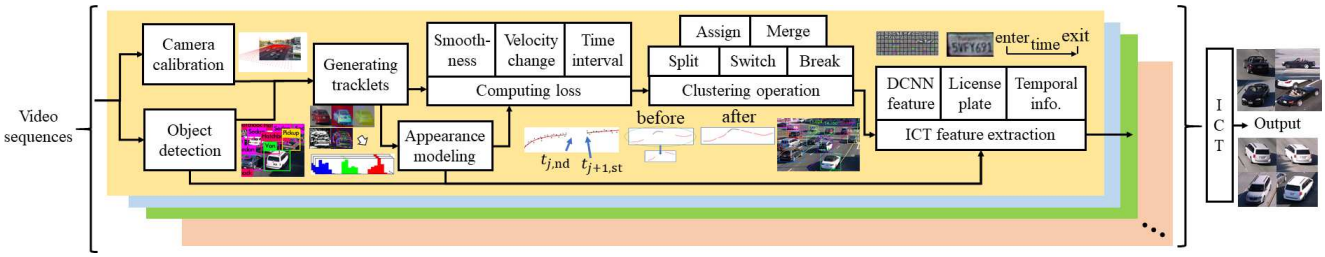


Fig. 2. Flow diagram of the proposed multi-camera tracking framework, where each *color* represents the processing flow of a camera.

In this paper, we propose an innovative framework for both SCT and ICT. A fusion of visual and semantic features is exploited for SCT, including a histogram-based adaptive appearance model designed to learn long-term appearance change. The computed loss function is applied to a bottom-up clustering scheme for the association of tracklets. Similarly, we also employ a fusion of various features for vehicle re-identification across cameras. Finally, an evolutionary algorithm is introduced to optimize camera parameters for 2D-to-3D backprojection, resulting in reliable 3D speed estimation of tracked vehicles in SCT.

The rest of this paper is organized as follows. The related works of our approach are reviewed in more details in Section 2. The system overview and description of each algorithmic component are covered in Section 3. In Section 4, we introduce the evaluation of our method on the *NVIDIA AI City Dataset* [9]. Finally, the conclusion is drawn in Section 5.

2. Related works

We follow the tracking-by-detection paradigm for vehicle tracking in each single camera and across multiple cameras. Some related methods are reviewed as follows.

2.1. Single-camera tracking (SCT)

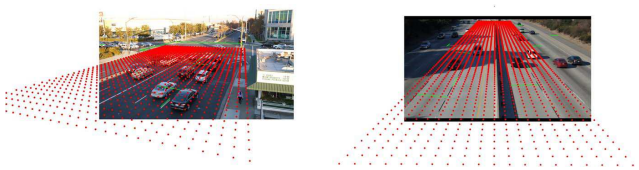
MOT within a single camera has been a challenging field mainly due to noise in object detection, occlusion and similar appearance among nearby objects. Most methods focus on formulating MOT as a data association problem, where many approaches are developed using semantic features of human objects [4, 5], but none is specifically designed for vehicle tracking. Effective modeling of object appearance is also key to the robustness of MOT. To overcome partial occlusion, Chu et al. [10, 11] propose to build multiple spatially weighted kernel histograms with binding constraints to model object appearance. However, for vehicle objects in traffic scenes, the occluded parts are usually not regular due to fast change of viewpoint along car movement. Therefore, Lee et al. [3] make use of 3D deformable models of vehicles to define multiple kernels in 3D space. Tang et al. [12] extend their work and combine kernel-based MOT with camera self-calibration for automatic 2D-to-3D backprojection [13], which is selected as the winning method in *NVIDIA AI City Challenge 2017*. More recently, Sochor et al. [2] introduce a simpler vehicle model using 3D bounding box for fine-grained recognition. The 3D modeling of vehicles can work fine for light traffic flow, where the shape of each car can be well segmented. However, the performance is degraded when the contours of vehicles are attached to each other. In [14], a pixel-based adaptive appearance model is proposed for multiple human tracking, in which a relatively long-term history of appearance change is explicitly encoded in a normalized matrix of pixel models. However, pixel-based appearance model can fail easily for vehicle re-identification, as the variation of viewpoints in two different cameras can contrast significantly.

2.2. Inter-camera tracking (ICT)

Vehicle re-identification is a frontier area with limited research in recent years. Yang et al. [7] propose to apply DCNNs for fine-grained vehicle categorization and model verification. Recently, Liu et al. [6] explore fusion of appearance features, such as texture, color and semantic attributes learned by DCNNs, where low-level and high-level semantic features are integrated for vehicle search. Nevertheless, all these appearance features are extracted from one or few instances of a vehicle, which can be affected by the poor quality in some specific frames. Moreover, these appearance-based methods can hardly distinguish among vehicles of the same model. In [6, 8], video-based license plate recognition and comparison are explored for vehicle re-identification. But they may not work properly when the traffic video resolution is low.

3. Methodology

The overview flow diagram of our proposed framework for SCT and ICT in a camera array is presented in Fig. 2. In each single camera view, we first employ evolutionary algorithm for camera calibration. For object detection, the state-of-the-art YOLOv2 detector [15] is adopted, which is trained on thousands of hand-labeled frames. Taking advantage of the calibrated camera, which allows the detected objects to be backprojected to 3D space, we can perform 3D SCT based on a bottom-up clustering strategy with a fusion of features in loss computation, among which the histogram-based appearance models are learned and used to resolve confusion between nearby targets. Meanwhile, other visual and semantic information, i.e., DCNN features, license plates, temporal information and



(a) (b)

Fig. 3. Visualization of the performance of the proposed evolutionary camera calibration. Projected (uniformly distributed) virtual grids on the ground plane are plotted as *red dots*. The *green solid lines* denote the manually selected line segments used to measure the reprojection error on the ground plane. (a) A scene at Loc4 in [9]. (b) A scene at Loc1 in [9].

detected car types, are also extracted for vehicle re-identification in the ICT task. Each algorithmic component will be elaborated in this section.

