

# Semantic Part Detection via Matching: Learning to Generalize to Novel Viewpoints from Limited Training Data

Yutong Bai\*, Qing Liu\*, Lingxi Xie, Weichao Qiu, Yan Zheng, Alan Yuille

# Problem: Semantic Part Detection

back arc door edge window (lower edge) wiper window (middle edge)



rear plate wheel side step wheel bumper wheel door wheel

Two examples of annotated semantic parts in the class car.

How to design this algorithm, which can learn from **limited training samples** and **generalized to novel viewpoints**.

# Motivation: why is this problem hard?

Why using limited training data?

Detecting semantic parts of an object is a challenging task, particularly because it is hard to annotate semantic parts and construct large datasets.

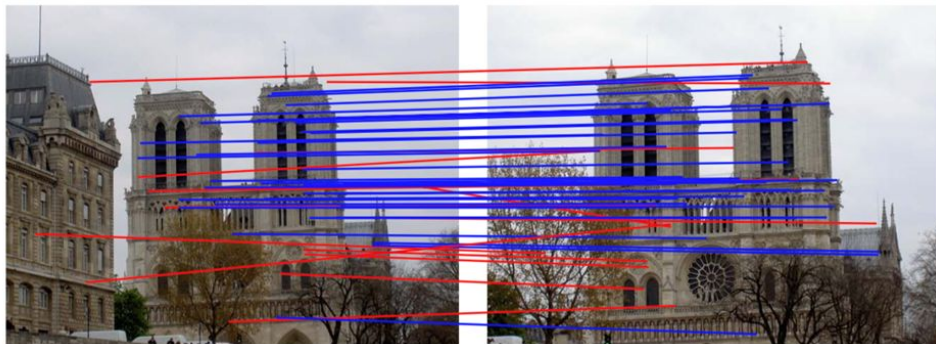
Why is this problem hard?

1. Objects with different viewpoints
2. Objects with rare viewpoints
3. Objects/parts with occlusion

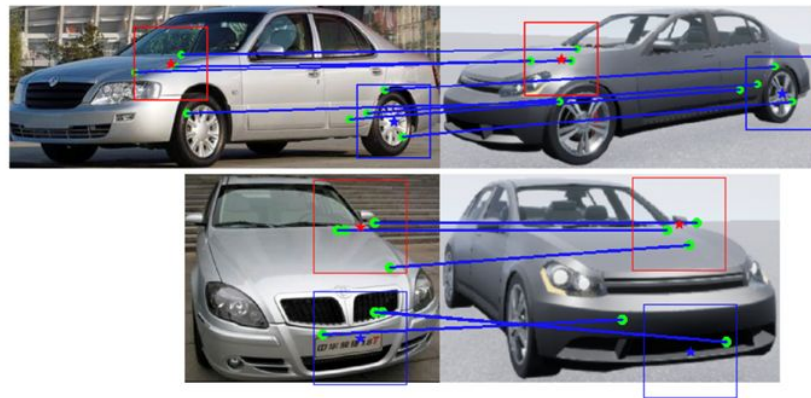


# Viewpoints are different

Matching algorithm using low-level feature



Matching algorithm using deep feature

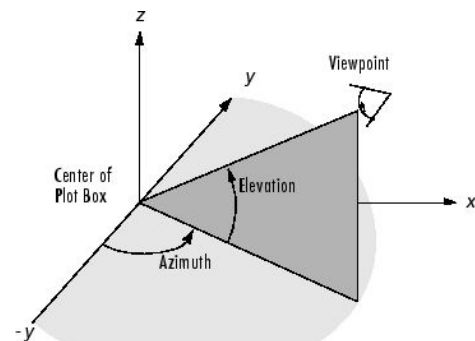
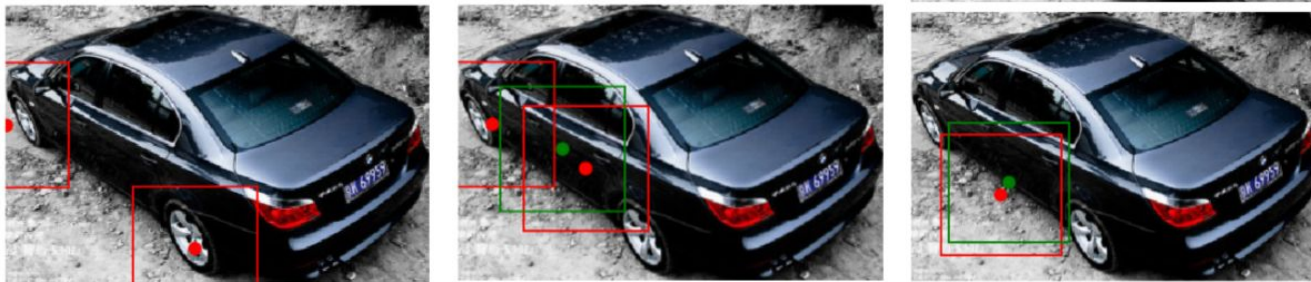


