Recognizing Car Fluents from Video

Bo Li\textsuperscript{1,*}, Tianfu Wu\textsuperscript{2}, Caiming Xiong\textsuperscript{3,*} and Song-Chun Zhu\textsuperscript{2}
\textsuperscript{1}Beijing Lab of Intelligent Information Technology, Beijing Institute of Technology
\textsuperscript{2}Department of Statistics, University of California, Los Angeles
\textsuperscript{3}Metamind Inc.

bol86@bit.edu.cn, {tfwu, szhu}@stat.ucla.edu, cxxiong@metamind.io

Abstract

Physical fluents, a term originally used by Newton [40], refers to time-varying object states in dynamic scenes. In this paper, we are interested in inferring the fluents of vehicles from video. For example, a door (hood, trunk) is open or close through various actions, light is blinking to turn. Recognizing these fluents has broad applications, yet have received scant attention in the computer vision literature. Car fluent recognition entails a unified framework for car detection, car part localization and part status recognition, which is made difficult by large structural and appearance variations, low resolutions and occlusions. This paper learns a spatial-temporal And-Or hierarchical model to represent car fluents. The learning of this model is formulated under the latent structural SVM framework. Since there are no publicly related dataset, we collect and annotate a car fluent dataset consisting of car videos with diverse fluents. In experiments, the proposed method outperforms several highly related baseline methods in terms of car fluent recognition and car part localization.

1. Introduction

1.1. Motivation and Objective

The term of physical fluents is first introduced by Newton [40] to represent the time-varying states of object states. In the commonsense literature, it is defined as the specifically varied object status in a time sequence [39]. As a typical instance, car fluent recognition has applications in video surveillance and self-driving. In autonomous driving, car fluents are very important to infer the road conditions and the intents of other drivers. The left of Fig. 1 shows an example, when seeing the left-front door of a roadside car is opening, the autonomous car should slow down, pass with caution, or be ready to stop. In the freeway context, the autonomous car should give way when the front car is blinking to require a left or right lane change (see the middle figure in Fig. 1). In the parking lot scenario, car part status change indicates info for reasoning about human-car interactions and surveillance, e.g., a woman is opening car trunk to pick up stuff (see the right of Fig. 1). In general, fluents recognition is also essential in inferring the minds of human and robotics in Cognition, AI and Robotics.

While there is a vast literature in computer vision on vehicle detection and viewpoint estimation [16, 21, 45, 46, 63, 43, 25, 35, 12], reasoning about the time-varying states of objects (i.e., fluents) has been rarely studied. Car fluents recognition is a hard and unexplored problem. As illustrated in Fig. 2, car fluents have large structural and appearance variations, low resolutions, and severe occlusions, which present difficulties at multiple levels. In this paper, we address the following tasks in a hierarchical spatial-temporal And-Or Graph (ST-AOG).

- i) Detecting cars with different part statuses and occlusions (e.g., in Fig. 2 (1.b), the frontal-view jeep with hood being open and persons wandering in the front. In the popular car datasets (such as the PASCAL VOC car dataset [15] and KITTI car dataset [20]), most cars do not have open parts (e.g., open trunk/hood/door) together with car-person occlusions. So, a more expressive model is needed in detecting cars in car fluent recognition.

- ii) Localizing car parts which have low detectability as individual parts (e.g., the open hood and trunk in Fig. 2 (2.c), the tail lights in the last row of Fig. 2) or even are occluded (e.g., the trunk in Fig. 2 (2.b)). This situation is
results show that our model is capable of jointly recognizing semantic car fluents, part positions, and part status in video. It outperforms several highly related state-of-the-art baselines in fluent recognition and part localization (sliding window-based). Besides, our model can be directly incorporated with popular improved dense trajectories (iDT) [59] and C3D features [55] to encode more general information.