3.1. Evolutionary optimization for camera calibration

From each camera view, we first manually label two pairs of vanishing lines, i.e., parallel line pairs on the 3D ground plane that are orthogonal with each other, from which we can derive the two vanishing points on the ground plane, noted V_X and V_Y. It has been proven [16] that all the camera parameters in a 3×4 projection matrix $\mathbf{P}$ can be computed from V_X and V_Y with some constraints on intrinsic camera parameters. To relax these constraints for more accurate estimation of camera parameters, we formulate an optimization problem to minimize the reprojection error. A set of N_{ls} line segments on the ground plane, each defined by two endpoints, noted $\{\overline{P_k Q_k}\}$, are manually selected, whose ground-truth 3D lengths are measured in the Google Maps [17]. Using the calculated camera parameters, the 2D endpoints of the line segments, noted $\{p_k\}$ and $\{q_k\}$, can be backprojected to 3D. Their Euclidean distances represent the estimated 3D lengths of the line segments. The absolute differences between estimations and ground truths are summed up to describe the reprojection error. Thus, the objective of our optimization problem is defined as follows.

$$\min_{\mathbf{P}} \sum_{k=1}^{N_{\text{ls}}} \left| \|P_k - Q_k\|_2 - \left\|\widehat{P_k} - \widehat{Q_k}\right\|_2 \right|,$$
$$\text{s.t. } \mathbf{P} \in \text{Rng}_{\mathbf{P}}, p_k = \mathbf{P} \cdot \widehat{P_k}, q_k = \mathbf{P} \cdot \widehat{Q_k}, \quad (1)$$

where $\{\widehat{P_k}\}$ and $\{\widehat{Q_k}\}$ denote the estimated endpoints of the selected line segments that are backprojected to the 3D ground plane. Additionally, $\text{Rng}_{\mathbf{P}}$ is the initial range for each camera parameter to be optimized.

The non-linear optimization problem in (1) can be iteratively solved by the *Estimation of Distribution Algorithm* (EDA) [18], which is a classic evolutionary optimization algorithm, to optimize the 11 camera parameters in $\mathbf{P}$. This iterative process stops until the mean of the estimated probability density function (pdf) is smaller than a specified threshold, $\epsilon_{\mathbf{P}}$. In Fig. 3, the visualization of our performance on two scenes of the *NVIDIA AI City Dataset* [9] is shown.

3.2. Object detection based on YOLOv2

The provided training dataset, *UA-DETRAC* [20], by the *NVIDIA AI City Challenge 2018* is not adopted by us due to its highly unbalanced data distribution, which gives little information about vehicle categorization. Therefore, we select 4,500 frames uniformly sampled from [9], where each of them contains 5 to 40 objects. The training data are manually labeled in 8 categories, including sedan, hatchback, bus, pickup, minibus, van, truck and motorcycle. The state-of-the-art object detector, YOLOv2 [15], is chosen by us for training and testing. The pretrained weights are used to initialize the network. An example of our qualitative performance can be viewed in Fig. 1.

After the 2D bounding box of each observation is derived from object detection, the optimized $\mathbf{P}$ from evolutionary camera calibration is used to backproject its foot point, i.e., the center of the bottom, to 3D space for speed estimation.

3.3. Loss function for data association

A bottom-up clustering strategy based on a fusion of visual and semantic features is proposed for SCT. First, the detected objects are grouped into tracklets based on spatio-temporal consistency [21]. Then we employ a clustering method to associate tracklets into longer trajectories. The clustering operations are determined by minimizing a loss function measuring the loss in the assignment of tracklets,

$$l = \sum_{i=1}^{n_{\text{v}}} l_i,$$
$$l_i = \lambda_{\text{sm}} l_{i,\text{sm}} + \lambda_{\text{vc}} l_{i,\text{vc}} + \lambda_{\text{ti}} l_{i,\text{ti}} + \lambda_{\text{ac}} l_{i,\text{ac}}, \quad (2)$$

where n_{v} is the number of vehicle identities in a camera, l_i is the clustering loss for the i-th trajectory, $l_{i,\text{sm}}$ is the trajectory smoothness loss, $l_{i,\text{vc}}$ is the velocity change loss, $l_{i,\text{ti}}$ is the time interval loss between adjacent tracklets, and $l_{i,\text{ac}}$ is the appearance change loss. We use λ to denote the regularization parameters of loss functions, which are empirically set as 0.2, 8, 25 and 0.5 respectively.

The smoothness of vehicle trajectories can be represented by Gaussian regression. We denote the x and y coordinates of observed tracklets in 2D as $\tilde{x}(t)$ and $\tilde{y}(t)$ respectively, which can be expressed as

$$\tilde{x}(t) = x(t) + \epsilon_x,$$
$$\tilde{y}(t) = y(t) + \epsilon_y, \quad (3)$$

where $x(t)$ and $y(t)$ are from the real trajectories and $\epsilon_x \sim \mathcal{N}(0, \sigma_x^2)$ and $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ are the Gaussian noise from detection results. Given a new set of frame indices $\mathbf{t}_*$, we want to predict the trajectories $\mathbf{x}_*$ and $\mathbf{y}_*$.