Here we are demonstrating matching based on feature similarity.

Objects viewed from slightly different angles can be matched based on feature similarity, but the algorithm will fail when the viewpoints vary a lot.

Our algorithm deals with the cross-domain problem. (Real to synthetic)

# Viewpoints are different



Up: a testing image with transferred annotations (red) and ground-truth (green). (The elevation is 20 degrees, which is a rear viewpoint.)

It is worth noting that the wheel has been completely self-occluded in the elevation angle of 20 degrees, but our model can still detect it. (The annotation for wheels is missing in this image in the dataset.)

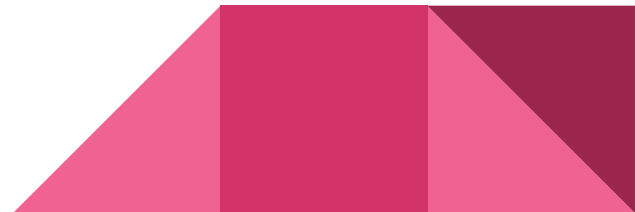
# Objects/parts are occluded



Examples of occlusion levels L0–L3 (left to right).

Also, occlusion makes it even harder.

New game: where's the wheels?



# Problem: Semantic Part Detection

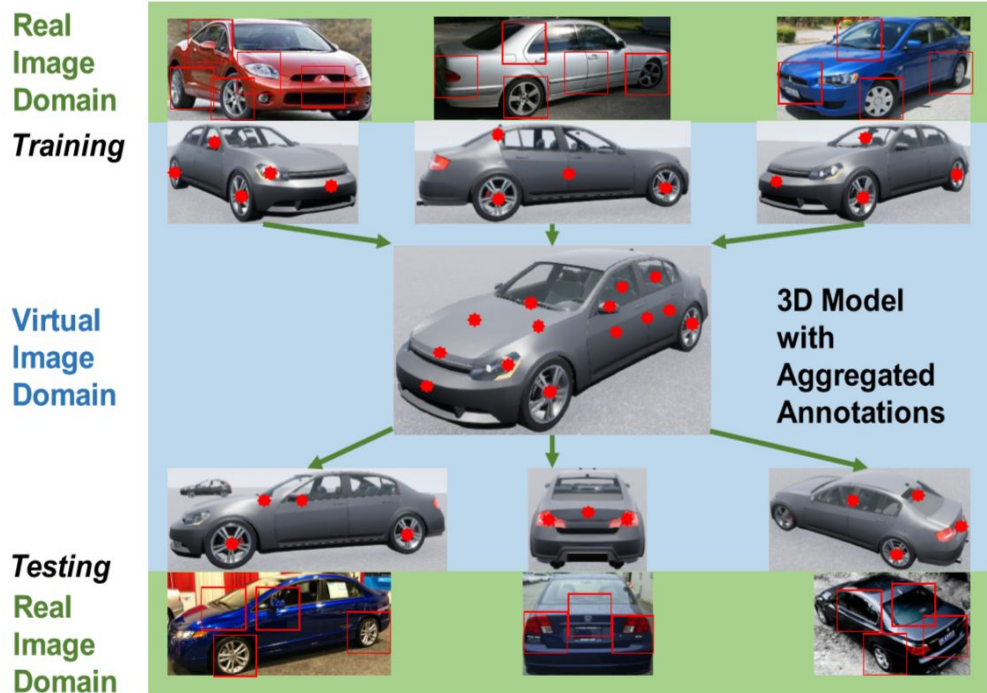
back arc door edge window (lower edge) wiper window (middle edge)



rear plate wheel side step wheel bumper wheel door wheel

Goal: an algorithm that can learn from limited training samples and generalized to novel viewpoints, meanwhile having the ability to deal with occlusions in an explainable way.

