Personal Driving Diary: Automated Recognition of Driving Events from First-Person Videos

M. S. Ryoo[a], Sunglok Choi[1b], Ji Hoon Joung[1c], Jae-Yeong Lee[1b], Wonpil Yu[b]

[a]*Mobility and Robotic Systems Section, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109*
[b]*Robot Research Department, Electronics and Telecommunications Research Institute, Daejeon, South Korea 305-700*
[c]*Robotics Research Department of Engine & Machinery Research Institute, Hyundai Heavy Industries Co., Ltd., Yongin, South Korea 446-716*

Abstract

In this paper, we introduce the concept of *personal driving diary*. A personal driving diary is a multimedia archive of a person's daily driving experience, describing important driving events of the user with annotated videos. This paper presents an automated system that constructs such multimedia diary by analyzing videos obtained from a vehicle-mounted camera. The proposed system recognizes important interactions between the driving vehicle and the other actors in videos (e.g., accident, overtaking, ...), and labels them together with its contextual knowledge on the vehicle (e.g., mean velocity) to construct an event log. A decision tree based activity recognizer is designed, detecting driving events of vehicles and pedestrians from the first-person view videos by analyzing their trajectories and spatio-temporal relationships. The constructed diary enables efficient searching and event-based browsing of video clips, which helps the users when retrieving videos of dangerous situations. Our experiment confirms that the proposed system reliably generates driving diaries by annotating the vehicle events learned from training examples.

Keywords: personal driving diary, driving activity recognition, first-person event detection, lifelogging

1. Introduction

Personal driving diary is a multimedia archive of a person's daily driving experience. It illustrates important driving events of the user, providing recorded videos of the events and describing when and where the events have occurred. Figure 1 shows an example driving diary. The primary goal of such driving diary is to enable an efficient searching of video segments with important vehicle events (e.g., accidents), and to summarize a driving history of the user for the statistical analysis of his/her driving habits and patterns (e.g., dangerous overtaking and sudden stops). The user will be able to retrieve and examine an event log (i.e., a diary) with videos taken from his/her vehicle, and use it for various purposes.

This paper presents an automated system that generates such multimedia diary by analyzing videos obtained from a vehicle-mounted camera. The objective is to construct a system that automatically annotates and summarizes obtained driving videos, enabling fast, efficient, and used-oriented browsing (and analysis) of events. The trend of mounting video cameras on vehicles is growing rapidly (e.g., 'black box cameras' for accident recording [1]), and most of vehicles will equip cameras observing the front in the near future corresponding to the societal interest. Our motivation is to provide a personal summary of vehicle events by utilizing such cameras, and develop an efficient way of retrieving important video segments.

In this paper, we design and implement a novel system integrating various computer vision components including visual odometry, pedestrian detection, vehicle detection, tracking, and activity-level event recognition. Several existing computer vision methodologies are extended for the first-person view video analysis, and are combined with our newly designed activity recognition component, reliably generating video diaries for the users. Our system first estimates the trajectory of the driving vehicle using visual odometry, and computes trajectories of other pedestrians and vehicles by detecting/tracking them. Next, the system captures salient atomic-level movements (e.g., decelerating) from the trajectories, and further analyzes interactions between vehicles and pedestrians. Driving events including sudden stopping and overtaking are annotated from the video as a result, and they are provided as a driving log (i.e., personal driving diary) for the driver.

A notable contribution of this paper is the introduction of

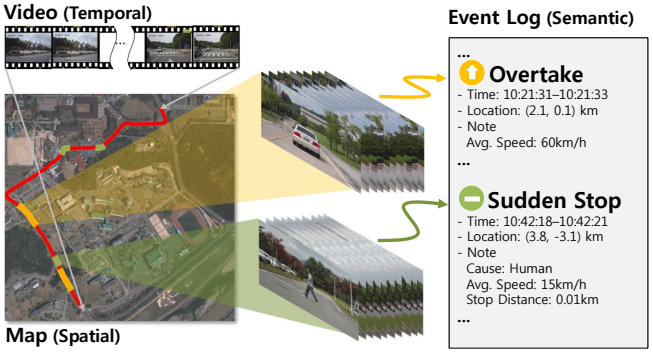


Figure 1: An example personal driving diary.

Email address: mryoo@jpl.nasa.gov (M. S. Ryoo)
[1]These three authors contributed equally to the paper.

the concept of personal driving diaries. We present a new idea that everyday driving experience of drivers can be annotated and archived, and discuss methodologies for the generation of such event-based personal driving diaries from first-person view videos. The personal driving diary constructed by our system will enable efficient searching (and retrieval) of vehicle events. Even though there has been previous attempts to apply computer vision algorithms for vehicle-mounted cameras (e.g., [2]), a system to understand vehicle activities (i.e., events) from them has not been studied in depth previously. In addition, our event recognition methodology which enables analysis of multiple object trajectories from the first-person viewpoint will benefit other types of life-logging systems where object trajectories are important (e.g., cooking).

More specifically, this paper aims to introduce and address the new problem of first-person driving video analysis/understanding. The paper attempts to provide answers to the question of what computer vision components are necessary and how they should be designed/extended/implemented for the understanding such first-person videos. Each of our components is designed to best capture dynamics of driving interactions, and the experimental results show their effectiveness. We particularly focus on the recognition of semantically meaningful driving events from the first-person view videos, which has been very limitedly explored in previous works.

This paper extends our previous work on personal driving diary [3] by providing more detailed descriptions of the proposed approaches, incorporating better algorithms, extending them, and discussing additional experiments to evaluate the performance of each vision component.

2. Related Works

Life-logging: Life-logging systems using wearable cameras have been developed to record a person's everyday experiences [4–6]. Hori and Aizawa [4] utilized multiple sensors (e.g., cameras, GPS, brain-wave analyzer, ...), automatically logging videos based on various keys from systems components such as a face detection and a GPS localization. Doherty et al. [6] also used a wearable camera. They have classified each image scene (i.e., frame) into a number of simple event categories using image features (e.g., SIFT), showing a potential that videos can be annotated based on user events.

However, most of previous life-logging systems focused on the elementary recording of entire video data [7], instead of constructing a semantic diary composed of videos of specific events. Previous systems attempted to construct general purpose achieves by relying on the index created by extracting simple image-based features, rather than performing a video-based analysis to interpret activity-level (i.e., complex) events. There also was an attempt to label location-based activities from GPS data [8], but it did not focus on semantic analysis of visual data (i.e., videos).

Human activity recognition: Human activity recognition is a computer vision methodology essential for analyzing videos [9]. Particularly, activity recognition methodologies utilizing spatio-temporal features from videos have obtained a large amount of interests [10–12]. In addition, hierarchical approaches have been studied to recognize various types of activities involving multiple humans and objects [13, 14]. However, even though previous systems successfully recognized events from videos with various settings (e.g., backgrounds), little attempts have been made to analyze activity videos from moving first-person view cameras. Few works attempted to recognize activities from wearable cameras recently [15], but they were not designed to detect high-level interactions from driving environments either.

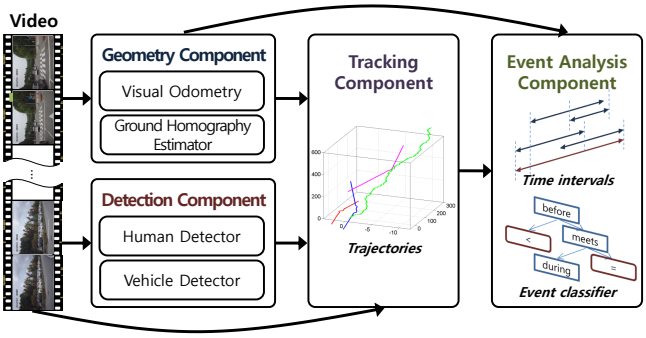


Figure 2: An overall architecture of our driving diary system.

Vehicle cameras: As described in the introduction, increasing number of vehicles are equipping cameras for safety and accident recording purposes these days [1]. In addition, various pedestrian detection algorithms have been developed and adopted for vehicle-mounted cameras [2], in order to support safer driving of drivers. However, most of the previous works limited themselves to accident prevention using simple per-frame detection, and did not attempt to analyze events from the videos. More specifically, previous works with vehicle cameras focused mainly on pedestrian detection and lane detection from image frames, rather than semantically analyzing ongoing events from video data.

3. Framework

In this section, we present an overall framework for our personal driving diary system. The idea is to provide a complete system architecture, so that an implemented system may be installed on a mobile camera system (e.g., a black box camera or a smart phone) to annotate videos taken from a driving vehicle. Various computer vision techniques are designed and adopted to extract semantic information from first-person view videos containing vehicle events.

Our driving diary system is composed of four components: geometry component, detection component, tracking component, and event analysis component. These components obtain visual inputs (i.e., videos) from the camera and interact with each other to analyze events involving the driving vehicle itself, the other appearing vehicles, and pedestrians. Figure 2 illustrates the overall architecture.