We denote the kernel function as $\kappa(t, t')$. Then the joint density of the observed data and the latent noise-free function on the test time indices are given by

$$\begin{pmatrix} \tilde{\mathbf{x}} \\ \mathbf{x}_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} \mathbf{K}_{\tilde{x}} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right),$$
$$\begin{pmatrix} \tilde{\mathbf{y}} \\ \mathbf{y}_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} \mathbf{K}_{\tilde{y}} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right), \quad (4)$$

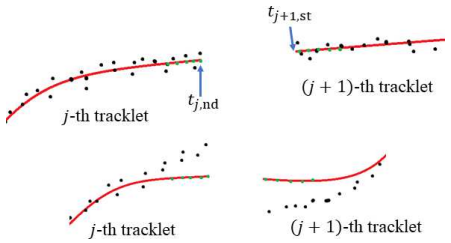


Fig. 4. Examples of smoothness loss between adjacent tracklets. *Top* figures present two tracklets that belong to the same trajectory. *Bottom* figures show two tracklets that do not belong to the same trajectory. *Black dots* show the detected locations at time t, i.e., $\tilde{\mathbf{p}}_j(t)$ and $\tilde{\mathbf{p}}_{j+1}(t)$. *Red curves* represent trajectories from regression, i.e., $\mathbf{p}_{*,j}(t)$ and $\mathbf{p}_{*,j+1}(t)$. *Green dots* show n_k neighboring points on the red curves around the endpoints of the tracklets at $t_{j,\mathrm{nd}}$ and $t_{j+1,\mathrm{st}}$.

where $\mathbf{K}_{\tilde{x}} = \kappa(\mathbf{t},\mathbf{t}) + \sigma_x^2\mathbf{I}$, $\mathbf{K}_{\tilde{y}} = \kappa(\mathbf{t},\mathbf{t}) + \sigma_y^2\mathbf{I}$, $\mathbf{K}_* = \kappa(\mathbf{t},\mathbf{t}_*)$ and $\mathbf{K}_{**} = \kappa(\mathbf{t}_*,\mathbf{t}_*)$. Then the posterior predictive densities can be defined as

$$p(\mathbf{x}_*|\mathbf{t}_*,\mathbf{t},\mathbf{x}) = \mathcal{N}(\mathbf{x}_*|\boldsymbol{\mu}_x,\boldsymbol{\Sigma}_x),$$
$$p(\mathbf{y}_*|\mathbf{t}_*,\mathbf{t},\mathbf{y}) = \mathcal{N}(\mathbf{y}_*|\boldsymbol{\mu}_y,\boldsymbol{\Sigma}_y), \quad (5)$$

where

$$\boldsymbol{\mu}_x = \mathbf{K}_*^T\mathbf{K}_{\tilde{x}}^{-1}\mathbf{x},$$
$$\boldsymbol{\Sigma}_x = \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}_{\tilde{x}}^{-1}\mathbf{K}_*,$$
$$\boldsymbol{\mu}_y = \mathbf{K}_*^T\mathbf{K}_{\tilde{y}}^{-1}\mathbf{x},$$
$$\boldsymbol{\Sigma}_y = \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}_{\tilde{y}}^{-1}\mathbf{K}_*. \quad (6)$$

Hence, we can use $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_y$ as the prediction results given frame indices $\mathbf{t}_*$.

- Smoothness loss

Smoothness loss is defined to measure the smoothness of tracklets which belong to the same trajectory. The tracklets in each trajectory is sorted by the entering time stamps. For each pair of adjacent tracklets in the i-th trajectory, we calculate their distance in the connected region, i.e.,

$$d_{j,\mathrm{sm}} = \sum_{t=t_{j,\mathrm{nd}}-n_\mathrm{k}}^{t_{j,\mathrm{nd}}} \left\| \frac{\tilde{\mathbf{p}}_j(t) - \mathbf{p}_{*,j}(t)}{\left(b_{j,\mathrm{w}}(t)^2 + b_{j,\mathrm{h}}(t)^2\right)^{1/2}} \right\|_2 + \sum_{t=t_{j+1,\mathrm{st}}}^{t_{j+1,\mathrm{st}}+n_\mathrm{k}} \left\| \frac{\tilde{\mathbf{p}}_{j+1}(t) - \mathbf{p}_{*,j+1}(t)}{\left(b_{j+1,\mathrm{w}}(t)^2 + b_{j+1,\mathrm{h}}(t)^2\right)^{1/2}} \right\|_2, \quad (7)$$

where $\tilde{\mathbf{p}}_j(t) = [\tilde{x}(t), \tilde{y}(t)]^T$ is the detected location of the j-th tracklet at time t, $\mathbf{p}_{*,j}(t) = [\mu_x(t), \mu_y(t)]^T$ is the predicted location of the j-th tracklet at time t using Gaussian regression, $b_{j,\mathrm{w}}(t)$ and $b_{j,\mathrm{h}}(t)$ respectively denote the width and height of the detected bounding box, $t_{j,\mathrm{st}}$ and $t_{j,\mathrm{nd}}$ respectively denote the starting and ending frame indices of the j-th tracklet, and $n_\mathrm{k} = 4$ is the number of neighboring points around the endpoints of adjacent tracklets for comparison. Examples are shown in Fig. 4.

The smoothness loss of the i-th trajectory is defined as

$$l_{i,\,\mathrm{sm}} = \sum_{j=1}^{n_i-1} d_{j,\mathrm{sm}}, \quad (8)$$

where n_i is the number of tracklets in the i-th trajectory.