# Framework: Detection by Matching

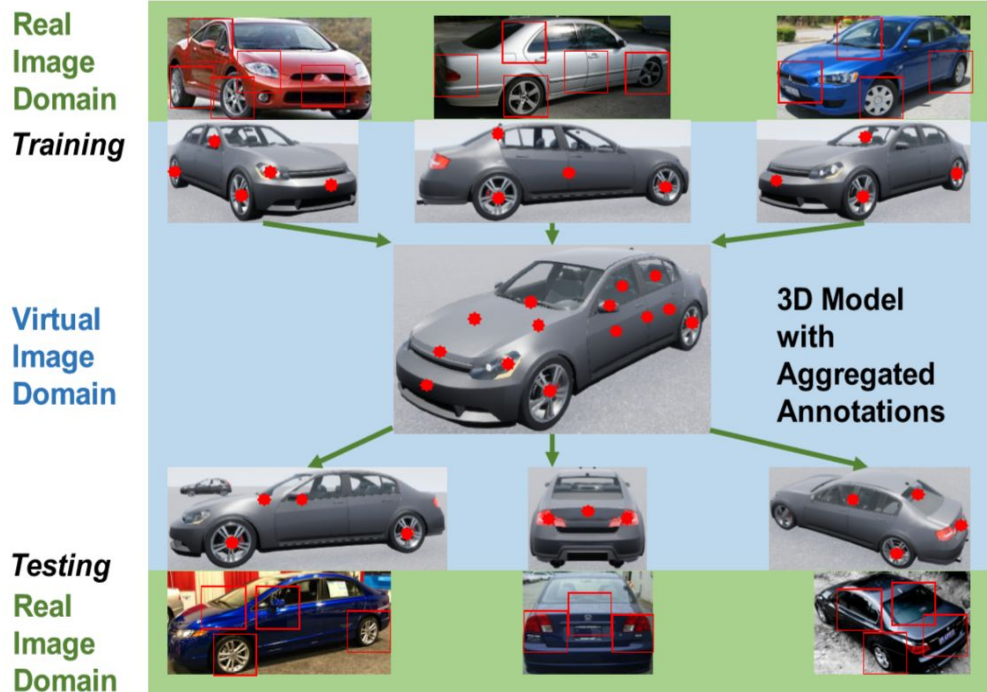


The pipeline starts with extracting deep features and applying a matching algorithm to find correspondence between images with similar viewpoints.

To deal with the problem of limited training data, we use a CAD model to provide 3D information. The semantic parts can be detected via matching cross different domains.



# Framework: Detection by Matching



## Pipeline:

### Training stage:

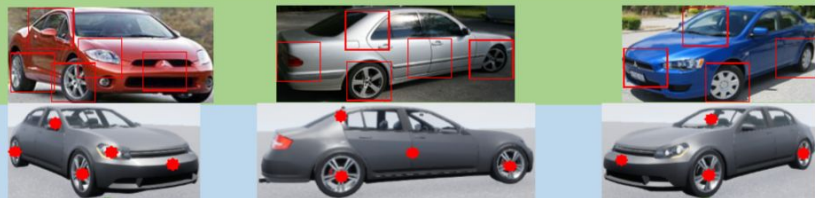
1. From a CAD model, we render synthetic images, which have same viewpoints as real training images.
2. Extract deep features from pool-4 layers of pretrained VGG-16 network for those images.
3. Fix the viewpoint and execute the matching algorithm between.
4. Aggregate the 2D parts to get their position on the 3D model.

### Testing stage:

1. Use viewpoint prediction method to estimate the pose of the testing image.
2. Render the synthetic images based on the estimated viewpoints.
3. Use the same matching algorithm to transfer the parts from synthetic model to real testing images.

# Two Key Components of this framework:

Real  
Image  
Domain  
*Training*



Virtual  
Image  
Domain



*Testing*  
Real  
Image  
Domain



1. **Matching algorithm**
2. Viewpoint prediction algorithm

# Semantic Matching and Semantic Consistency

**Goal:** matching  $\mathcal{M}$ : 
$$\mathcal{M}_n = \mathbf{M}(\mathbf{I}_n, \mathbf{I}'_n) = \left\{ (l_{n,w}, l'_{n,w}) \right\}_{w=1}^W$$

$\mathcal{M}_n$ : a set containing  $W$  pairs which  $l_{n,w}$ -th feature in  $\mathcal{V}_n$  and  $l'_{n,w}$ -th feature in  $\mathcal{V}'_n$  are matched, where  $\mathcal{V}_n$  and  $\mathcal{V}'_n$  represents sets of features extracted from real image and synthetic image.