The geometry component uses a visual odometry algorithm to measure the ego-motion of the camera. That is, the trajectory of the driving vehicle is obtained with respect to its initial position, enabling our diary to record the vehicle's relative location on the map and provide ego-motion information to the other components. The detection component detects pedestrians and vehicles at every image frame of the input video. In addition, based on the geometrical structure of the scene analyzed by the geometry component, it estimates locations of the detected objects in metric coordinates (relative to the starting location). The tracking component applies object tracking algorithms to obtain trajectories of the detected pedestrians and vehicles.

Finally, our event analysis component annotates all ongoing events from continuous streams of videos using the vehicle's ego-trajectory from the geometry component and the other trajectories from the tracking component. High-level events such as 'overtaking' and 'sudden stopping caused by pedestrians' are recognized hierarchically using trajectory-based features. Our event detection component is designed to probabilistically consider spatio-temporal relationships among obtained vehicle/pedestrian trajectories. Events are annotated together with the driving vehicle's velocity and other contextual information.

As a result, our system converts an input driving video into a diary of semantically meaningful events. A user interface has been designed so that the user retrieves videos of interesting events from the diary. An interface to add new event to be annotated (by providing additional labeled training videos) is supported by our system as well. In the following sections, we discuss each of the components in detail.

4. Geometry Component

The geometry component derives geometric information from driving image sequences: (1) visual odometry estimates ego-motion of the camera-mounted vehicle, and (2) ground plane homograpy allows conversion of positions of detected objects (e.g., pedestrians or vehicles) on a road from image coordinates to metric coordinates. The goal is to obtain the trajectory of the driving vehicle and to enable computation of those of the other appearing pedestrians/vehicles in metric coordinates.

4.1. Visual Odometry

Monocular visual odometry is adopted to generate a trajectory of a camera mounted on our vehicle. That is, we compute an approximate trajectory of the driving vehicle based on ego-motion displayed in its videos. The idea is that, even though visual odometry suffers from drift errors similar to the other dead reckoning techniques by its nature (in contrast to GPS-based localization), it is able to provide accurate/reliable piecewise local trajectory information for our driving event analysis component. Our visual odometry approach is composed of five steps: feature extraction/matching, outlier rejection, relative pose estimation, scale estimation, and motion accumulation. Figure 3 describes each step with its implemented techniques.

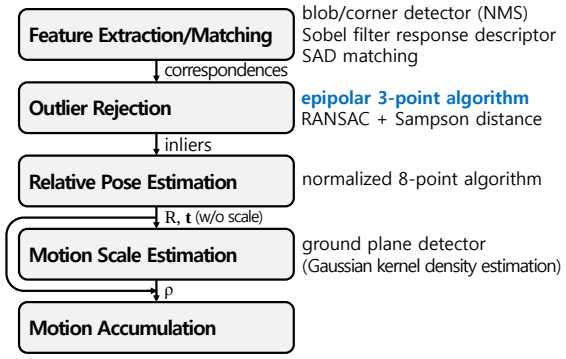


Figure 3: Five steps of our visual odometry is presented with their specification. We utilize a novel 3-point algorithm based on epipolar geometry with constrained motion in outlier rejection. We adopt feature extraction/matching and scale estimation implemented in the well-known open-source software, LIBVISO2 [16].

First, we use a blob/corner detector and sum-of-absolute-differences (SAD) matching to generate feature correspondences between an adjacent pair of images [16]. Here, certain correspondences do not associated with camera motion because they are generated from moving objects or wrong feature matching. In order to remove such outliers and estimate camera motion accurately, we designed an epipolar version of a 3-point algorithm in conjunction with RANSAC. Our novel 3-point algorithm is based on epipolar geometry with constrained motion, which is different from the conventional 3-point algorithm for stereo configurations with a known depth. Finally, once outliers are removed, we apply the normalized 8-point algorithm [17] to estimate camera motion from inliers in the least-squares sense and use the constant distance between the ground and camera to resolve scale ambiguity of the estimated motion.

Ourlier rejection for visual odometry: Our outlier rejection is more specialized in driving situations in the view of motion models. Generally, motion is represented in 6-degree-of-freedom (DoF) and its epipolar geometry is written by an essential matrix. However, a vehicle on a road exhibits locally planar motion, making the 6-DoF general motion model redundant for short-term motion estimation. The planar motion is represented by three variables, (x, y, θ) in the rectangular coordinate or (ρ, ϕ, θ) in the polar coordinate. We additionally consider pitch rotation, ϵ, for small and abrupt motion due to vibration and bumps on a road. Overall, the 4-DoF motion is represented by

$$\mathrm{R} = \mathrm{R}_y(\theta)\mathrm{R}_x(\epsilon) \quad \text{and} \quad \mathbf{t} = \rho \begin{bmatrix} \sin\phi \\ 0 \\ \cos\phi \end{bmatrix}, \tag{1}$$

and its epipolar constraint is written by

$$\mathbf{x}'^{\top}\mathrm{E}\mathbf{x} = 0 \quad (\mathrm{E} = [\mathbf{t}]_{\times}\mathrm{R}), \tag{2}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are a pair of corresponded features between two adjacent images. Our novel 3-point algorithm finds motion, (ϕ, θ, ϵ), using Newton's method with three pairs of correspondences. The scale of motion, ρ, is separately estimated based

Figure 4: Feature correspondences are classified into inliers (red) and outliers (blue) in outlier rejection. This image is from KITTI dataset (No. 7, 729th frame).

on our prior knowledge on the distance between the ground and the camera.

We use our 3-point algorithm to identify outliers with a framework of RANSAC. Our proposed algorithm requires less number of correspondences than previous 5-point or 8-point algorithm, so RANSAC needs exponentially less number of iterations than the others. This makes our outlier rejection much faster than the others. In addition, the proposed method is more robust against noise because it only considers major components of motion and ignores minor parts. Figure 4 presents an example outlier rejection by the proposed algorithm.

4.2. Ground Plane Homography

Planar homography is utilized to estimate the relative position of observed objects from a single camera. The ground plane is a good clue to derive the position because it is the most significant plane during driving and all on-road objects are in contact with the ground. Ground plane homography H_G is a point-to-point transformation between the ground region on an image and the ground plane in the metric world. Therefore, the metric position of an object $\mathbf{X}$ is acquired by

$$\mathbf{X} = \mathrm{H}_\mathrm{G}\mathbf{x}\,, \tag{3}$$

where $\mathbf{x}$ is a pixel-unit point where the object meets the ground in the image. In our application, we assume that the ground plane homography is constant because the mounted camera is fixed on the vehicle and a road is locally flat. We acquired the ground plane homography using a 4-point algorithm with more than 10 pairs of correspondences.

5. Detection Component

The goal of the detection component is to locate pedestrians and vehicles appearing in input video. Our detection component adopts and extends state-of-the-art object recognition methodologies, so that they are able to perform more reliable detection of pedestrians and vehicles in the driving environment based on image features. The location and size of each object in the scene (i.e., a pedestrian or a vehicle) are estimated per image frame, and they are transformed into metric coordinates using the geometry component.

The detection component estimates the locations of pedestrians in each frame as 2-D rectangles in the image. Our detection component is based on the image-based pedestrian detector introduced in [18], which improves the part-based detector [19]

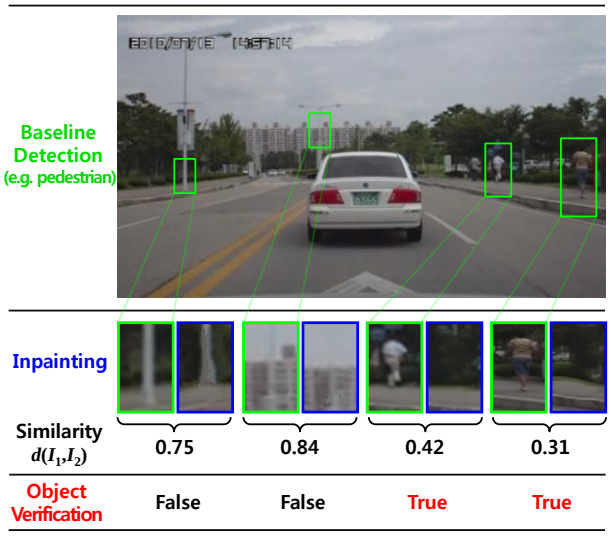


Figure 5: An overall process of our object detection using inpainting. A bounding box obtained from baseline detector is designated as the inpainting region, and the similarity is measured by comparing the inpainted region and the original region. Finally, our object detector selects only the object candidates which have low similarity between original region (left) and inpainted region (right).

with an inpainting technique. In our previous work, we proposed a new application of conventional inpainting algorithms whose original goal was a restoration of damaged paintings, so that the inpainting distinguishes the failures of the baseline detector (e.g., a part-based detector). The idea was that an original image region (e.g., a bounding box) and its inpainting result are similar only when the region corresponds to a background.