- Velocity change loss

Since the velocity of vehicles cannot change abruptly, we use acceleration to measure the velocity change between two adjacent tracklets. If high acceleration is detected in the connected region, the two tracklets are less likely to hold the same identity. We calculate the maximum acceleration around each endpoint of a tracklet as follows,

$$a_{j,\mathrm{st}} = \max_{t=t_{j,\mathrm{st}}-k,\ldots,t_{j,\mathrm{st}}+n_\mathrm{k}} \left\| \frac{\mathbf{p}_{*,j}(t+1) + \mathbf{p}_{*,j}(t-1) - 2\mathbf{p}_{*,j}(t)}{\left(b_{j,\mathrm{w}}(t)^2 + b_{j,\mathrm{h}}(t)^2\right)^{1/2}} \right\|_2,$$
$$a_{j,\mathrm{nd}} = \max_{t=t_{j,\mathrm{nd}}-n_\mathrm{k},\ldots,t_{j,\mathrm{nd}}+n_\mathrm{k}} \left\| \frac{\mathbf{p}_{*,j}(t+1) + \mathbf{p}_{*,j}(t-1) - 2\mathbf{p}_{*,j}(t)}{\left(b_{j,\mathrm{w}}(t)^2 + b_{j,\mathrm{h}}(t)^2\right)^{1/2}} \right\|_2, \quad (9)$$

Then the velocity change loss is defined as

$$l_{i,\mathrm{vc}} = \sum_{j=1}^{n_i} \left(a_{j,\mathrm{st}} + a_{j,\mathrm{nd}}\right). \quad (10)$$

- Time interval loss

If a pair of adjacent tracklets have long time interval in between, they are less likely to share the same trajectory. The time interval loss is defined based on the difference between two endpoint time stamps, $t_{j+1,\mathrm{st}}$ and $t_{j,\mathrm{nd}}$, of two adjacent tracklets accordingly,

$$l_{i,\mathrm{ti}} = \max_j \left[\left(t_{j+1,\mathrm{st}} - t_{j,\mathrm{nd}}\right)^3 / 10^6 \right]. \quad (11)$$

- Appearance change loss

Reliable description of object appearance is key to the reduction of identity switches, especially for nearby vehicles with similar appearance. We propose a histogram-based adaptive appearance model for the computation of appearance change loss, which is elaborated in Section 3.4.

3.4. Histogram-based adaptive appearance modeling

The appearance model of the j-th tracklet, noted $\mathbf{m}_j$, contains a set of n_m observed concatenated histogram vectors.

$$\mathbf{m}_j = \{m_{1,j}, m_{2,j}, \ldots, m_{n_\mathrm{m},j}\}. \quad (12)$$

In our experiments, we use a combination of RGB color histogram, HSV color histogram, Lab color histogram, *Linear Binary Pattern* (LBP) histogram and *Histogram of Oriented Gradients* (HOG) histogram for feature description. As there are 11 channels with 8 bins each, we have each copy of $m_{k,j} \in \mathbb{R}^{88}$. For each tracklet, we keep n_m copies of continuously updated $\{m_{k,j}\}$ to "memorize" variations of the appearance. The value in each bin is normalized between 0 and 1. An example of feature maps and the corresponding histograms is shown in Fig. 5.

To build and update this appearance model, each cropped object region within the detected bounding box is used to build histograms. When the observation is occluded by other(s), the occluded area is removed from the object region before our building the concatenated histograms of visual features. The pixel values for histogram construction are spatially weighted by Gaussian (kernel) distribution,

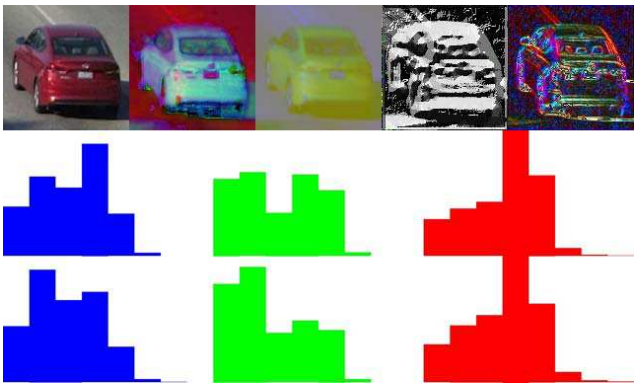


Fig. 5. An example of the construction of an adaptive appearance model. *The first row* respectively presents the RGB, HSV, Lab, LBP and gradient feature maps for an object instance in a tracklet, which are used to build feature histograms. *The second row* shows the original RGB color histograms and *the third row* demonstrates the Gaussian spatially weighted histograms, where the contribution of background area is suppressed.

$$w(p) = \exp\left[-\frac{\|p-p_c\|_2{}^2}{2(w^2+h^2)}\right], \quad (13)$$

where p denotes a pixel location in the visible object region, p_c denotes the center of mass of the visible area, and w and h are the width and height of the object region respectively. The spatial weight $w(p)$ is maximum around p_c where the object usually occupies, which should be emphasized in our feature description. As $w(p)$ decreases when p gets further from p_c, we can suppress the background area.

Since the object instances that are closer to the camera should enjoy more reliable appearance description, the learning rate, noted $\alpha_j(t)$, of $\mathbf{m}_j$ is inversely proportional to the depths of vehicles' 3D foot points, defined as

$$\alpha_j(t) = \frac{\min_{t=t_{j,\mathrm{st}},\ldots,t_{j,\mathrm{nd}}}[D_j(t)]}{D_j(t)}, \quad (14)$$

where $D_j(t)$ denotes the depth. For each tracklet, n_m concatenated histogram vectors extracted from instances with the smallest depths are inserted into $\mathbf{m}_j$. Any other encountered histogram vector can be randomly swapped with an existing element $m_{k,j}$ with probability $\alpha_j(t)$.