$\mathbf{I}_n$  : real image;  $\mathbf{I}'_n$  : synthetic image;  $\mathbf{M}(\cdot)$ : matching function

**Objective:** minimize the overall loss function, which contains two parts:

- 1) **Semantic matching loss:** depicts the accuracy of the matching algorithm
- 2) **Semantic consistency loss:** ensures the parts' consistency seen from different viewpoint

# Semantic Matching

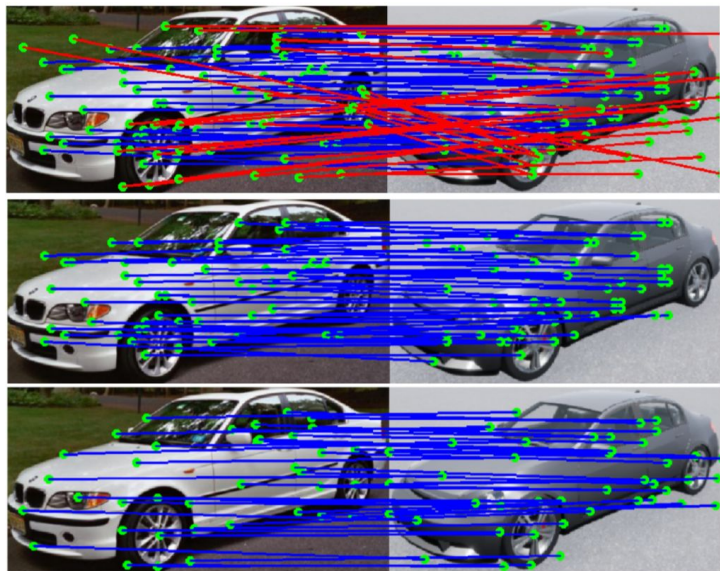
1) Semantic matching loss function:

$$\alpha(\mathcal{M}_n) = \lambda_1 \cdot \sum_{w=1}^W \left| \mathbf{v}_{n,l_n,w} - \mathbf{v}'_{n,l'_n,w} \right|^2 + \lambda_2 \cdot \sum_{1 \leq w_1 < w_2 \leq W} \left| \Delta \mathbf{u}_{l_n,w_1,l_n,w_2} - \Delta \mathbf{u}'_{l'_n,w_1,l'_n,w_2} \right|^2$$

$\mathbf{v}_{n,l_n,w}$  and  $\mathbf{v}'_{n,l'_n,w}$  represents the deep features,  $\Delta$  denotes the oriented distance between two feature coordinates

The first term measures the **similarity in appearance** of the matched patches, and the second term measures the **spatial consistency** among all matched patch pairs.

# Matching: Patch Similarity & Spatial Consistency



The matching process between real and synthesized images.

The first row depicts the result using the matched pool-4 features. The second represents the filtered matching pairs with spatial consistency. The third is the final matched key points after using pool-3 features to refine.

# Semantic Consistency

## 2) Semantic consistency loss

Based on  $\mathcal{M}_n$ , we can compute a coordinate transformation function,  $\mathbf{T}(\cdot)$ , which maps the bounding box of each semantic part of  $I_n$  to the corresponding region on  $I_n$

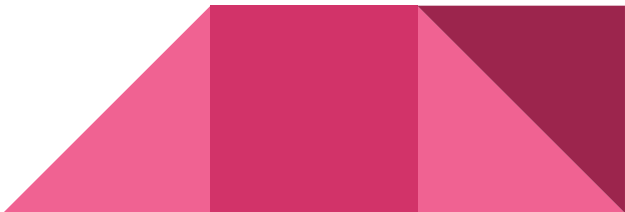
$$\mathbf{b}'_{n,m} = \mathbf{T}(\mathbf{b}_{n,m} \mid \mathcal{M}_n), \quad s'_{n,m} = s_{n,m}$$

where  $\mathbf{b}_{n,m}$  represents the parts on real image,  $\mathbf{b}'_{n,m}$  the transferred parts on synthetic image (in 2D image). After aggregating  $\mathbf{b}'_{n,m}$  into parts on