For our pedestrian detection in a driving environment, we further improve the inpainting-based pedestrian detection. The idea is that the concept of inpainting to recover background regions (e.g., Figure 5) are especially sutiable for a driving environment because a picture taken from a vehicle-mounted camera in a driving environment consists of the continuous structures which are suitable for consistent inpainting result (e.g., woods, road, sky, and so on). For our detection component, we implement a new image-to-image similarity calculation algorithm to better consider the characteristic of the exemplar-based inpainting. The original algorithm calculates the difference by comparing a pixel at the same position in an inpainting result and an original image. Its limitation is that the exemplar-based inpainting simply pastes a square-shaped exemplar patch to the inpainting region (to recover the region), making the brute-force pixel-wise comparison between two images (i.e., the image regions before inpainting and after inpainting) inaccurate.

Therefore, we compare the adjacent pixel to find the best matched pixel using Equation (4).

$$d(I_1, I_2) = \frac{1}{N(\Omega)} \sum_{\mathbf{p}\in\Omega} \min\Big(f\big(I_1(\mathbf{p}), I_2(\mathbf{p}-\mathbf{k})\big)\Big)\cdot s(\mathbf{p}) \tag{4}$$

where $d(I_1, I_2)$ is the difference between two images, and Ω is the inpainted region. $f(I_1(\mathbf{p}), I_2(\mathbf{p}-\mathbf{k}))$ is 1 if the difference between a pixel of inpainted region $I_1(\mathbf{p})$ and a pixel around $\mathbf{p}$ of original image $I_2(\mathbf{p}-\mathbf{k})$ is small enough otherwise 0. $N(\Omega)$

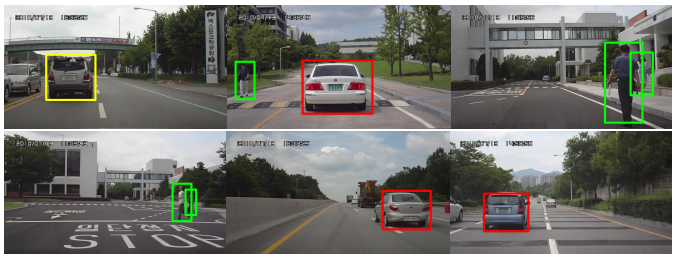


Figure 6: Example pedestrian/vehicle detection results obtained from our detection component.

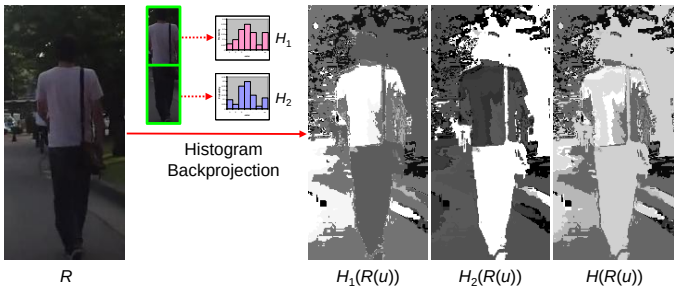


Figure 7: A sample image and corresponding backprojection images using 2×1 partition model: (a) input image, (b) backprojection image using the upper part histogram of the object, (c) backprojection image using the lower part histogram of the object, (d) conventional backprojection image using single histogram model.

is the area of the inpainted region, and we divide $N(\Omega)$ to normalize the similarity. $s(\mathbf{p})$ is the weight based on object shape.

Our system adopts the Viola and Jones method [20] to detect rear-view of appearing vehicles. The Violar and Jones method extracts the Haar-like features and trains object detectors using AdaBoost. Our implementation was trained to recognize three categories of vehicles (sedans, SUVs, and buses), so that it covers diverse vehicles appearing in a driving environment. We take a picture and crop the rear view of vehicles in the picture, and group them by the types of vehicles to train them independently.

Once detection results (i.e., image locations of appearing pedestrians and vehicles) are obtained and their locations in the metric coordinate are estimated using our geometry component, they are passed to the tracking component. Figure 6 shows example detections from our videos. The tracking component which we discuss in the following section computes 3-D XYT trajectories of vehicles and pedestrians based on the detection results.

6. Tracking Component

The objective of the tracking component is to estimate 3-D XYT trajectories of all detected pedestrians and vehicles in metric coordinates. The tracking component maintains a single hypothesis for each object. Detection results from the detection component (at every frame) are matched with the maintained object hypotheses using a greedy algorithm described in [21]. For each of unmatched detections, a new hypothesis is created and an object model is built from the corresponding image patch. Whenever the match fails, a visual tracker is applied to update the unmatched object hypotheses. The actual trajectories are generated by applying extended Kalman filters (EKFs) with a constant velocity model in metric coordinates.

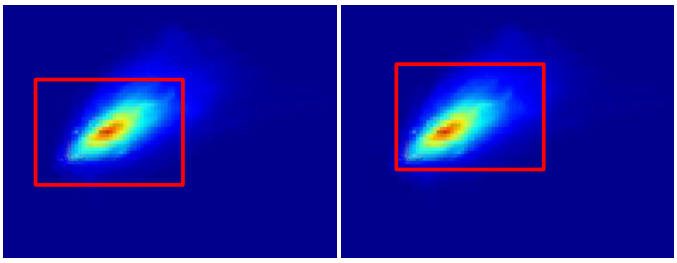


Figure 8: Mean-shift (left) versus proposed likelihood measure (right).

A visual tracker, which searches for the best matching object region in the next frame based on its appearance in the current frame, plays an important role of tracking unmatched objects and recovering possible detection errors. In our previous work, we developed the idea of a partition-based mean-shift tracker [22] where an object model is represented by multiple patch histograms and its mean-shift iteration is applied on histogram backprojection densities to locate the target (Figure 7).

For the tracking of pedestrians in our driving environment, we take advantage of the improved version of our partition-based mean-shit tracker that obtains significantly better tracking performance by substituting the mean-shift iteration by a new likelihood measure [23]. The proposed likelihood measure is computed by averaging the local density estimates from histogram backprojection and given by

$$L(u) = \frac{1}{M} \sum_{i=1}^{M} K(u_i - u) D_{g(u_i)}(u_i), \tag{5}$$

where $K(\cdot)$ is a radially symmetric kernel defining an influence zone, $\{u_i\}_{i=1,\ldots,M}$ is the pixel locations within a local window centered at u, the function g associates the partition index to the pixel locations, and $D_{g(u_i)}(u_i)$ is the density estimates computed from $g(u_i)$-th patch histogram.

The difference of the proposed likelihood measure from the mean-shift is illustrated in Figure 8. The mean-shift framework implicitly assumes that D_j represents the probability distribution of the target location, finding the local peak of the density as the center location of the object. On the other hand, in our approach we interpret D_j as the probability of each pixel being an object. As a result, our tracker chooses the window that contains the largest number of object pixels as the object location by maximizing Equation (5).

For the tracking of rigid objects (vehicles), we adopt a simple HSV template tracker [22]. For each initial detection of a vehicle, a HSV template is built from the corresponding image region, forming an object model. A target object is then located by minimizing the average color difference between the object model and candidate region. Different from histogram-based approaches, template matching utilizes raw colors of an object

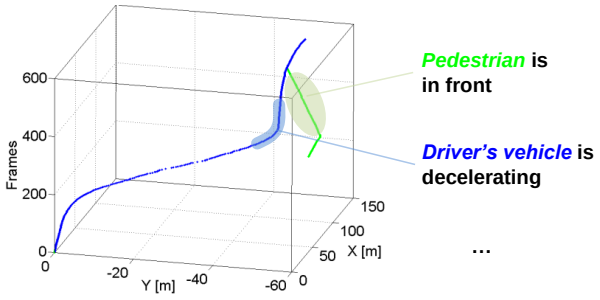


Figure 9: An illustration of atomic sub-events detected from the trajectory of the driving vehicle (green) and that of the pedestrian (blue). The event of 'decelerating' is detected solely based on the XYT trajectory of the vehicle, and the event of 'person in front' is detected based on features from the two trajectories.

with their spatial information in pixel level, and therefore it is less sensitive to background clutters and is able to locate an object more accurately if the object has a rigid body and if the orientation is fixed.

7. Event Analysis Component

The role of the event analysis component is to label all ongoing events of the vehicle given a continuous video sequence. Our driving diary essentially is a set of labeled driving events, and the event analysis component serve as a core to generate it by detecting driving events from videos. We extended the previous approach of spatio-temporal relationship match (STR-match) [12] that obtained successful results on human activity recognition, so that events are learned and recognized in an additive fashion. Learned driving events are represented in terms of simpler sub-events, and they are recognized by hierarchically analyzing the relationships among the detected sub-events. We first discuss recognition of atomic-level sub-events in Subsection 7.1, and present our hierarchical driving event recognition and learning approach in Subsection 7.2. Finally, we present our probabilistic driving event detection approach in Subsection 7.3.