2. Related Work and Our Contributions

We briefly review the most related topics in 3 streams:

**Car Detection and View Estimation.** In computer vision and intelligent transportation, there is a considerable body of work on car detection and view estimation [16, 21, 45, 46, 63, 43, 54, 25, 35, 12]. Though those works have successfully improved the performance on popular benchmarks [15, 20], the output is usually the car bounding box and a quantized rough view. Some other works [53, 72, 65] aim to get car configurations with detailed or fine-grained output to describe the more meaningful car shape, rather than a bounding box. However, all of these works generally regarded car as a static rigid object, while pay little attention to the functionalities of semantic car parts and the fact that cars presenting large geometry and appearance transformations during car fluents change.

**Video-based Tasks in Vision, Cognition, AI and Robotics.** In computer vision, a significant effort has been devoted to video-based tasks: event [38, 28, 64], action detection/recognition [61, 73, 60, 41] and pose estimation [67, 1, 18]. These papers are related to our work, however, most of them are based on human models, while very little work has done on car fluents and related car part status estimation. In cognitive science, the concept of fluents is mainly used to represent the object status changing in time series [39], and it has been used in causality inference [39, 26, 7, 19]. Furthermore, fluents are also related to action-planning in AI and Robotics [5, 66].

**Human-Object Recognition.** As mentioned in [19], the status of object fluent change is caused by some other objects. In our case, the car fluent change is mainly conducted by humans. Thus our work is related to part recognition [49, 34] and human-object interactions [50, 33, 23, 68, 6, 51, 14]. But, here we jointly and explicitly model car detection, part localization and status estimation.

**Our Contributions.** This paper makes three contributions to car fluent recognition:

i) It presents a ST-AOG to represent the spatial-temporal context in car fluent recognition. Car fluent recognition is a new task in the computer vision literature.

ii) It presents a new car fluent dataset with per-frame annotations of cars, car types, semantic parts, part statuses and viewpoints.

iii) It provides a new method for car fluent recognition and part status estimation, and outperforms several state-of-the-art baseline methods.
3. Representation

3.1. Spatial-Temporal And-Or Graph

Fig. 3 illustrates our ST-AOG, it models a car fluent by the viewpoint, car type, semantic parts and corresponding part statuses hierarchically. In addition to the appearance and deformation, it also models the temporal transitions of the semantic parts.

Utilizing $I_i$ and $I_{i+1}$ to denote neighbouring frames, the ST-AOG is defined by a 3-tuple,

$$\mathcal{G} = (V, E, \Theta)$$  \hspace{1cm} (1)

where $V$ is the node set consisting of a set of non-terminal nodes (i.e., And-nodes $V_{And}$ and Or-nodes $V_{Or}$) and a set $V_T$ of terminal-nodes (i.e., appearance templates), $E = \{E_s, E_M\}$ is the edge set maintaining the structure of the graph. Specifically, $E_s$ is the spacial edges accounting for composition and deformation, $E_M$ is the lateral edges accounting for temporal transitions. $\Theta = (\Theta_{app}, \Theta_{def}, \Theta_{M}, \Theta_{bias})$ is the set of parameters, and we have,

i) the appearance parameters $\Theta_{app} = (\Theta_{app}^t, t \in V_T)$ consists of the appearance templates for all terminal-nodes. Our model is general, in which any type of features can be used. In this paper, we use the histogram of oriented gradient (HOG) feature [13] and convolutional neural network (CNN) feature [32] as the appearance feature for illustration.

ii) the deformation parameters $\Theta_{def}$ are defined for all in-edges of all terminal-nodes, $(v, t) \in E (v \in V_{And} \cup V_{Or} \text{ and } t \in V_T)$, to capture the local deformation when placing a terminal-node $t$ in a feature pyramid. For the deformation feature, we adopt the standard separable quadratic function of the displacement of $t$ with respect to its parent node $v$, $(dx, dx^2, dy, dy^2)$, as done in the deformable part-based models [16, 2] and the car AOG model [37].

iii) the transition parameters $\Theta_M$ are defined on the single car and semantic part nodes. They are used to weight the effects of temporal transition between $I_i$ and $I_{i+1}$. As illustrated on the right of Fig. 3, we compute 3 types of motion flows for car fluents:

a) global motion flow. We compute the optical flow of the whole car (e.g., body), and utilize it as the global feature to localize car in time. This global feature is very helpful for moving cars, especially in the far view, in which the semantic part is too small to be detected.