Therefore, the appearance change loss in (2) can be defined as

$$l_{i,\mathrm{ac}} = \frac{\sum_{k_0=1}^{n_m}\sum_{k_1=1}^{n_m}\#\left\{\|m_{k_0,j}-m_{k_1,j+1}\|_\mathrm{B} > \epsilon_\mathrm{m}\right\}}{{n_m}^2}, \quad (15)$$

where $\epsilon_\mathrm{m} = 0.3$ is a threshold for histogram distance. The loss in (15) is equivalent to the ratio of histogram vectors from two appearance models that are mismatched. The Bhattacharyya distance is adopted for the measurement of histogram distance.

After a group of tracklets are associated in SCT, their appearance models are merged together based on depth information following similar update scheme in (14). The merged appearance model, noted $\mathbf{m}_i$, is used to describe the appearance change along the entire vehicle trajectory, which will be employed in ICT.

3.5. Optimization by bottom-up clustering

For the j-th tracklet, noted τ_j, we compute the changes of loss, noted Δl_j, for five separate operations (*assign*, *merge*, *split*, *switch* and *break*) and select the operation with the minimum loss-change value, i.e.,

$$\Delta l_j{}^* = \arg\min_{\Delta l_j}\left(\Delta l_{j,\mathrm{as}}, \Delta l_{j,\mathrm{mg}}, \Delta l_{j,\mathrm{sp}}, \Delta l_{j,\mathrm{sw}}, \Delta l_{j,\mathrm{bk}}\right), \quad (16)$$

where $\Delta l_{j,\mathrm{as}}$, $\Delta l_{j,\mathrm{mg}}$, $\Delta l_{j,\mathrm{sp}}$, $\Delta l_{j,\mathrm{sw}}$ and $\Delta l_{j,\mathrm{bk}}$ respectively stand for the changes of loss for *assign*, *merge*, *split*, *switch* and *break* operations. If $\Delta l_j{}^*$ is greater than 0, which means the loss increases after performing the selected operation, no change is made for this tracklet. All tracklets are iteratively clustered into trajectories until convergence. Since the loss decreases or remains unchanged after each selected operation, we are guaranteed to reach convergence.

- Assign operation

The trajectory set of τ_j, noted $S(j)$, is a set of tracklets including τ_j, which belongs to a trajectory. We search through all the trajectory sets and assign τ_j to the trajectory that generates the minimum loss. To be specific, the change of loss after *assign* operation is given by

$$\Delta l_{j,\mathrm{as}} = \min_i\left(l(S(j)\backslash\tau_j) + l(S_i \cup \tau_j)\right) - \left(l(S(j)) + l(S_i)\right). \quad (17)$$

The first term in (17) is the updated loss after *assign* operation while the second term is the original loss.

- Merge operation

For each tracklet, we *merge* its trajectory set $S(j)$ with another trajectory set if lower loss can be obtained. Similarly, the change of loss is calculated as

$$\Delta l_{j,\mathrm{mg}} = \min_i\left(l(S(j) \cup S_i)\right) - \left(l(S(j)) + l(S_i)\right). \quad (18)$$

- Split operation

Split operation is used to *split* a tracklet from the current trajectory set, which becomes an independent trajectory set. The change of loss is defined as

$$\Delta l_{j,\mathrm{sp}} = \left(l(\tau_j) + l(S(j)\backslash\tau_j)\right) - l(S(j)). \quad (19)$$

- Switch operation

For a trajectory set $S(j)$, we denote all the tracklets after τ_j as $S_\mathrm{aft}(j)$ and all others as $S_\mathrm{bef}(j)$. In the *switch* operation, $S_\mathrm{bef}(j)$ and $S_{i,\mathrm{bef}}$ are switched for the calculation of the change of loss,

$$\Delta l_{\mathrm{sw}} = \min_i\left(l(S_\mathrm{bef}(j) \cup S_{i,\mathrm{aft}}) + l(S_\mathrm{aft}(j) \cup S_{i,\mathrm{bef}})\right) - \left(l(S(j)) + l(S_i)\right). \quad (20)$$

- Break operation

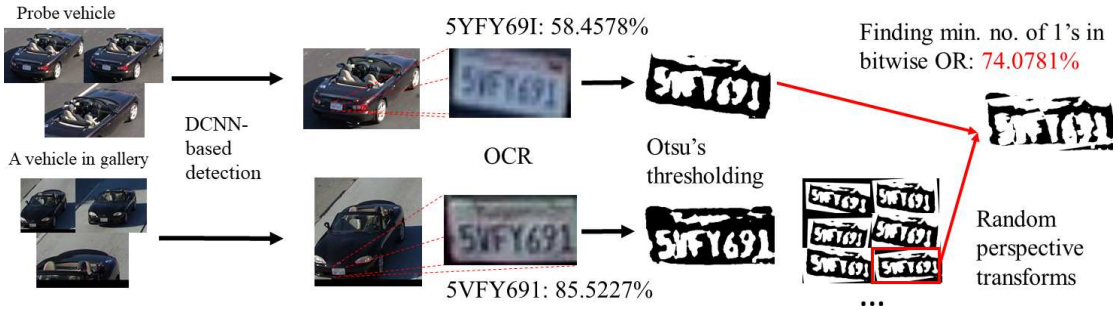


Fig. 6. Demonstration of license plate comparison in low resolution. The confidence score in OCR for the license plate above is considered too low. The segmented binary license plate images are used for calculating the comparison loss.

