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

Yutong Bai[1]*, Qing Liu[1]*, Lingxi Xie[1,2]✉, Weichao Qiu[1], Yan Zheng[3], Alan Yuille[1]
[1]Johns Hopkins University [2] Huawei Noah's Ark Lab [3]University of Texas at Austin
{ytongbai, 198808xc, yan.zheng.mat, qiuwch, alan.l.yuille}@gmail.com qingliu@jhu.edu

Abstract

Detecting semantic parts of an object is a challenging task, particularly because it is hard to annotate semantic parts and construct large datasets. In this paper, we present an approach which can learn from a small annotated dataset containing a limited range of viewpoints and generalize to detect semantic parts for a much larger range of viewpoints. The approach is based on our matching algorithm, which is used for finding accurate spatial correspondence between two images and transplanting semantic parts annotated on one image to the other. Images in the training set are matched to synthetic images rendered from a 3D CAD model, following which a clustering algorithm is used to automatically annotate semantic parts of the CAD model. During the testing period, this CAD model can synthesize annotated images under every viewpoint. These synthesized images are matched to images in the testing set to detect semantic parts in novel viewpoints. Our algorithm is simple, intuitive, and contains very few parameters. Experiments show our method outperforms standard deep learning approaches and, in particular, performs much better on novel viewpoints. For facilitating the future research, code is available: https://github.com/ytongbai/SemanticPartDetection

1. Introduction

Detecting and parsing an object has been a long-lasting challenge in computer vision and has attracted a lot of research attention [8, 9]. Recently, with the development of deep networks, this research area has been dominated by an approach which starts by extracting several regional proposals and then determines if each of them belongs to a specific set of object classes [11, 42, 3, 31, 41]. The success of these approaches [8, 28] motivates researchers to address the more challenging task of understanding the objects at a finer level and, in particular, to parse it into semantic parts, which was defined to be those components of an object with semantic meaning and can be verbally described [58]. A particular challenge lies in that annotating semantic parts is much more difficult and time-consuming than annotating objects, which makes it harder to directly apply deep networks to this task.

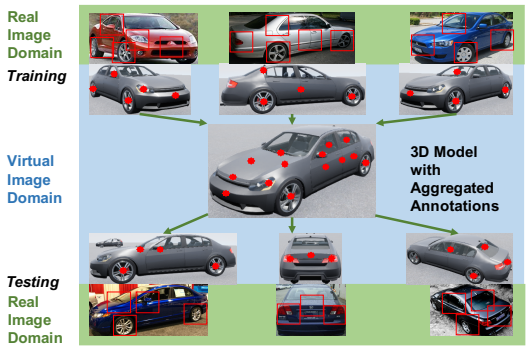

Figure 1. The flowchart of our approach (best viewed in color). The key module is the matching algorithm which finds spatial correspondence between real and synthesized images in similar viewpoints. This enables us to match the training data to a 3D CAD model and thereby annotate it. The CAD model can then be used to detect the semantic parts on images in the testing set, by reusing the matching algorithm.

In this paper, we address the problem of semantic part detection in the scenario that only a small amount of training data are available and the object is seen from a limited range of viewpoints. The overall framework is illustrated in Figure 1. Our strategy is to design a matching algorithm which finds correspondences between images of the same object seen from roughly the same viewpoint. This can be used to match the training real images to the rendered images of a 3D CAD model, enabling us to annotate the semantic parts of the 3D model automatically. The same matching algorithm can then be used in the testing time, which transplants the annotated semantic parts on the CAD model to the testing images, even though their viewpoints may not have appeared in the training set.

In this pipeline, the key component is the matching algo-

*The first two authors contributed equally to this work.

rithm. For simplicity, we only assume it to work on two images, one real and one synthesized, with similar viewpoints. The viewpoint of the real image is provided by ground-truth. Meanwhile, the synthesized image can be rendered using the viewpoint of the real image. On these two images, regional features are extracted using a pre-trained network, and matched using an optimization algorithm with geometric constraints considered. Our framework has potential to enable more accurate matching algorithms to be used in the future to improve the performance of object parsing.

The major contribution of this work is to provide a simple and intuitive algorithm for semantic part detection which works using little training data and can generalize to novel viewpoints. It is an illustration of how virtual data can be used to reduce the need for time-consuming semantic part annotation. Experiments are performed on the VehicleSemanticPart (VSP) dataset [52], which is currently the largest corpus for semantic part detection. Our approach achieves better performance than standard end-to-end methods such as Faster R-CNN [42] and DeepVoting [58] in *car*, *bicycle* and *motorbike*. The advantages become even bigger when the amount of training data is small.

The remainder of this paper is organized as follows. Section 2 briefly reviews the prior literature, and Section 3 presents our framework. After experiments are shown in Section 4, we conclude this work in Section 5.