b) local motion flow. We compute the optical flow of “mobile” car part (e.g., door), and utilize it as the temporal feature to recognize the position and status of the part. Here, the “mobile” car part is the part that can be open or closed, which has motion features during fluent change.

c) iconic motion flow. For car lights, the most distinctive feature is its intensity change and temporally periodical turn on/off statuses. This is related to the work in iconic change [3, 4] and change detection [47]. Here, we use the appearance and optical flow templates to localize car lights, and compute the temporal intensities (vector of normalized intensities) to recognize the periodical light status change (e.g., left-turn/right turn).

For optical flow, we compute the pyramid feature to cope with different scales.

iv) the bias parameters $\Theta_{bias}$ are defined for the child nodes of all Or-nodes to encode either the prior (e.g., different appearance templates for the “open” part), or the compatibility of parsing scores among different sub-categories (e.g., different viewpoints of a jeep).
Although the hierarchical structure of our ST-AOG is similar to those spatial-only models used in [37, 46] for car detection, we introduce the semantic part Or-nodes (to be used to define fluent) for detailed part status modelling in both spatial and temporal domains.

A parse tree, pt, is an instantiation of the ST-AOG being placed at a location in the spatial-temporal feature pyramid. It is computed by following the breadth-first-search order of the nodes in G, selecting the best child node at each encountered Or-node (based on the scoring function to be defined later on). The bold black edges in Fig. 3 show the parse trees of ST-AOG on three neighbouring video frames. For a video with N frames, we can get its parse graph as pg = \{pt^{i,i+1}\}_{i=1}^{N-1}.

Based on the pg, we extract frame-level part bounding boxes and statuses, and utilize them to design spatial-temporal features for fluent recognition. To capture long-term spatial-temporal info, our model can also integrate iDT [59] and C3D features [55] (see Section 6.2).

### 3.2. The Scoring Function

The scoring function of ST-AOG is recursively defined w.r.t. its structure. For simplicity, we will use v and \( \vec{v} \) (\( v \in V \)) to represent the temporally neighbouring nodes (on frames \( I_i \) and \( I_{i+1} \)) below.

Let \( O \in V_{Or} \) denotes the Or-node in the ST-AOG, \( A \in V_{And} \) be the parent node of a terminal-node \( t \in V_T \). We model \( t \) by a 4-tuple \( (\theta^{pp}_t, \theta^{dft}_t, \sigma_t, a|A\) where \( \sigma_t \) is the scale factor for placing \( t \) w.r.t. \( A \) in the feature pyramid (\( \sigma_t \in \{0,1\} \)), and \( a|A \) is the anchor position of \( t \) relative to \( A \).

i) Given \( A, \vec{t} \) and their locations \( l_A, l_t \), the scoring function of placing \( t \) at the position \( l_t \) is then defined by,

\[
S(I_t|A, l_A) = \max_{l_t} [\theta^{pp}_t, \Phi^{opp}(l_t) - l_t, l_t] < \theta^{dft}_t, \Phi^{dft}(l_t, l_A) > + \theta^A_M ||l_t - l_A + F(l_t)||^2_2
\]

where \( \Phi^{opp}(l_t) \) is the appearance features (HOG or CNN) and \( \Phi^{dft}(l_t, l_A) \) is the deformation features. \( \theta^A_M \) is the motion flow weight of \( t \). \( F(l) \) is the motion flow between frames \( I_i \) and \( I_{i+1} \) computed at position \( l \).

ii) Given \( A \) and its location \( l_A \) at frame \( I_{i+1} \), the scoring function of \( A \) is defined by,

\[
S(O, I_{i+1}, I_A) = \sum_{v \in ch(A)} [S(v|A, l_A) + b_A] + \theta^A_M ||l_A - l_v|A + F(l_v)| ||^2_2
\]

where \( ch(A) \) is the set of child nodes of \( A \), \( b_A \) is the bias parameter. \( \theta^A_M \) is the motion flow weight of \( A \).

iii) Given \( O \) and its location \( l_O \) at frame \( I_{i+1} \), the scoring function of \( O \) is defined by,

\[
S(O, l_{i+1}|O, l_O) = \max_{v \in ch(O)} [S(v|O, l_O) + \theta^O_M ||l_O - l_v|O + F(l_v)| ||^2_2]
\]

where \( ch(O) \) is the set of child nodes of \( O \), \( l_O \) is the position of \( O \). \( \theta^O_M \) is the optical flow weight of \( O \).