We split $S(j)$ into $S_{\text{aft}}(j)$ and $S_{\text{bef}}(j)$ for the computation of the change of loss in *break* operation,

$$\Delta l_{\text{bk}} = \left(l\left(S_{\text{bef}}(j)\right) + l\left(S_{\text{aft}}(j)\right)\right) - l\left(S(j)\right). \quad (21)$$

3.6. Feature fusion for ICT

Like SCT, another fusion of visual and semantic features is extracted for ICT. Besides the histogram-based adaptive appearance models for the computation of loss function in SCT, we also exploit DCNN features, detected license plates, detected car types and traveling time information in our feature fusion. The loss function for ICT is defined as

$$L = \sum_{I=1}^{N_{\text{v}}} L_I,$$
$$L_I = L_{I,\text{ac}} \times L_{I,\text{nn}} \times L_{I,\text{lp}} \times L_{I,\text{ct}} \times L_{I,\text{tt}}, \quad (22)$$

where N_{v} is the number of vehicles appeared in all cameras, L_I is the loss for the I-th vehicle, $L_{I,\text{ac}}$ is the appearance change loss, $L_{I,\text{nn}}$ is the matching loss of DCNN features, $L_{I,\text{lp}}$ is the license plate comparison loss, $L_{I,\text{ct}}$ is the mis-classified car type loss, and $L_{I,\text{tt}}$ is the traveling time loss.

In two camera views for ICT, for each probe vehicle trajectory, a trajectory from the gallery with the lowest loss is selected for vehicle re-identification. If the selected loss is considered too high, i.e., over a threshold ϵ_{L}, the vehicle is assigned a new identity.

- Appearance change loss

The computation of appearance change loss in ICT is similar to SCT, except the appearance model merged from all tracklets in a trajectory set is used to describe each vehicle identity from a camera. The definition is as follows,

$$L_{I,\text{ac}} = \frac{\sum_{k_0=1}^{nm}\sum_{k_1=1}^{nm}\#\left\{\left\|m_{k_0,i_\text{p}} - m_{k_1,i_\text{g}}\right\|_B > \epsilon_\text{m}\right\}}{{nm}^2}, \quad (23)$$

where i_{p} and i_{g} are the indices of probe and gallery for I respectively.

- Matching loss of DCNN features

For the extraction of DCNN features, we make use of the *GoogLeNet* [22] model that is fine-tuned on the *CompCars* [7] dataset. From each trajectory within a camera, we select 3 representative views of object instances. A 1024-dim feature vector is extracted from each object instance, noted $f_{k,i}$. Thus, the matching loss of DCNN features is given by Bhattacharyya distance as

$$L_{I,\text{nn}} = \frac{\sum_{k_0=1}^{3}\sum_{k_1=1}^{3}\left\|f_{k_0,i_\text{p}} - f_{k_1,i_\text{g}}\right\|_B}{9}. \quad (24)$$

- License plate comparison loss

The license plates are essential to large-scale vehicle re-identification. However, the resolution of the *NVIDIA AI City Dataset* [9] is not sufficient to support automatic license plate recognition, which brings about a major challenge in ICT. Hence, we propose a license plate comparison scheme for low-resolution images. The process of license plate comparison is demonstrated in Fig. 6.

First, we train another DCNN model [15] to detect the license plate region in each cropped vehicle image. For each trajectory set, 3 representative views of object instances are selected for license plate recognition. The license plate detector is run on cropped vehicle images and the detected region with the highest score is chosen for comparison. Then, all the characters are segmented based on a vertical histogram that finds gaps between the plate characters. An *Optical Character Recognition* (OCR) phase analyzes each character independently, which generates the most possible characters and the confidence. If the confidence scores of two license plates are both above a threshold $\epsilon_{\text{lp}} = 0.8$, the recognized characters are considered correct. The license plate comparison loss is thus calculated as the portion of characters that are mismatched,

$$L_{I,\text{lp}} = \frac{\sum_{k=1}^{\min\left(\left|s_{i_\text{p}}\right|,\left|s_{i_\text{g}}\right|\right)}\left[s_{i_\text{p}}(k) \wedge s_{i_1}(k)\right]}{\min\left(\left|s_{i_\text{p}}\right|,\left|s_{i_\text{g}}\right|\right)}, r_{i_\text{p}} > \epsilon_{\text{lp}} \;\&\; r_{i_\text{g}} > \epsilon_{\text{lp}}, \quad (24)$$

where s_i denotes the string of characters of a license plate and r_i is the corresponding confidence of recognition.