7.1. Atomic Sub-event Detection

Our system first recognizes nine types of elementary sub-events that consist complex vehicle events, which serve as building blocks of our hierarchical event detection process: 'car passing another', 'car passed by another', 'car is at front of another', 'car at behind of another', 'cars side-by-side', 'accelerating', 'decelerating', 'vehicle stopped', and 'pedestrian in front'. These semantic sub-events are used as the system's vocabulary to describe complex driving events. The system recognizes the sub-events using four types of features directly extracted from local 3-D XYT trajectories of the driving vehicle and the other objects (i.e., pedestrians and vehicles): 'orientation', 'velocity', 'acceleration', and 'relative XY coordinate of the interacting vehicle'. Based on the features computed from the trajectories, sub-events are inferred using Bayesian classifiers. More specifically, one Bayesian classifier using four types of features are constructed per sub-event, and it is applied for all possible video segments to decide whether the segment matches the sub-event (i.e., sliding windows). As a result, time intervals (i.e., pairs of starting time and ending time) of all occurring sub-events are recognized, and are provided to the system for the further analysis. Figure 9 shows example sub-events detected from XYT trajectories.

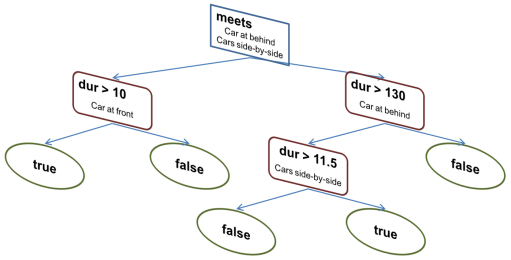


Figure 10: Example spatio-temporal relationship (STR) decision tree of a driving event 'overtake'. The left child of a node is activated when the relation corresponding to the node is true, and the right child is activated other wise.

7.2. Hierarchical Decision Tree-based Event Detection

We implement a decision-tree version of the STR-match to recognize vehicle activities from detected sub-events. Our event analysis component learns decision-tree classifiers from training examples, automatically mining important spatio-temporal patterns (i.e., relationships) among sub-events. That is, we statistically train an event detector per activity, which will make the videos containing the corresponding event to reach a leaf node with the 'true' label when tested with the decision tree.

Our STR decision tree is a binary decision tree where each node of it corresponds to a predicate describing a condition of a particular sub-event (e.g., its duration *greater* than a certain threshold) or a relationship between two sub-events (e.g., a time interval of one sub-event must occur *during* the other's). As single-sub-event predicates, we use the simple two predicates with a threshold parameter (*greater* and *less*). Allen's temporal predicates [24] (*equals*, *before*, *meets*, *overlaps*, *during*, *starts*, and *finishes*) and their inverse predicates are adopted to describe relations between two sub-events. These predicates not only describe that certain sub-events must occur in order for the activity to occur, but also describe necessary temporal relations among the sub-events' time intervals.

The recognition is performed by traversing the tree from the root to one of its leaves, sequentially testings whether its sub-event detection results satisfy the predicate of each node. If it does, the recognition system traverses to the left child of the node. Otherwise, it must go to the right child. Figure 10 shows an example STR decision tree learned from training video sequences. The decision tree illustrates that in order for a driving event of 'overtaking' to occur, its sub-events 'car at behind of another car', 'cars side-by-side', and 'car at front of another car' must occur while satisfying a particular structure.

The decision trees are learned by iteratively searching for the predicate which maximizes the *gain* given the sub-event detection results of training sequences. The new node (i.e., the pred-

icate) providing the maximum information gain is added to the tree one by one based on training examples. The gain of the decision tree caused by adding a new predicate to one of its leaves is defined as follows:

$$Gain(S,n) = Entropy(S,n) - \sum_{b} \frac{|S_b|}{|S|} Entropy(S_b,n) \tag{6}$$

where b is a binary variable, S is a set of training examples, and S_b is the subset of S having value b for node n. Here, the entropy is defined as:

$$Entropy(S,n) = -p_0^n \log_2(p_0^n) - p_1^n \log_2(p_1^n) \tag{7}$$

where p_0^n is the fraction of negative examples in S for node n, and p_1^n is the fraction of positive examples in S. If S is divided into two sets of an identical size, the entropy is 1 and we have the gain of 0.

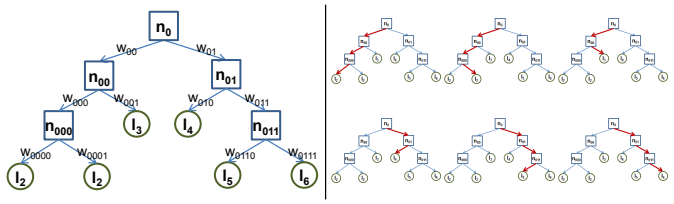
Essentially, our learning algorithm is searching for the predicate that divides the training examples into two sets whose size difference is the greatest (i.e., most unbalanced). Each of the left child and the right child of the added node either becomes a leaf node that decides that the driving event has occurred, or becomes an intermediate node waiting for another predicate to be added. A greedy search strategy is applied to find the STR decision tree that provides maximum gain given training videos.

In order to make our STR decision tree learning incremental (i.e., in order to enable addition of user-specific events), we take advantage of the incremental tree induction (ITI) method [25]. The ITI method is incorporated into our STR tree learning process, which recursively updates the trees after each addition of a new video example to ensure the optimum gain. That is, we are representing driving events in terms of decision tree composed of temporal predicates, and are taking advantage of the incremental tree update method for online learning. Our trees allow a user to feed videos of the new event to be annotated.

As a result, our system recognizes complex vehicle events (e.g., overtaking) incrementally learned from training videos. The personal driving diary is constructed by concatenating annotated driving events while describing other context including locations of the vehicle, vehicle tracking histories, and/or pedestrian tracking histories.

7.3. Probabilistic Event Detection

This subsection presents a probabilistic approach to recognize driving events from videos. The deterministic recognition approach using a decision tree presented in the previous subsection is able to make a hard decision whether the video contains an event or not, but it is unable to adjust the sensitivity of such detections (i.e., there is no decision threshold) or provide any confidence values for the detections. This makes its integration with real-world applications very challenging particularly when there are multiple types of events requiring different levels of sensitivities (e.g., dangerous events vs. normal events). Therefore, in this subsection, we extend our spatio-temporal relationship tree approach, so that it explicitly computes probabilities (i.e., confidences) of driving events while recognizing them.



(a) An example decision tree (b) Ancestor edges for each leaf

Figure 11: (a) An example decision tree with variables to denote nodes and edges, and (b) a figure illustrating ancestor edges for each leaf node which needs to be enumerated during the probability computation.

We view the learned decision tree as layers of semantic rules that will lead to a particular decision given an unknown video observation. The idea is to compute the probability of the observation reaching each leaf node of the tree by satisfying all semantic rules of its ancestor edges, and integrate them to make the probabilistic decision.

Let l indicate a particular leaf node of the decision tree, and let W_l denote a set of its ancestor edges. For example, in Figure 11, the set W_{l_5} corresponding to the leaf l_5 will be: $W_{l_5} = \{w_{01}, w_{011}, w_{0110}\}$. We compute the probability of an event a occurring in the given video v (i.e., $P(a|v)$) by multiplying the probability of W_l being satisfied given the video (i.e., $P(W_l|v)$) and the posterior probability of the leaf node corresponding to the event a (i.e., $P(a|l, W_l)$):

$$P(a|v) = \sum_{l} P(a|l, W_l) P(W_l|v). \tag{8}$$

Thus, in our new tree representation, each leaf node maintains a value describing the posterior probability $P(a|l, W_l)$ to support the probability computations,

$$P(a|l, W_l) = \frac{p_1^l}{p_0^l + p_1^l} \tag{9}$$

where p_0^l is the fraction of positive training samples satisfying W_l, and p_1^l is the fraction of positive training examples satisfying W_l. These fractions are computed during the decision tree learning period discussed in Subsection 7.2.

$P(W_l|v)$ is the probability of binary conditions W_l being satisfied given the video input v. As discussed in Subsection 7.2, W_l is composed of predicates with one/two parameters, which must be tested based on the atomic sub-events detected from trajectory features. Here, in order to compensate for noisy observations and recover errors in estimating atomic sub-events from features probabilistically, we introduce a new variable $T = \{t_1, ..., t_m\}$ which is a set of time intervals of atomic sub-events:

$$P(W_l|v) = \sum_{T} P(W_l|T) P(T|v) \approx \max_{T} P(W_l|T) P(T|v). \tag{10}$$

The idea is to consider multiple possible time intervals t_i per atomic sub-event i (instead of using one fixed time interval as in the deterministic method), which serves as an input to our probabilistic decision tree. We define $P(W_l|T)$ as 1 only when

all conditions W_l are satisfied for the input time intervals T, and 0 otherwise. $P(T|v) = \prod_i P(t_i|v)$ is defined using the atomic sub-event detection results as follows:

$$P(t_i|v) = \exp\big(-\min_{q_i} ||t_i - q_i||\big) \tag{11}$$

where q_i is a time interval of the ith atomic sub-event detected in Subsection 7.1, and $||t_i - q_i||$ is the distance between two time intervals.