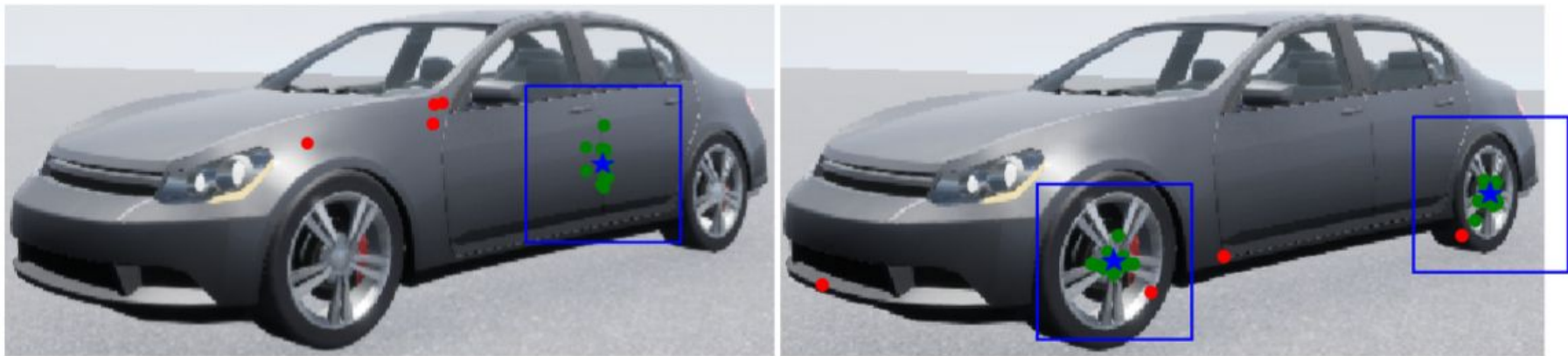
# Semantic Consistency

$$\beta(\mathcal{B}, \mathcal{C}) = \lambda_3 \cdot \sum_{n=1}^N \sum_{m=1}^{M_n^*} \min_{s_{n,m} = c_{m^*}} \text{dist}(\mathbf{b}'_{n,m}, \mathbf{c}_{n,m^*}) + \lambda_4 \cdot |\mathcal{C}|$$

where  $\mathcal{C}$  has  $M^*$  elements, each of which has a semantic part ID,  $c_{m^*}$ .  $\mathbf{c}_{n,m^*}$  is the projected coordinate of the  $m^*$ -th semantic part in the same viewpoint. The distance between  $\mathbf{b}'_{n,m}$  and  $\mathbf{c}_{n,m^*}$ ,  $\text{dist}(\cdot, \cdot)$  is measured by the Euclidean distance between the centers. The total distance of matching together with a regularization term  $|\mathcal{C}|$  contributes to the penalty  $\beta(\mathcal{B}, \mathcal{C})$



# Examples for Semantic Consistency



**Two examples of how semantic consistency improves the semantic part annotation on the 3D model.** The red dots represent incorrectly transferred semantic part annotations that get eliminated during our aggregation process using 3D semantic consistency. The green dots are the reasonable annotations that are used to get the final annotation for the targeted semantic part, which is represented by the blue dots at the center of blue bounding-boxes.

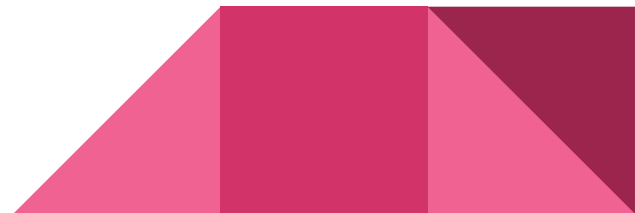


# Semantic Matching and Semantic Consistency

**Final objective: minimize the overall loss function,**

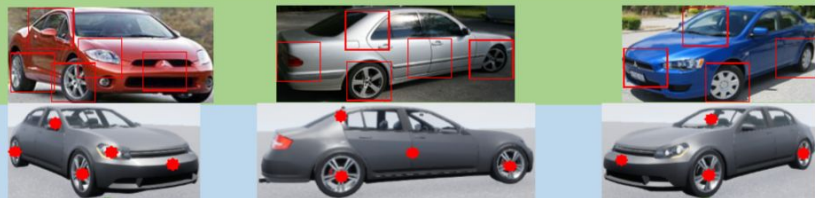
which contains the above two:

$$\mathcal{L}(\mathbb{A}, \mathcal{D}) = \sum_{n=1}^N \alpha(\mathcal{M}_n) + \beta(\mathcal{B}, \mathcal{C}).$$