For temporal term \( ||l_v - l_O + F(l_v)||^2_2, v \in V \), since the scoring function of \( v \) is conditioned on \( \vec{v} \) in time, while \( \vec{v} \) is conditioned on its child nodes or parent node in space, there are loops in this structure, and we couldn’t use DP to compute above scoring functions directly. As similar to [11], we resort to LBP [62] to get an approximate solution. For computational efficiency, the motion flows are computed only at nodes corresponds to the car and its semantic parts. For other nodes, their motion flow are implicitly embedded, as their score maps are related to nodes with temporal links, and their scoring functions are computed as above but without the temporal terms.

Given a fluent video clip with \( N \) frames, the overall score of ST-AOG will be:

\[
S(I_{1:N}|pg, G) = \frac{1}{N - 1} \sum_{i=1}^{N-1} S^{i,i+1}(O, l_O|\vec{O}, l_{\vec{O}}) \tag{5}
\]

where \( O \) is the root Or-node in ST-AOG. In probabilistic model, Eqn. (5) can be interpreted as a log-posterior probability up to a constant. Our objective is to maximize Eqn. (5), given all appearance, deformation, and motion features from training videos.

### 4. Inference by LBP and DP

To cope with the loop introduced by motion transition, we extend the traditional inference procedure in AOG [37, 69] with LBP. Given an input video, Our inference procedure includes 4 steps:

i) For each frame \( I_i \), we omit the temporal links between it and its neighbour frames, and compute the appearance feature pyramid, optical flow pyramid, and score maps for all nodes in layer \( 3 \) to \( 6 \) in the ST-AOG by the Depth-First-Search (DFS) order; This step is similar to the inference algorithm in [37].

ii) After we get the score maps of the semantic part Or-nodes and the single car And-nodes, we further integrate the score maps with optical flow maps, as can be seen on the left of Fig. 4. For each semantic part \( p \) with its parent root node \( r \), we focus on four nodes, i.e., \( p, r, \vec{p}, \vec{r} \), and omit other spatial links connected to them. At the beginning, we send message from \( p \) to \( r \), then from \( r \) to the rest to update the message. When the difference of the propagated message in two consecutive iterations doesn’t change, we compute the last “belief” that transmitted to \( r \) as \( r' \)'s score map. This procedure can be efficiently implemented by distance transform [17].

iii) After we get the score maps from ii), we further compute the score maps for nodes in the upper layers of ST-AOG. By this procedure, the score map of the root Or-node for neighbouring frames \( I_i \) and \( I_{i+1} \) can be computed by maximizing Eqn. (4) on each spatial-temporal point.
iv) For each spatial-temporal sliding window with score being greater than the threshold \( \tau \), we follow the Breadth-First-Search (BFS) order to retrieve its parse tree (including the whole car windows, semantic part windows and part statuses at each frame) by taking \( \text{argmax} \) at each encountered Or-node.

### 4.1. Post-Refinement

As analysed in [44], we can get more accurate part localizations by generating multiple object instances in an image. In this paper, we utilize a similar method to generate multiple car part proposals: First of all, we set low detection threshold to generate multiple car proposals around the target car, then we execute a conservative non-maximum suppression (NMS) procedure to override the observation term and select windows that are not local maxima. By backtracing the parse tree \( pt \), we can get the semantic part proposals from each car proposal.