Otherwise, if either of the license plates fails to be recognized properly, which is common for low-quality images, the cropped images of license plate regions are normalized and segmented into binary images by Otsu's method for comparison. Because of potential perspective difference between two license plates, we perform $N_{\text{pt}} = 100$ random perspective transforms on the gallery image. Each transformed image is compared with the probe image

Table 1. Quantitative comparison of speed estimation on the *NVIDIA AI City Dataset* [9]

Rank	Team	S1 Score
1	**Ours**	**1.0000**
2	team79	0.9162
3	team78	0.8892
4	team24	0.8813
5	team12	0.8331
6	team4	0.7924
7	team65	0.7654
8	team6	0.7174
9	team40	0.6564
10	team26	0.6547
11	team18	0.6226
12	team45	0.5953
13	team39	0.0000

Table 2. Quantitative comparison of multi-camera tracking on the *NVIDIA AI City Dataset* [9]

Rank	Team	S3 Score
1	**Ours**	**0.7106**
2	team37	0.2861
3	team79	0.0785
4	team18	0.0074
5	team28	0.0026
6	team41	0.0024
7	team53	0.0002
8	team6	0.0001
9	team10	0.0000
10	team31	0.0000

* Bold entries indicate the rank #1 in each comparison.

by bitwise OR operation. In this scenario, the license plate comparison loss is proportional to the 1's in the combined binary image,

$$L_{I,\mathrm{lp}} = \min_{1,2,\ldots,N_{\mathrm{pt}}} \frac{\sum_{k=1}^{n_{\mathrm{lp}}}\left[g_{i_{\mathrm{p}}}(k) \vee \widetilde{g_{i_{\mathrm{g}}}}(k)\right]}{n_{\mathrm{lp}}}, r_{i_{\mathrm{p}}} \le \epsilon_{\mathrm{lp}} \mid r_{i_{\mathrm{g}}} \le \epsilon_{\mathrm{lp}}, \quad (25)$$

where g_i is the binary image of a license plate and n_{lp} represents the normalized size of license plate images. In each of the N_{pt} iterations, $\widetilde{g_i}$ gives a randomly perspective transform of g_i.

- Mis-classified car type loss

From the output of our DCNN-based object detector [15], the vehicle types in 8 categories can be derived. Along a trajectory set in a video, the majority vote of car type, noted c_i, can be used to inform the vehicle identity. Hence, the mis-classified car type loss can be computed as follows

$$L_{I,\mathrm{ct}} = \begin{cases} 0, & c_{i_{\mathrm{p}}} = c_{i_{\mathrm{g}}} \\ 0.5, & c_{i_{\mathrm{p}}} \neq c_{i_{\mathrm{g}}} \end{cases}. \quad (26)$$

- Traveling time loss

From SCT in 3D space, we can compute the driving speed of each vehicle. From the given driving distance between two cameras, the expected traveling time can thus be estimated. The traveling time loss is designed as a normal distribution around the mean of the expectation,

$$L_{I,\mathrm{tt}} = f\left(t_{i_{\mathrm{g}},\mathrm{st}} - t_{i_{\mathrm{p}},\mathrm{nd}} \mid \frac{d_{i_{\mathrm{p}},i_{\mathrm{g}}}}{v_{i_{\mathrm{p}}}}, \frac{1}{2\pi}\right), \quad (27)$$

in which $f(\cdot)$ is the pdf of normal distribution, $d_{i_{\mathrm{p}},i_{\mathrm{g}}}$ indicates the traveling distance between two cameras and $v_{i_{\mathrm{p}}}$ is the estimated 3D speed. $L_{I,\mathrm{tt}}$ is applied only when there is time overlap between two video sequences.

4. Experimental results

Our proposed method is submitted for evaluation on the *NVIDIA AI City Challenge 2018*, in which we participate in both tracks of 3D speed estimation and multi-camera tracking. Our team achieves the rank #1 in each of the two tracks. The visualization of our qualitative performance in each track is made available at the following link http://allison.ee.washington.edu/thomas/aicity18/. Detailed analyses on our performance are as follows.

4.1. 3D speed estimation

The dataset for traffic flow analysis consists of 27 videos, each 1 minute in length, recorded at 30 fps and 1080p resolution. Our task is to estimate the speed of all vehicles on the main thruways in all frames of all given videos. The ground-truth speed data have been collected via in-vehicle tracking for a subset of cars in each video, which we call ground-truth vehicles. The evaluation is based on the ability to localize these vehicles and predict their speed. The performance evaluation score for this track is computed as

$$S1 = DR \times (1 - NRMSE), \quad (28)$$

where DR is the detection rate and $NRMSE$ is the normalized *Root Mean Square Error* (RMSE) of speed. The $S1$ score ranges between 0 and 1, and higher $S1$ scores are better. DR is computed as the ratio of detected ground-truth vehicles and the total number of ground-truth vehicles. A vehicle is said to be detected if it was localized in at least 30% of frames it appeared in. A vehicle is localized if at least one predicted bounding box exists with *Intersection-Over-Union* (IOU) score of 0.5 or higher relative to the annotated bounding box for the vehicle. $NRMSE$ is the normalized RMSE score across all teams, obtained via min-max normalization given all team submissions.

There are 13 submissions in total for this track. The quantitative comparison of $S1$ scores across all teams is presented in Table 1. Our method achieves perfect DR score, as the association of tracklets based on clustering operations can maximize the utility of detected bounding boxes. We also generate the lowest RMSE (4.0963 mi/h) in speed estimation, as the proposed camera calibration scheme can minimize reprojection error on the ground plane, which leads to robust speed estimation after SCT.

4.2. Multi-camera tracking

The dataset for multi-camera vehicle detection and re-identification contains 15 videos, each around 0.5-1.5 hours long, recorded at 30 fps and 1080p resolution. The task for each team is to identify all vehicles that pass through each of the 4 recorded locations at least once in the given set of videos. This track is evaluated based on tracking accuracy and localization sensitivity for a set of ground-truth vehicles that were driven through all camera locations. Specifically, the evaluation score is computed as

$$S3 = 0.5 \times (TDR + PR), \quad (29)$$

where TDR is the trajectory detection rate and PR is the localization precision. The $S3$ score ranges between 0 and 1, and higher $S3$ scores are better. TDR is the ratio of correctly identified ground-truth vehicle trajectories and the total number of ground-truth vehicle trajectories. PR is the ratio of correctly localized bounding boxes and the total number of predicted boxes across all videos.