# Two Key Components:

Real  
Image  
Domain  
*Training*



Virtual  
Image  
Domain



*Testing*  
Real  
Image  
Domain



1. Matching algorithm
2. **Viewpoint prediction algorithm**

# Similarity-based Viewpoint Prediction

**Energy function:**

$$E = \lambda \sum_{n=1}^N V_n + \frac{\mu}{N} \sum_{n=1}^N d_n + \frac{\gamma}{M} \sum_{m=1}^M \cos \langle \mathbf{v}_A^m, \mathbf{v}_B^m \rangle$$

- 1. Similarity of the parts:** the similarity of the corresponding parts in two images,  $V_n$ , is measured by the inner-product of their features, after being normalized.
- 2. Absolute position of these features:**  $d_n$  is the Euclidean distance between the 2D coordinates of two matched features.
- 3. Relative position of these parts:** calculating the cosine distance of the adjacent pairs of matched features.

# Results

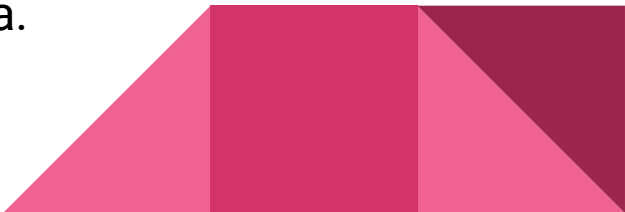
Approach	16 Training Samples				32 Training Samples				64 Training Samples			
	L0	L1	L2	L3	L0	L1	L2	L3	L0	L1	L2	L3
Faster R-CNN [42]	31.64	15.57	10.51	8.35	43.05	20.72	14.81	11.95	<b>55.43</b>	29.03	19.05	13.75
DeepVoting [58]	33.28	18.09	13.92	10.26	37.66	21.91	16.35	12.12	49.23	30.01	21.86	15.52
Ours	<b>42.25</b>	<b>27.44</b>	<b>23.87</b>	<b>14.03</b>	<b>44.06</b>	<b>28.29</b>	<b>23.76</b>	<b>14.58</b>	45.39	<b>33.93</b>	<b>25.24</b>	<b>16.75</b>

Table 1. Semantic part detection accuracy (by mAP, %) of different approaches under different number of training samples and occlusion situations. L0 through L3 indicate occlusion levels, with L0 being non-occlusion and L3 the heaviest occlusion.

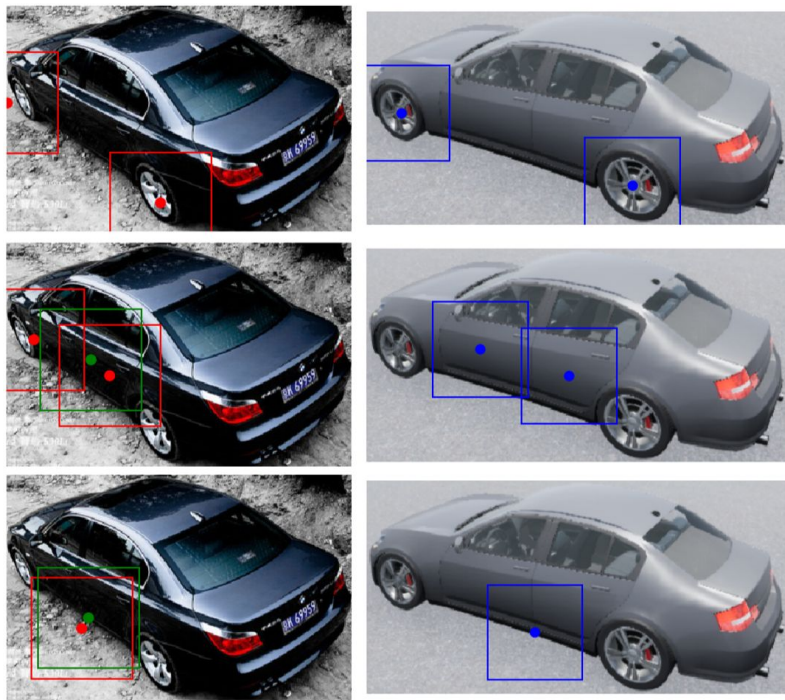
# Ability of Dealing with Unseen Viewpoints

Approach	# Training Samples		
	16	32	64
Faster R-CNN	16.02	21.80	19.91
DeepVoting	8.59	27.71	33.82
Ours	<b>45.32</b>	<b>47.03</b>	<b>45.88</b>

Table 2. Results (by mAP, %) of applying models trained with images under the viewpoint of an elevation angle  $0^\circ$  to unseen viewpoints (an elevation angle  $20^\circ$ ). Our model can generalize better to unseen viewpoints and the performance is almost constant regardless of the number of training data.