The right of Fig. 4 illustrates our part-based HMM inference method. For a semantic part \( p \), we can write its bounding box sequence in a given video (\( N \) frames) as a stochastic set \( \Omega_p = \{ p^1, \ldots, p^{N-1} \} \), then the overall linking scores of part \( p \) in a video can be computed as:

\[
S(\Omega_p) = \sum_{i=1}^{N-1} S(p^i) + \theta_p \psi(p^{i+1}) + \lambda \sigma(p^i, p^{i+1})
\]

\[
S(p^i) = \theta_p \phi_p (I_i) + \theta_p \phi_p (I_{i+1}) \psi(p^{i+1}) = (dx, dz, dx, dz, dx, dz)
\]

where we model \( S(p^i) \) as the appearance and deformation score, \( \theta_p \) \& \( \phi_p \) are the corresponding parameter and feature, \( \theta_p \phi_p (I_i) \) \& \( \theta_p \phi_p (I_{i+1}) \psi(p^{i+1}) \) is the bounding box overlap of \( p \) between \( I_i \) and \( I_{i+1} \). For each semantic part in the video, we seek the optimal path \( \Omega_p^* \) by maximizing Eqn. (6).

### 5. Learning by Latent Structural SVM

The training data. In our car fluent dataset, for each video \( X_v, v \in \{1, \ldots, W\} \), we have its annotated parse graph \( pg(X_v) \) including the car bounding boxes, viewpoints, car types, semantic part bounding boxes, part statuses and corresponding fluent labels.

For parameter optimization, the objective function of ST-AOG is:

\[
\min_{\Theta, \xi} \quad \mathcal{E}(\Theta) = \frac{1}{2} \|\Theta\|^2 + C \sum_{v=1}^{W} \xi_v \quad \text{s.t.} \quad \xi_v > 0; \quad \forall_v, \quad \bar{pg}(X_v) \neq pg(X_v),
\]

\[
S(pg(X_v)) - S(\bar{pg}(X_v)) \geq L(pg(X_v), \bar{pg}(X_v)) - \xi_v
\]

where \( C \) is the regularization parameter, \( \bar{pg} \) is the predicted parse graph, and \( L(pg(X_v), \bar{pg}(X_v)) \) is the surrogate loss function. To accurately predict the viewpoint, car type, part bounding boxes and part statuses, we specify \( L(pg(X_v), \bar{pg}(X_v)) = \frac{1}{vN} \sum_{j=1}^{vN} \ell(pt(X_j), \hat{pt}(X_j)) \) (\( vN \) is the number of frames in \( X_v \)), and define \( \ell(pt, \hat{pt}) \) by,

\[
\ell(pt, \hat{pt}) = \begin{cases} 
  1 & \text{if view or car type is different} \\
  1 & \text{if overlap}(pt, \hat{pt}) < 0.3 \\
  1 & \text{if } \text{pt} \text{ status}(p) \neq \text{pt} \text{ status}(p) \\
   0 & \text{otherwise}
\end{cases}
\]

where the second term computes the overlap between two parse trees (they have the similar structure, otherwise the loss is computed based on the first term), and \( \text{overlap}(pt, \hat{pt}) \) is defined by the minimum overlap of bounding boxes of all nodes in the parse trees. The third term check the semantic part status.

The training procedure has the following two steps:

i) **Initialization**. We use standard SVM to train the semantic car part templates on the annotations. For each view, we first cluster the semantic parts by their status, then cluster the “open” parts by their aspect ratios, as they have large appearance and geometry variations. After getting these clusters, we train a template for each of them. These templates construct the Terminal-nodes in ST-AOG.

ii) **Learning the whole ST-AOG** under the LSSVM. As the groundtruth annotations may not be the most discriminative, we allow parts to move around but with a certain overlap with the groundtruth. Thus the optimization becomes a latent structural problem. We iterate the following two steps to get a local optima:

#### Step 1 - Positives Relabelling: Do loss-augmented inference for each training video with current model by:

\[
\Omega_p^* = \text{argmax} \sum_{p=1}^{P} S(\Omega_p) + L(pg(X_v), \bar{pg}(X_v))
\]

#### Step 2 - Update Weights: With the inferred part bounding box and part status sequences, we use standard Structural SVM to optimize the appearance, deformation, and motion parameters \( \Theta \) in ST-AOG.