The computation of $P(a|v)$ requires consideration of $O(c^m)$ number of time interval combinations in principle, where m is the number of atomic sub-events used in the decision tree and c is the number of time intervals candidates being considered.

In order to avoid an exponential number of computations and enable efficient approximation of the solution, we further propose the following greedy recursive formulation: $P(a|v) = F(T, n_r)$ where n_r is the root node, and

$$\begin{aligned} F(T,n) = \max_{T_n} \; & F(T,n0)P(w_{n0}|T_n)P(T_n|v) \\ & + F(T,n1)P(w_{n1}|T_n)(1 - P(T_n|v)) \end{aligned} \tag{12}$$

for all non-leaf nodes n and

$$F(T,l) = P(a|l) \tag{13}$$

for all leaf nodes l. T_n is the set of time intervals related to the node n (either a single interval or a pair of intervals), and $n0$ is the left child of the node n and $n1$ is the right child of n. The time complexity to compute $F(T, n_r)$ is $O(c^2 \cdot |n|)$ in the worse case, where $|n|$ is the number of nodes.

Our formulation is different from traditional probabilistic decision trees [26] in the aspect that our approach interprets noisy video observations using the time interval set variable T, which is designed particularly for the driving event detection.

8. Experiments

In this section, we evaluate the accuracy of the personal driving diary generated by our system. Our driving diary is an event-based log of the user's driving history, implying that the correctness of the diary must statistically be evaluated by measuring the event annotation accuracy. For our experiment, we constructed a new dataset with driving video scenes taken from a first-person view camera attached to a vehicle. Using this dataset involving various types of driving events, we tested our system's ability to annotate time intervals of ongoing events.

In Subsection 8.1, we first introduce the new dataset composed of driving videos. Next, in Subsection 8.2, we measure the performance of our system's each component using the established dataset as well as other existing public datasets. Finally, in Subsection 8.3, we evaluate the accuracy of the constructed driving diary, which is the event annotation performance.

8.1. Dataset

Our dataset focuses on six types of common driving events which are semantically important: long stopping, overtake, overtaken, sudden acceleration, sudden stop - pedestrian, and sudden stop - vehicle. A 'long stopping' describes the situation which the driving vehicle was staying stationary for more than 15 seconds. A 'sudden stop - pedestrian' indicates that the car was suddenly stopped because of the pedestrian ahead, and a 'sudden stop - vehicle' corresponds to an event of the car being stopped by another car in its front.

We have collected more than 100 minutes of driving videos from a vehicle-mounted camera. The camera was mounted under the rear-view mirror, observing the front. The dataset is segmented into 52 scenes, where each of them contains 0 to 3 events. As a result, a total of 60 event occurrences (i.e., 10 per event) has been captured by our dataset, and their time interval ground truths are provided. In addition, image locations (i.e., bounding boxes) of pedestrians and vehicles appearing in the videos were labeled by human annotators.

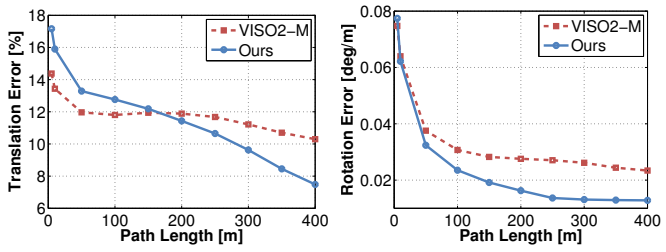
8.2. Component-wise Evaluation

This subsection presents performances of the system's low-level components used to obtain 3-D XYT trajectories: geometry component, detection component, and tracking component. Since our new dataset is a pure video dataset without any absolute location information of the vehicle, we use a public dataset to measure the performance of the geometry component. For the evaluation of the detection component and tracking component, our new driving diary dataset was directly used.

8.2.1. Geometry Component - Visual Odometry

Our visual odometry was verified on KITTI visual odometry dataset [27] because this dataset accompanies with the true trajectories acquired by highly precise GPS/IMU system, OXTS RT3003. The dataset was captured by a stereo camera, a pair of PointGrey Flea2 cameras whose resolution was 1241x376 with 10Hz framerate, respectively. We only used left-side images among the given stereo sequence because our system aims at a single camera. We used four image sequences (No. 4, 5, 6, and 7) which have consistent calibration parameters. Four sets are composed of 271, 2761, 1101, and 1101 numbers of images, respectively.

We compared performance of our visual odometry with LIBVISO-2 [16]. We used its monocular version, known as VISO2-M. Its outlier rejection is based on RANSAC with 6-DoF general motion model (fundamental matrix) and Sampson error. In contrast to our approach, each motion hypothesis is generated by eight correspondences with the normalized 8-point algorithm. To focus on verifying our proposed outlier rejection, we adopted same feature tracking and scale estimation with VISO2-M. VISO-M and our visual odometry had same values of parameters except two RANSAC parameters, the number of iterations and outlier threshold. VISO2-M was configured at 2000 iterations and 10^{-5} threshold (default), but our visual odometry was at 60 iterations and 10^{-7} threshold. Their performance was quantified by translational error, rotational error [27], and computing time. Two error measures were calculated on all available subsequences of 10 different path lengths. Our implementation used a single core (C/C++) and computing time was measured at Intel Core i7 CPU at 2.80GHz.



(a) Translational Error (b) Rotational Error

Visual Odometry	VISO2-M	Ours
Computing Time	145.4±12.1 [msec]	86.5±11.8 [msec]

(c) Computing Time

Figure 12: Accuracies were measured in terms of (a) translational errors and (b) rotational errors. They are described with respect to 10 different path lengths, $[5, 10, 50, 100, 150, 200, \cdots, 400]$. Computing time (c) is whole processing time per frame including feature extraction/matching, outlier rejection, pose estimation, and scale estimation.

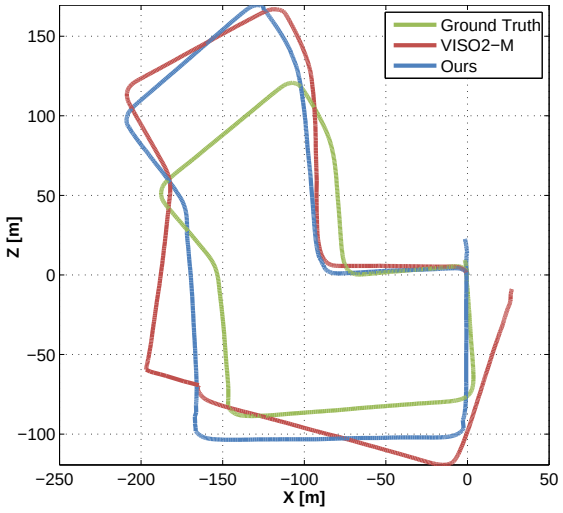


Figure 13: Trajectories with their ground truths are described for KITTI dataset (No. 7).

The accuracy and computing time of each method is presented in Figure 12. Example trajectories are also shown in Figure 13. The proposed visual odometry displays almost 30 percents less rotational error for path lengths great than 150 meters. It is because of the fact that our 4-DoF motion model is more robust to noise than 6-DoF motion model when the vehicle undergoes (approximate) planar motion. Even though the proposed method shows more translational error in short path length, it obtains better results after 150 meters path length. The proposed outlier rejection does not take into account y-directional translation and rolling motion, making inliers for pose estimation have less y-directional translation and rolling than the ground truth. This causes less accurate scale estimation for short ranges. However, the proposed method showed better results (i.e., less translational drift) in long range, since it provides more accurate rotational angles. The proposed method also spent around 40 percents less computing time because our outlier rejection requires less number of iteration in RANSAC.

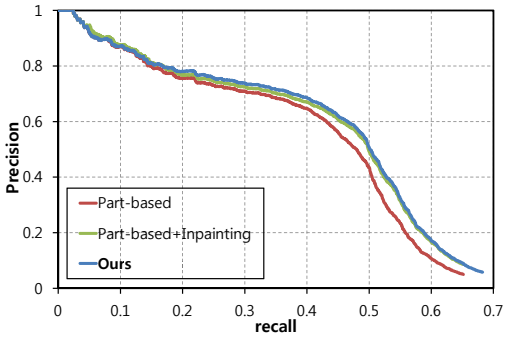


Figure 14: Improvement of precision-recall curve with part-based detector using our dataset. We are able to observe that our approach which considers the characteristic of exemplar-based inpainting obtains better precision-recall graphs.