# Robustness on Unseen Viewpoints



Approach	# Training Samples		
	16	32	64
Faster R-CNN	16.02	21.80	19.91
DeepVoting	8.59	27.71	33.82
Ours	<b>45.32</b>	<b>47.03</b>	<b>45.88</b>

Results (by mAP, %) of applying models trained with images under the viewpoint of an elevation angle  $0^\circ$  to unseen viewpoints (an elevation angle  $20^\circ$ ).

# Transfer Across Different Prototypes


Approach	<i>sedan</i>	<i>SUV</i>	<i>mini- van</i>	<i>hatch- back</i>
Faster R-CNN	43.05	41.16	32.18	30.04
DeepVoting	37.66	37.18	30.67	31.62
Ours	<b>47.27</b>	<b>42.80</b>	<b>35.58</b>	<b>32.02</b>

Results of applying a model trained with sedan images to other prototypes (SUV, minivan, and hatchback). Our model can generalize better to unseen prototypes.



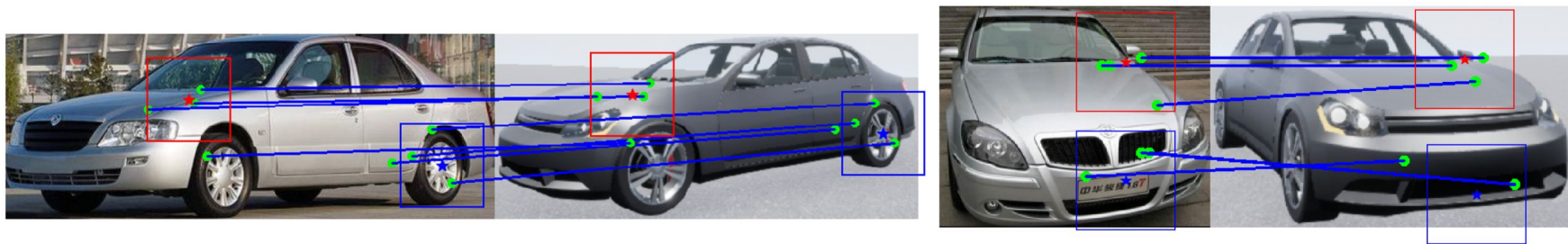
# Extending to Other Vehicle Types

The ability of training with few samples enables our model to be easily extended to these unrepresented classes. The accuracies of **our approach** for **bicycle** and **motorbike** are **67.16%** and **42.81%**, respectively. Due to the lack of training data for these categories, although we can manage to train an end-to-end model, the performance is inevitably worse than us. Faster R-CNN reports average accuracies of 44.02% and 29.79% for bicycle and motorbike, and the numbers for DeepVoting are 59.43% and 31.88%, respectively. This shows the practical advantage compared to prior work which only focused on car parsing.





# Interpreting Semantic Part Detection



The matching result is evidence for our approach to detect semantic parts. By visualizing the matching result, we can understand why the approach makes correct or incorrect predictions. Frames represent the detected semantic parts that are transferred from the synthetic image(right) to the testing image (left). Each semantic part is located based on nearby feature matching (shown in blue lines).

# Conclusions

In this paper, we present a novel framework for semantic part detection. The pipeline starts with extracting regional features and applying our robust matching algorithms to find correspondence between images with similar viewpoints. To deal with the problem of limited training data, an additional consistency loss term is added, which measures how semantic part annotations transfer across different viewpoints.

By introducing a 3D model as well as its viewpoints as hidden variables, we can optimize the loss function using an iterative algorithm. In the testing stage, we directly apply the same algorithms to match the semantic parts from the 3D model back to each 2D image and achieve high efficiency in the testing stage. Experiments are performed to detect semantic parts of car, bicycle and motorbike in the VSP dataset.

***Our approach works especially well with very few training images, on which other competitors often heavily over-fit the data and generalize badly.***



Thanks!