In our implementation, we start with a small number of negative examples, and iteratively increase them in the 2nd step as the iCCCP method [70]. Besides, we also maintain a cache for hard negative mining. Since computing on every frame is not necessary, we slide the spatial-temporal window by every 3 frames.
6. Experiments

6.1. Car-Fluent Dataset

The introduced Car-Fluent dataset includes 647 video clips, containing basically 16 semantic car part fluents with multiple car types, camera viewpoints and occlusion conditions. All the fluents and their population ratios are shown on the left of Fig. 6. The videos in Car-Fluent dataset are collected from various sources (youtube, movies, VIRAT [42], etc.), and captured by both static cameras and moving cameras. Based on the number of semantic fluent instances in each video, we split these videos into 2 sets: the first set consists of 507 videos where each video contains a single instance of each car fluent. We randomly select 280 videos in this dataset for training, and the rest for testing; The second split consists of 140 videos that are just used for testing, in which each video is much longer and has multiple instances of diverse kinds of car fluents. All the videos are used for three important tasks: 1) semantic fluent recognition (e.g. opening hood, change left lane), 2) car part localization (e.g. where is the hood? Where is the light?), and 3) car part status estimation (e.g., is the trunk open?).

All videos are annotated with viewpoints, car types, bounding boxes of both whole car and semantic parts, and part statuses. The type of part status are “open/on”, “close/off”, and “occluded”. In this paper, we focus on “semantic” and “functional” parts, all these parts are related to the fluents listed in Fig. 6. We adopt VATIC [57] to annotate videos on the introduced dataset. 7 workers are hired to clip the sequence of semantic car fluents and annotate the videos. Three independent annotations were collected for each video and combined via outlier rejection and averaging. It took them about 500 hours in total. Finally, all labelling results were thoroughly checked by the authors. For more details, please refer to our supplementary.

6.2. Fluent Recognition

To verify our ST-AOG on fluent recognition, we compare it with space-time interest points (STIP) [31], improved dense trajectory (iDT) [59], and the deep learning based C3D method [55]. For STIP and iDT, we use the softwares in [31, 59] to extract corresponding features inside the car bounding boxes from each video, and try various methods to construct the video descriptors. For the bag-of-words (BoW) method, we try different size of codebooks (i.e., 400, 800, 1000, 1200, 1600, 2000), and select the best ones for iDT and STIP. For fisher vectors (FV) and VLAD vectors, we use 256 components for as common choices in [64, 52, 29]. For feature dimension reduction, we use PCA, and experiment with different reduced dimensions, i.e., 1, 4, 1/3, 1/2, 1 of the original dimension, and utilize the reduced dimensions that best balance the performance and storage cost in corresponding features, i.e., half of the original dimension for both STIP and iDT.

For C3D [55], since we don’t have enough videos to train a deep network, we first pre-train a network on the Sport1M dataset [30] and VIRAT dataset [42]. Then we finetuned a model on the Car-Fluent dataset. As in [55], we extract fc6 features from each video and use the mean features as the video descriptors. We also tried computing the FV and VLAD vectors for C3D method, but get low performance.

For our ST-AOG, based on the predicted semantic part positions and part statuses extracted from parse graph pg, we compute temporal part status (TPS) features as follows:

\[
\phi_T = \{\phi_1(p^i_s), \phi_2(p^i_s, p^{i+1}_s), d(p^i_s, p^{i+1}_s)\}_{i=1}^{N-1}
\]

where \(\phi_1(s) = [0, \ldots, 1(s), \ldots, 0]^{T}\) is a \(z_p\) dimension vector, here \(z_p\) is the number of statuses of part \(p\), \(\phi_2(s^i, s^{i+1}) = [0, \ldots, 1(s^i), 1(s^{i+1}), \ldots, 0]^{T}\) is a \(z_p \times z_p\) dimension vector, \(d(p^i_s, p^{i+1}_s)\) is the temporal transition of
Figure 7. Examples of part status estimation results on our Car-Fluent dataset. For good visualization, we cropped the cars from original images. Different semantic parts are shown by rectangles in different colors (solid for “close/off” status and dashed for “open/on” status). The black lines are used to illustrate variations of the relative positions between whole car and car parts with different statuses. The top 2 rows show successful examples. The bottom row shows failure cases, which are mainly caused by occlusion and mis-detected car view. by failure, we mean some part statuses are wrong (e.g., for the 3rd example, the door is correct, but the hood and trunk are wrong). Best viewed in color and magnification.