8.2.2. *Detection Component*

We report the performance of pedestrian and vehicle detectors measured with our dataset. Our driving video dataset includes a total of 22,594 annotated rear views of vehicle and 3,598 pedestrians.

The precision-recall graph of the pedestrian detector described in Section 5 is presented in Figure 14. The part-based detector was used as a baseline pedestrian detector, and it detects the 2514 of true positives and 93,600 of the false positives. As a result, the average precision (AP) of the baseline detector was 0.399. The AP of [18] with the part-based detector and the inpainting was 0.419. The average precision of our method with an improved similarity measure for the driving environment was 0.424, and it lost 79 true positives while filtering out 54,219 false positives. In driving videos, our approach was able to eliminate 57.9% of false positives while losing only 3.15% of true positives.

The precision for the vehicle detector was 0.539 and the recall was 0.109. The detectors obtained results which were sufficient for the tracking component to reliably generate trajectories.

8.2.3. *Tracking Component*

In this section, we evaluate the performance of our visual tracker on real video sequences and compare the performance with state of the art tracking algorithms. For this experiment, we selected 10 pedestrian video sequences and 10 vehicle sequences from our driving video dataset. These test videos involve a large amount of scale changes of target objects since they are obtained from a driving environment. The tracking performance is evaluated by using two quality measures: (1) a tracking accuracy and (2) a successful tracking rate. The tracking accuracy is measured by the object detection-criterion of the VOC challenge [28], which is given by

$$\lambda_{accuracy} = \frac{|\Omega \cap \Psi|}{|\Omega \cup \Psi|}, \tag{14}$$

where Ω denotes a detected bounding box and Ψ the ground truth bounding box. The second measure of successful tracking

Table 1: Performance of the pedestrian trackers – Tracking accuracies

Seq.	Frames	MS	P-MS	VTD	PN	CT	TM	**P-ML**
ped01	156	0.70	0.76	0.11	0.10	0.74	**0.79**	**0.79**
ped02	152	**0.56**	0.52	0.51	0.50	0.42	0.50	0.55
ped03	83	0.44	0.51	0.39	0.37	**0.54**	0.23	0.51
ped04	78	0.33	0.47	0.42	0.52	0.33	0.43	**0.55**
ped05	500	0.71	0.72	0.41	0.24	0.32	0.56	**0.74**
ped06	78	0.53	0.53	0.55	0.49	0.47	0.45	**0.60**
ped07	94	0.11	0.11	0.11	**0.29**	0.24	0.10	0.11
ped08	171	0.75	0.75	0.63	0.46	0.36	0.39	**0.77**
ped09	153	0.50	0.59	0.61	0.43	0.47	0.35	**0.75**
ped10	166	0.46	**0.54**	0.33	0.28	0.31	0.45	0.48
Avg.		0.51	0.55	0.41	0.37	0.42	0.43	**0.58**

Table 2: Performance of the pedestrian trackers – Successful tracking rate

Seq.	Frames	MS	P-MS	VTD	PN	CT	TM	**P-ML**
ped01	156	**1.00**	**1.00**	0.13	0.09	0.97	**1.00**	**1.00**
ped02	152	**0.77**	0.52	0.48	0.68	0.19	0.29	0.74
ped03	83	0.41	0.53	0.41	0.53	**0.59**	0.12	**0.59**
ped04	78	0.19	0.38	0.38	0.44	0.38	**0.63**	**0.63**
ped05	500	0.99	**1.00**	0.50	0.20	0.30	0.75	**1.00**
ped06	78	0.44	**0.69**	0.63	0.63	0.31	0.31	0.63
ped07	94	0.11	0.11	0.11	**0.32**	0.26	0.11	0.11
ped08	171	**1.00**	**1.00**	0.83	0.34	0.46	0.29	**1.00**
ped09	153	0.36	0.81	0.55	0.29	0.42	0.29	**1.00**
ped10	166	0.35	0.62	0.44	0.38	0.41	**0.68**	0.65
Avg.		0.56	0.66	0.44	0.39	0.43	0.45	**0.73**

Table 3: Performance of the vehicle trackers – Tracking accuracies

Seq.	Frames	MS	P-MS	VTD	PN	CT	P-ML	**TM**
veh01	230	0.82	0.83	**0.94**	0.71	0.91	0.87	**0.94**
veh02	227	0.49	0.55	**0.78**	0.67	**0.78**	0.55	0.77
veh03	237	0.48	0.48	0.70	0.73	**0.80**	0.52	0.73
veh04	212	0.63	0.76	0.77	0.67	0.82	0.77	**0.84**
veh05	235	0.46	0.53	0.79	0.62	**0.82**	0.52	0.80
veh06	238	0.75	0.77	**0.86**	0.70	0.83	0.81	0.83
veh07	370	0.49	0.57	0.74	0.62	0.72	0.67	**0.78**
veh08	208	0.66	0.67	0.70	0.67	0.73	0.76	**0.78**
veh09	234	0.70	0.75	0.87	0.71	**0.89**	0.79	**0.89**
veh10	510	0.33	0.36	**0.85**	0.50	0.84	0.34	**0.85**
Avg.		0.58	0.63	0.80	0.66	0.81	0.66	**0.82**

Table 4: Performance of the vehicle trackers – Successful tracking rate

Seq.	Frames	MS	P-MS	VTD	PN	CT	P-ML	**TM**
veh01	230	0.98	**1.00**	**1.00**	0.89	**1.00**	**1.00**	**1.00**
veh02	227	0.30	0.50	**1.00**	0.83	**1.00**	0.72	**1.00**
veh03	237	0.44	0.65	**1.00**	0.92	**1.00**	0.73	0.98
veh04	212	0.77	**1.00**	**1.00**	0.93	**1.00**	**1.00**	**1.00**
veh05	235	0.28	0.55	**1.00**	0.81	**1.00**	0.43	**1.00**
veh06	238	0.96	0.94	**1.00**	0.94	**1.00**	**1.00**	**1.00**
veh07	370	0.45	0.82	**1.00**	0.92	0.85	0.99	**1.00**
veh08	208	0.90	0.95	0.95	0.95	0.98	**1.00**	**1.00**
veh09	234	0.87	0.94	**1.00**	0.96	**1.00**	0.96	**1.00**
veh10	510	0.34	0.39	**1.00**	0.56	**1.00**	0.38	**1.00**
Avg.		0.63	0.77	1.00	0.87	0.98	0.82	**1.00**

rate is computed by the ratio of the successfully tracked frames. A tracked frame is considered as successful if the tracking accuracy is larger than 0.5.

As described in Section 6, we use our partition-based maximum likelihood tracker (P-ML) for the pedestrian tracking and the template tracker (TM) for the vehicle tracking. In addition to these two trackers, we also implemented two baseline trackers and tested three recent state-of-the-art trackers, comparing their results with our tracking results. Comanicui's mean-shift tracker (MS) [29] and the partition-based mean-shift tracker (P-MS) [22] are the two baseline trackers we implemented. VTD tracker [30], PN tracker [31], and Context Tracker (CT) [32] are the three state-of-the-art trackers we also tested. For each of these three trackers, we used the program code distributed by the original authors of the tracking algorithm.

Table 1 shows the average tracking accuracies of the tested methods on pedestrian tracking, and Table 2 shows the results in terms of successful tracking rates. We are able to observe that the proposed histogram-based tracker with maximum likelihood localization (P-ML) shows the best performance for the pedestrian tracking. Particularly, the results in Table 2 show that our visual tracker is able to track target objects successfully for most cases. One exception was the 'ped07' sequence, where our tracker failed at very early stage of tracking due to heavy occlusion.

Similarly, Table 3 shows the average tracking accuracies of the tested methods on vehicle tracking, and Table 4 shows their results in terms of successful tracking rates. In the case of the vehicle tracking, our template tracker and VTD tracker obtained the best performances. The approaches had less difficulty tracking rigid vehicles compared to the tracking of non-rigid pedestrians in general. Figure 15 shows sample snapshots of our visual tracker on the test dataset.

8.3. Personal Driving Diary Evaluation

Finally, in this subsection, we evaluate the accuracy of the constructed personal driving diary using our video dataset. We measured the event annotation accuracies (i.e., detection accuracies) of our system. That is, given a continuous video stream, for each event, the system was asked to provide all time intervals that it believes to contain the event. In our experiments, an event annotation was treated as a correct annotation if and only if the labeled interval overlaps more than 50% with the ground truth label (provided by the human). Otherwise, it was treated as a false positive.