There are 10 teams submitted to this track, whose performance is summarized in Table 2. As can be seen, vehicle re-identification under low resolution is such a challenging task that most teams cannot identify even a single vehicle across all 4 locations, which leads to zero TDR and heavily penalized PR. However, with the effective fusion of visual and semantic features in ICT, our proposed method can successfully identify 3 out of 7 ground-truth vehicles with a PR of 0.9925.

5. Conclusion

In this paper, we propose a multi-camera tracking system based on fusion of visual and semantic features. In SCT, the loss function consists of motion, temporal and appearance attributes. Especially, a histogram-based adaptive appearance model is designed to encode long-term appearance change for enhanced robustness. The change of loss is incorporated with a bottom-up clustering strategy for the association of tracklets. Furthermore, the proposed appearance model together with DCNN features, license plates, detected car types and traveling time information are combined for the computation of cost function in ICT. Finally, robust 2D-to-3D backprojection is achieved with EDA optimization applied to camera calibration. Our superior performance in both speed estimation and vehicle re-identification is presented in the experimental results on the *NVIDIA AI City Dataset* [9].

Acknowledgment

The authors would like to thank many people who helped in the improvement of the performance of the proposed system: Chen Bai, Ge Bao, Jiarui Cai, Bill Cheung-Daihe, Tianhang Gao, Yijin Lee, Ching Lu, Shucong Ou, Ningyang Peng, Mingxin Ren, Jingwen Sun, Yi-Ting Tsai, Yun Wu, Chumei Yang, Xiao Tan, Hao Yang, and Jiacheng Zhu.

References

[1] T. Liu, Y. Liu, Z. Tang and J.-N. Hwang. Adaptive ground plane estimation for moving camera-based 3D object tracking. IEEE Int. Workshop Multimedia Signal Processing, 2017.

[2] J. Sochor, J. Špaňhel and A. Herout. BoxCars: Improving fine-grained recognition of vehicles using 3-D bounding boxes in traffic surveillance. IEEE Trans. Intelligent Transportation Syst., 2018.

[3] K.-H. Lee, J.-N. Hwang and S.-I Chen. Model-based vehicle localization based on three-dimensional constrained multiple-kernel tracking. IEEE Trans. Circuits Syst. Video Technol., 25(1):38-50, 2014.

[4] A. Milan, S. Roth and K. Schindler. Continuous energy minimization for multitarget tracking. IEEE Trans. Pattern Analysis & Machine Intelligence, 36(1):58-72, 2014.

[5] A. Milan, K. Schindler and S. Roth. Multi-target tracking by discrete-continuous energy minimization. IEEE Trans. Pattern Analysis & Machine Intelligence, 38(10):2054-2068, 2016.

[6] X. Liu, W. Liu, H. Ma and H. Fu. Large-scale vehicle re-identification in urban surveillance videos. Proc. IEEE Int. Conf. Multimedia Expo, 2016.

[7] L. Yang, P. Luo, C. C. Loy and X. Tang. A large-scale car dataset for fine-grained categorization and verification. Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 3973–3981, 2015.

[8] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos and E. Kayafas, E. License plate recognition from still images and video sequences: A survey. IEEE Trans. Intelligent Transportation Syst., 9(3):377-391, 2008.

[9] NVIDIA AI City Challenge – Data and Evaluation. [Online] Available: https://www.aicitychallenge.org/?page_id=9.

[10] C.-T. Chu, J.-N. Hwang, H.-Y. Pai and K.-M. Lan. Tracking human under occlusion based on adaptive multiple kernels with projected gradients. IEEE Trans. Multimedia, 15:1602-1615, 2013

[11] Z. Tang, J.-N. Hwang, Y.-S. Lin and J.-H. Chuang. Multiple-kernel adaptive segmentation and tracking (MAST) for robust object tracking. Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 1115-1119, 2016.

[12] Z. Tang, G. Wang, T. Liu, Y.-G. Lee, A. Jahn, X. Liu, X. He and J.-N. Hwang. Multiple-kernel based vehicle tracking using 3D deformable model and camera self-calibration. arXiv preprint arXiv:1708.06831, 2017.

[13] Z. Tang, Y.-S. Lin, K.-H. Lee, J.-N. Hwang, J.-H. Chuang and Z. Fang. Camera self-calibration from tracking of moving persons. Proc. Int. Conf. Pattern Recognition, 260-265, 2016.

[14] Z. Tang, R. Gu and J.-N. Hwang. Joint multi-view people tracking and pose estimation for 3D scene reconstruction. Proc. IEEE Int. Conf. Multimedia Expo, 2018.

[15] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. arXiv preprint arXiv: 1612.08242, 2016.

[16] B. Caprile and V. Torre. Using vanishing points for camera calibration. Int. J. Computer Vision, 4(2):127-139, 1990.

[17] Google Maps. [Online] Available: https://maps.google.com/.

[18] P. Larranaga and J. A. Lozano. Estimation of distribution algorithms: A new tool for evolutionary computation. Springer Science & Business Media, 2nd edition, 2002.

[19] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, S. Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. arXiv preprint arXiv:1511.04136, 2015.

[20] G. Wang, J.-N. Hwang, K. Williams and G. Cutter. Closed-loop tracking-by-detection for ROV-based multiple fish tracking. Int. Conf. Pattern Recognition Workshop Comput. Vis. Analysis Underwater Imagery, 7-12, 2016.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. Going deeper with convolutions. Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2015.