<table>
<thead>
<tr>
<th>Baselines</th>
<th>STIF-BoW</th>
<th>iDT-BoW</th>
<th>STIF-FV</th>
<th>STIF-VLAD</th>
<th>iDT-FV</th>
<th>iDT-VLAD</th>
<th>CID</th>
<th>TPS</th>
<th>TPS+DT</th>
<th>TPS+DT+CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP</td>
<td>25.8</td>
<td>35.1</td>
<td>35.2</td>
<td>41.6</td>
<td>41.7</td>
<td>41.3</td>
<td>31.0</td>
<td>34.4</td>
<td>45.0</td>
<td>50.8</td>
</tr>
</tbody>
</table>

Table 1. Results of baseline models and our ST-AOG in terms of mean precision (MP) on Car-Fluent Recognition.

part $p$ between neighboring frames. For each car light, we also compute the mean intensity difference over the light bounding box across neighboring frames, and append it to $\varphi_T$. We compute FV and VLAD vectors on TPS to construct the video descriptors as STIF and iDT. For our TPS, we get the best performance with 53 reduced feature dimensions for PCA and 256 centers for VLAD vectors.

We use the one-against-rest approach to train multi-class SVM classifiers for above methods. For evaluation, we plot confusion matrix, and report the mean precision (MP) in fluent recognition, equivalent to the average over the diagonal of the confusion matrix.

The first 6 columns in Table 1 show the MP of above methods. As can be seen, the “iDT-VLAD” get the best performance, while C3D and our TPS get relatively lower performances on this dataset, which verified the importance of motion features. For C3D, we think the relatively low performance is due to the lack of training data and it doesn’t take advantage of car positions. As TPS, iDT and C3D belong to different type of features, we augment our TPS by integrating it with iDT and C3D features. To this end, we first combine our TPS with iDT, specifically, we try various combinations of FV and VLAD vectors, and find the best one is concatenating TPS VLAD vectors and iDT VLAD vectors with square root normalization, as can be seen in Table 1, this improves the performance by about 4; then we further combine TPS and iDT VLAD vectors with C3D feature, as can be seen in the last column, we improve the performance to 50.8, which outperforms the best single feature by 9.5. From these results we can see, ST-AOG provides important cues in car fluent recognition, and it’s very general to integrate with traditional motion features and deep learning features.

The middle of Fig. 6 shows the confusion matrix of our “TPS+iDT+C3D” method. We can see the merged feature performs pretty well on fluents, i.e., “Open/Closed Hood”, “Change Left/Right Lane”, “Left/Right Turn”, but totally miss fluents, e.g., “Open/Closed LB Door”, “Close RB Door”. For fluents related to the car back door, the low performance may be attributed to their ambiguous features and relatively low population ratios on current dataset.

6.3. Part Localization and Part Status Estimation

Since part location and status are important to fluent recognition, in this experiment, we further examine the frame-level part localization and status estimation results.