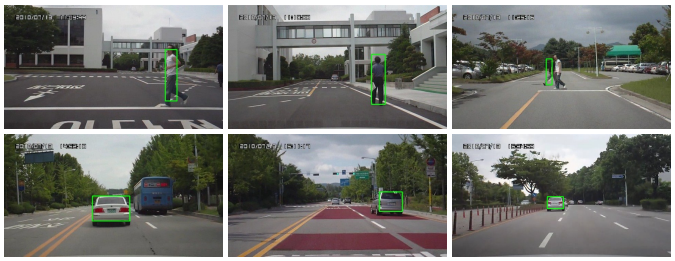


Figure 15: (*first row*) Sample snapshots of our histogram-based tracker (P-ML) on pedestrian sequences. The sequence name is ped04, ped05, ped07 in order. (*second row*) Sample snapshots of our template tracker (TM) on vehicle sequences. The sequence name is veh01, veh04, veh10.

We implemented (i) the deterministic version of our approach described in Subsection 7.2 and (ii) our probabilistic event recognition approach of Subsection 7.3. In addition, in order to illustrate the advantage of the recognition methodology proposed in this paper over existing state-of-the-arts, we implemented (iii) a SVM classification-based approach using local spatio-temporal features [11]. This baseline approach follows the concept of bag-of-words, representing each event as a histogram of 'visual words', which essential are clusters of spatio-temporal features capturing similar local motion (e.g., [10, 11, 33]). This approach does not require background subtraction or tracking, and it has been confirmed to be robust to noisy inputs and illumination changes. For all three recognition approaches, a sliding window technique was applied to detect events from continuous videos.

We used the random validation setting where we randomly select 30 video segments (i.e., 5 occurrences per event) as training data and use the other videos as testing data. Since this training/testing split contains randomness (i.e., they are randomly selected), we performed this validation process for 50 rounds and averaged their results. For each round, the event analysis component takes advantage of the given training examples to learn the spatio-temporal decision tree classifiers. In order to confirm the incremental property of our learning, the training videos have been provided to the system sequentially. Once trained, our recognition approach was applied to the remaining testing videos.

We measured the performances of recognition systems in terms of *precision* and *recall* values. Precision is defined as $tp/(tp + fp)$ where tp is the number of true positives and fp is the number of false positive, and recall is defined as $tp/(tp + fn)$ where fn is the number of false negatives. In general, these precision and recall values change as the system changes the detection threshold, and we are able to obtain precision-recall curves (PR curves) for each event.

More specifically, we obtained (1) average F-measures and (2) average precision-recall graphs of the recognition approaches. F-measure is a performance evaluation measure often used in the data retrieval area and is defined as:

$$F_\beta = (1 + \beta^2) \cdot \frac{pr \cdot rc}{\beta^2 \cdot pr + rc} \qquad (15)$$

where pr is the precision and rc is the recall value. We used

Table 5: Recognition accuracy (i.e., F-measure) of the three event detection approaches.

Events	ST feature	Rel. tree	**Prob. rel.**
Long stopping	0.726	1.000	1.000
Overtake	0.559	0.596	0.611
Overtaken	0.448	0.640	0.666
Sudden acceleration	0.497	0.833	0.841
Sudden stop - pedestrian	0.342	0.766	0.808
Sudden stop - vehicle	0.536	0.783	0.866
Total	0.518	0.770	**0.799**

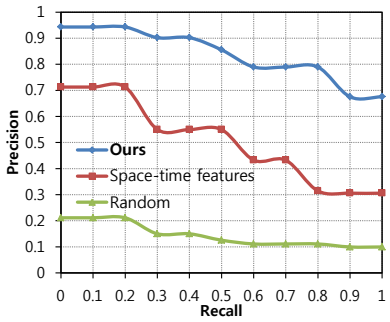


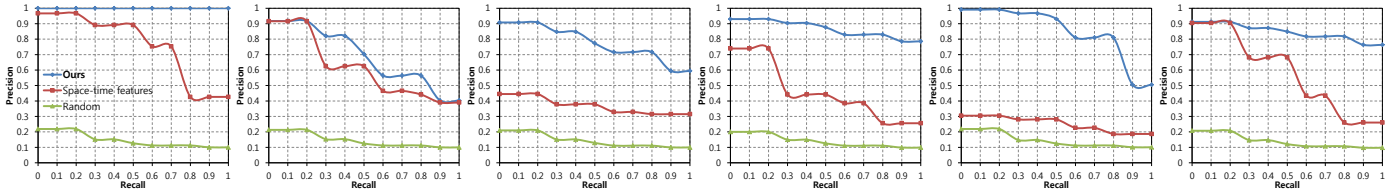
Figure 16: Average precision-recall curves comparing driving event detection performances of the recognition approaches. A 'random selection' strategy was also tested to illustrate the advantages of our proposed approach.

F_1 score where $\beta = 1$. Since the deterministic decision tree only provides a single pair of [precision, recall] values, it was only used for computing its F-measure (i.e., it provides a single point in the precision-recall dimension, not a curve).

Table 5 compares F-measures of the three driving event recognition approaches. We are able to observe that our approaches that explicitly considers vehicle/pedestrian trajectories obtain superior performance compared to the baseline approach using spatio-temporal features. Also, we are able to clearly observe the benefits of the probabilistic recognition: it compensates for noisy observation errors, resulting better performance.

Similarly, Figure 16 illustrates average precision-recall curves of the event detection approaches. We are able to observe that our system successfully annotates ongoing events in continuous video streams, reliably constructing appropriate personal driving diaries. By explicitly estimating the trajectory of the driving vehicle as well as those of appearing pedestrians and vehicles, our approach was able to extract meaningful features from videos and annotate semantic events considering spatio-temporal relations among the sub-events probabilistically. Figure 17 shows average PR curves of the system per event. Overall, our approach showed better results compared to the baseline approach following the concept of bag-of-words of local space-time features, which only focus on a set of local motion while ignoring other information.

In Figure 18, our system interface describing retrieved videos, relative locations of the vehicle on the map, and pedestrian/vehicle trajectories are illustrated. In addition, example videos of important driving events annotated using our system is shown in Figure 19.



(a) Long stopping (b) Overtake (c) Overtaken (d) Sudden acceleration (e) Sudden stop - pedestrian (f) Sudden stop - vehicle

Figure 17: Average precision-recall curves of the systems obtained for each event class. We are able to observe that our approach outperforms the baseline approach using local spatio-temporal video features in all event classes.

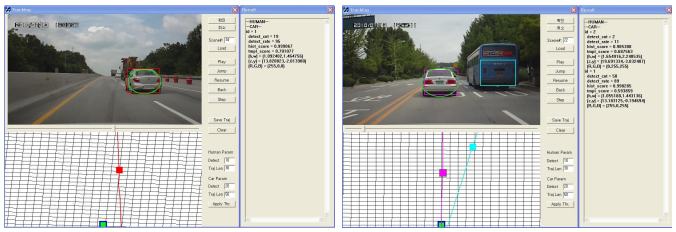


Figure 18: Video retrieval interface of our diary.

9. Conclusion

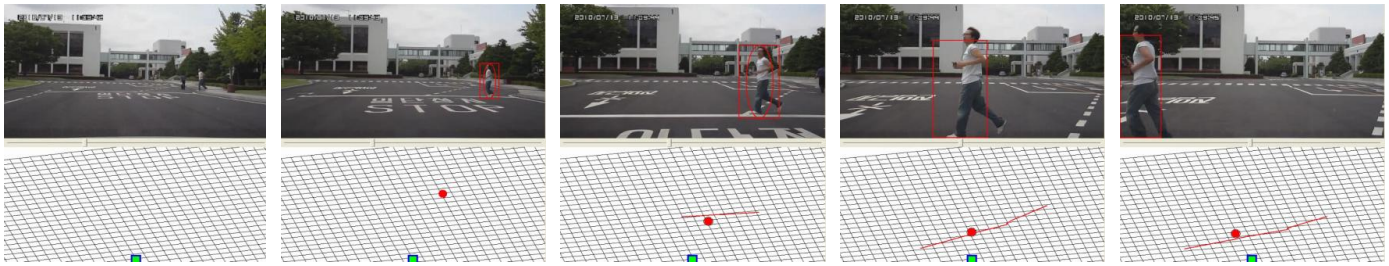
We introduced the concept of *personal driving diary*. We proposed a system that automatically constructs event-based annotations of driving videos, enabling efficient browsing and retrieval of users' driving experiences. The experimental results confirmed that our system reliably detects driving events from continuous video streams, generating a multimedia achieve of driving events. We were able to observe that our probabilistic approach explicitly considering spatio-temporal relations among vehicle/pedestrian trajectories obtains a superior performance compared to the state-of-the-art approach of utilizing bag-of-words of local spatio-temporal features. This event recognition approach will potentially benefit other types of life-logging/video-summarization systems that requires semantic analysis of object trajectories.