6.3.1 Part Localization

For semantic part localization, we use following baselines: (i) *Strongly supervised deformable part-based model (SS-DPM)* [2]. SSDPM is an open source software, it achieved state-of-the-art performance on localizing semantic parts of animals on PASCAL VOC 2007 dataset [15]. (ii) *And-Or Structures* [36]. In the literature, [36] also attempted to localize the semantic parts by replacing the latent ones in DPM. (iii) *Deep DPM (DP-DPM)* [22]. The authors in [22] argued that by replacing HOG features with Deep CNN features can improve the detection performance of original DPM [16]. We use the same trick as [36] to add semantic parts on DP-DPM. We also try to compare our model with [67, 27, 8, 9], but find it’s hard to compare with them,
For our ST-AOG, we use several variants for comparison. 1) "ST-AOG-HOG" is the instance of our ST-AOG based on HOG features. 2) "ST-AOG-HOG-CSC" is the cascade version of our model. To improve the speed, we use the recently developed Faster-RCNN [48] for cascade detection of cars. Then based on these car detections, we further run the ST-AOG for part localization. 3) "ST-AOG-CNN" is the instance of our ST-AOG based on the pyramid of CNN features, here, we use the max3 layer of the CNN architecture for comparison as [22]. To cope with small cars, we double the size of source images for CNN.

For evaluation, we compute the detection rate of the semantic parts (e.g., "left-front door", "hood"). Here, we assume the whole car bounding boxes are given, and each method only outputs 1 prediction with the highest confidence for each groundtruth car. Table 2 shows the quantitative results. We can see on almost all semantic parts, our model outperforms baseline models by a large margin. We believe this is because ST-AOG jointly model the view, part occlusion, geometry and appearance variation, and temporal shifts. For "ST-AOG-HOG-CSC", we just use it as a reference to improve the speed of ST-AOG, since it has the nearly real-time speed, and can be plugged in our model as an outlier-rejection module. Surprisingly, CNN features perform worse than HOG features on the Car-Fluents dataset. In experiments, we find the extracted CNN feature on deep pyramid [22] is too coarse (the cell size is 16), even when we resize original images by 2 times, it still miss many small parts. Based on recent study of using CNN for key-points prediction [56], we believe a more effective method is required to integrate CNN feature with graphical models.

### 6.3.2 Part Status Estimation

To the best of our knowledge, no model tried to output car part status in our case in the literature. To evaluate the part status, we compare different versions of our model and analyse the effects of different components. Specifically, we compute the part status detection rates, that is, a part status is correct if and only if its position and status (e.g., open, close) are both correct.

On the right of Fig. 6, we show the quantitative results of 4 versions of our ST-AOG. Here, we use HOG features for simplicity. “ST-AOG-ver1” refers to the ST-AOG without both view and part status penalty in Eqn. (8), “ST-AOG-ver2” refers to the ST-AOG without view penalty in Eqn. (8), and “ST-AOG” is the full model with all penalty in Eqn. (8). To investigate the effect of motion flow, we also compare the ST-AOG without motion flow, i.e., the “S-AOG”. As can be seen, the view penalty is the most important factor, then is the part status and motion flow, and part status seems to have more importance. Interestingly, for some parts (e.g., “RB-Door”), adding more penalty or temporal info will decrease the performance.

Fig. 7 shows some qualitative running examples of ST-AOG. The first two rows show the successful results, the last row shows the failure examples. For better visualization, we cropped the image to just contain the interested car. We can see our model can localize parts and recognize their statuses fairly good with different viewpoints and car types, but may report wrong results when people occlude the car, the viewpoints are mis-detected, or there are background clutters.

### 7. Conclusion

This paper proposed a ST-AOG to recognize car fluents at semantic part-level from video. The ST-AOG integrates the motion features in temporal, and a LBP-based DP method can be used in inference. The model parameters are learned under the latent structural SVM (LSSVM) framework. To promote the research of fluents, we have collected a new Car-Fluents dataset with detailed annotations of viewpoints, car types, part bounding boxes, and part statuses. This dataset presents new challenges to computer vision, and complements existing benchmarks. Our experiments verified the ability of proposed ST-AOG on fluent recognition, part localization, and part status estimation.

In future work, we will integrate human and car jointly, and study human-car interactions to help understand human actions/intents based on car fluent recognition.

**Acknowledgement:** B. Li is supported by China 973 Program under Grant NO. 2012CB316300. T.F. Wu and S.C. Zhu are supported by DARPA SIMPLEX Award N66001-15-C-4035, ONR MURI project N00014-16-1-2007, and NSF IIS 1423305. We thank Tianmin Shu, Bruce Xiaohan Nie and Dan Xie for helpful discussions.
References