Acknowledgments

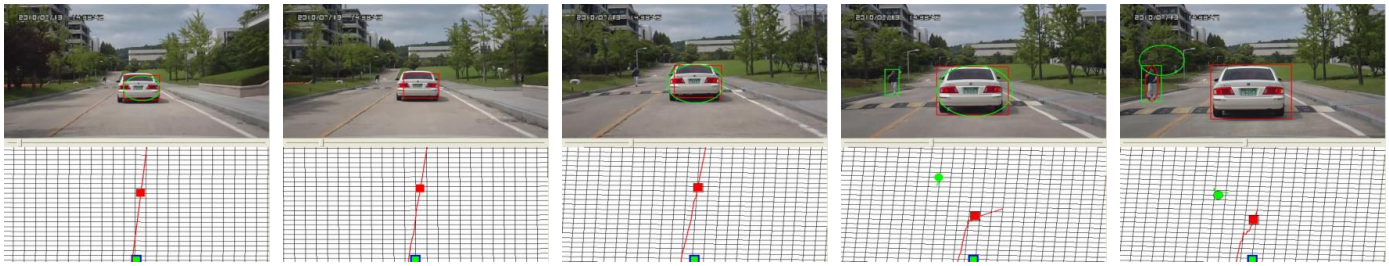
This work was supported partly by the R&D program of the Korea Ministry of Knowledge and Economy (MKE) and the Korea Evaluation Institute of Industrial Technology (KEIT) [The Development of Low-cost Autonomous Navigation Systems for a Robot Vehicle in Urban Environment, 10035354]. This work was partially supported by the R&D project of Hyundai Heavy Industry Co., Ltd. The writing of this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

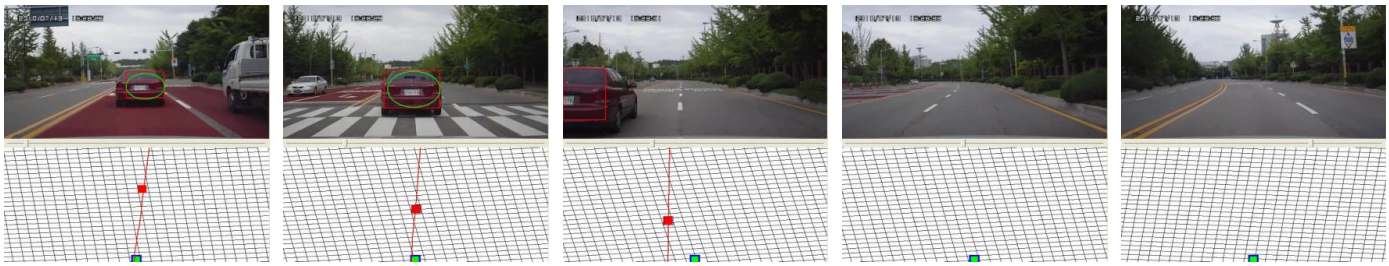
[1] US Patent 20040201697A1, "Black-box" video or still recorder for commercial and consumer vehicles (2004).
[2] T. Gandhi, M. M. Trivedi, Pedestrian protection systems: Issues, survey, and challenges, IEEE T ITS 8 (3) (2007) 413–430.
[3] M. S. Ryoo, J. Lee, J. H. Joung, S. Choi, W. Yu, Personal driving diary: Constructing a video archive of everyday driving events, in: WACV, 2011.
[4] T. Hori, K. Aizawa, Context-based video retrieval system for the life-log applications, in: ACM MIR, 2003.
[5] J. Gemmell, L. Williams, K. Wood, R. Lueder, G. Bell, Passive capture and ensuing issues for a personal lifetime store, in: ACM CARPE, in conjunction with ACM MM, 2004.
[6] A. R. Doherty, C. O. Conaire, M. Blighe, A. F. Smeaton, N. E. O'Connor, Combining image descriptors to effectively retrieve events from visual lifelogs, in: ACM MIR, 2008.
[7] A. J. Sellen, S. Whittaker, Beyond total capture: A constructive critique of lifelogging, Communications of the ACM 53 (5) (2010) 70–77.
[8] L. Liao, D. Fox, H. Kautz, Extracting places and activities from gps traces using hierarchical conditional random fields, International Journal of Robotics Research 26 (1) (2007) 119–134.
[9] J. K. Aggarwal, M. S. Ryoo, Human activity analysis: A review, ACM Computing Surveys 43 (2011) 16:1–16:43.
[10] C. Schuldt, I. Laptev, B. Caputo, Recognizing human actions: a local SVM approach, in: ICPR, 2004.
[11] P. Dollar, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, in: IEEE Workshop on VS-PETS, 2005.
[12] M. S. Ryoo, J. K. Aggarwal, Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities, in: ICCV, 2009.
[13] S. Hongeng, R. Nevatia, F. Bremond, Video-based event recognition: Activity representation and probabilistic recognition methods, Computer Vision and Image Understanding (CVIU) 96 (2) (2004) 129–162.
[14] M. S. Ryoo, J. K. Aggarwal, Semantic representation and recognition of continued and recursive human activities, International Journal of Computer Vision (IJCV) 32 (1) (2009) 1–24.
[15] K. M. Kitani, T. Okabe, Y. Sato, A. Sugimoto, Fast unsupervised ego-action learning for first-person sports videos, in: CVPR, 2011.
[16] A. Geiger, J. Ziegler, C. Stiller, StereoScan: Dense 3d reconstruction in real-time, in: IV, 2011.
[17] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Combridge, 2003.
[18] J. H. Joung, M. S. Ryoo, S. Choi, W. Yu, S. Kim, Reliable object detection and segmentation using inpainting, in: IROS, 2012.
[19] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part based models, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (9) (2010) 1627–1645.
[20] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: CVPR, 2001.
[21] B. Wu, R. Nevatia, Tracking of multiple, partially occluded humans based on static body part detection, in: CVPR, 2006.
[22] J.-Y. Lee, W. Yu, Moving object tracking in driving environment, in: International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2011.
[23] J.-Y. Lee, W. Yu, Visual tracking by partition-based histogram backprojection and maximum support criteria, in: IEEE International Conference on Robotics and Biomimetics (RoBio), 2011.
[24] J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (11) (1983) 832–843.
[25] N. C. Utgoff, P. E. abd Berkman, J. A. Clouse, Decision tree induction based on efficient tree restructuring, Machine Learning 29 (1997) 5–44.
[26] J. R. Quinlan, Induction of decision trees, in: Readings in Machine Learn-



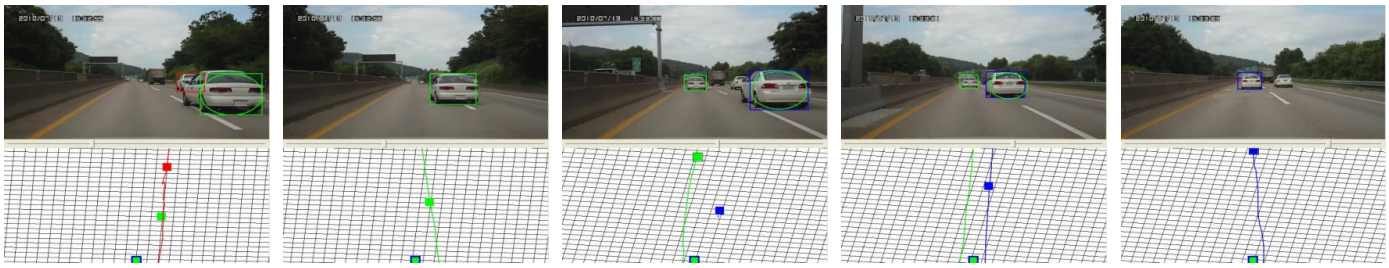
(a) A vehicle sudden stop event caused by a pedestrian



(b) A vehicle sudden stop event caused by another vehicle in the front



(c) A sequence of two other vehicles overtaking the driving vehicle



(d) A sequence of two vehicle overtaken event - two other vehicles are overtaking the driving vehicle

Figure 19: Example video sequences of annotated driving events. First-person view videos of various driving events are shown.

ing, Morgan Kaufmann, 1990, originally published in *Machine Learning* 1:81–106, 1986.

[27] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: CVPR, 2012.

[28] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, IJCV 88 (2) (2009) 303–338.

[29] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (5) (2003) 564–577.

[30] J. Kwon, K. Lee, Visual tracking decomposition, in: CVPR, 2010.

[31] Z. Kalal, J. Matas, K. Mikolajczyk, P-N learning: Bootstrapping binary classifiers by structural constraints, in: CVPR, 2010.

[32] T. B. Dinh, N. Vo, G. Medioni, Context tracker: Exploring supporters and distracters in unconstrained environments, in: CVPR, 2011.

[33] J. C. Niebles, H. Wang, L. Fei-Fei, Unsupervised learning of human action categories using spatial-temporal words, IJCV 79 (3) (2008) 299–318.