2. Related Work

In the past years, deep learning [26] has advanced the research and applications of computer vision to a higher level. With the availability of large-scale image datasets [5] as well as powerful computational devices, researchers designed very deep neural networks [25, 44, 46] to accomplish complicated vision tasks. The fundamental idea of deep learning is to organize neurons (the basic units that perform specified mathematical functions) in a hierarchical manner, and tune the parameters by fitting a dataset. Based on some learning algorithms to alleviate numerical stability issues [35, 45, 22], researchers developed deep learning in two major directions, namely, increasing the depth of the network towards higher recognition accuracy [16, 20, 19], and transferring the pre-trained models to various tasks, including feature extraction [6, 43], object detection [11, 10, 42], semantic segmentation [32, 2], pose estimation [36], boundary detection [55], *etc*.

For object detection, the most popular pipeline, in the context of deep learning, involves first extracting a number of bounding-boxes named regional proposals [1, 50, 42], and then determining if each of them belongs to the target class [11, 10, 42, 3, 31, 41]. To improve spatial accuracy, the techniques of bounding-box regression [23] and non-maximum suppression [17] were widely used for post-processing. Boosted by high-quality visual features and end-to-end optimization, this framework significantly outperforms the conventional deformable part-based model [9] which were trained on top of handcrafted features [4]. Despite the success of this framework, it still suffers from weak explainability, as both object proposal extraction and classification modules were black-boxes, and thus easily confused by occlusion [58] and adversarial attacks [54]. There were also research efforts of using mid-level or high-level contextual cues to detect objects [52] or semantic parts [58]. These methods, while being limited on rigid objects such as vehicles, often benefit from better transferability and work reasonably well on partially occluded data [58].

Figure 2. Two examples of annotated semantic parts in the class *car*. For better visualization, we only show part of the annotations.

Another way of visual recognition is to find correspondence between features or images so that annotations from one (training) image can be transplanted to another (testing) image [15, 24, 37, 48, 49, 30]. This topic was noticed in the early age of vision [38] and later built upon handcrafted features [34, 18, 56]. There were efforts in introducing semantic information into matching [29], and also improving the robustness against noise [33]. Recently, deep learning has brought a significant boost to these problems by improving both features [43, 59] and matching algorithms [7, 14, 60, 21, 49], while a critical part of these frameworks still lies in end-to-end optimizing deep networks.

Training a vision system requires a large amount of data. To alleviate this issue, researchers sought for help from the virtual world, mainly because annotating virtual data is often much easier and cheaper [40]. Another solution is to perform unsupervised or weakly-supervised training with consistency that naturally exists [60, 13, 61]. This paper investigates both of these possibilities.

3. Our Approach

3.1. Problem: Semantic Part Detection

The goal of this work is to detect *semantic parts* on an image. We use $\mathcal{P}$ to denote the image-independent set of semantic parts, each element in which indicates a verbally describable component of an object [52]. For example, there are tens of semantic parts in the class *car*, including *wheel*, *headlight*, *license plate*, *etc*. Two *car* examples with semantic parts annotated are shown in Figure 2.

Let the training set $\mathcal{D}$ contain N images, and each image, $\mathbf{I}_n$, has a spatial resolution of $W_n \times H_n$. A set $\mathcal{S}_n^\star$ of

$M_n^\star$ semantic parts are annotated for each image, and each semantic part appears as a bounding box $\mathbf{b}_{n,m}^\star$ and a class label $s_{n,m}^\star \in \{1, 2, \ldots, |\mathcal{P}|\}$, where m is the index.

The goal of this work is to detect semantic parts in a testing image, in particular when the number of training images is very small, *e.g.*, N is smaller than 50.

3.2. Framework: Detection by Matching

We desire a function $\mathcal{S} = \mathbf{f}(\mathbf{I}; \boldsymbol{\theta})$ which receives an image $\mathbf{I}$ and outputs the corresponding semantic part set $\mathcal{S}$. In the context of deep learning, researchers designed end-to-end models [42] which start with an image, pass the signal throughout a series of layers, and output the prediction in an encoded form. With ground-truth annotation $\mathcal{S}_n^\star$, a loss function $\mathcal{L}_n$ is computed between $\mathcal{S}_n = \mathbf{f}(\mathbf{I}_n; \boldsymbol{\theta})$ and $\mathcal{S}_n^\star$, and the gradient of $\mathcal{L}_n$ with respect to $\boldsymbol{\theta}$ is computed in order to update $\boldsymbol{\theta}$ accordingly. DeepVoting [58] went a step further by explicitly formulating spatial relationship at high-level inference layers so that the model can deal with partial occlusion. Despite their success, these approaches often require a large number of annotations to avoid overfitting (see Section 4.2.1), yet their ability to generalize to unseen viewpoints is relatively weak (see Section 4.2.3).

This paper works from another perspective which, instead of directly optimizing $\mathbf{f}(\cdot)$, adopts an indirect approach to find semantic correspondence between two images with similar viewpoints. This is to say, if a training image $\mathbf{I}_n$ is annotated with a set of semantic parts, $\mathcal{S}_n^\star$, and we know that $\mathbf{I}_n$ is densely matched to a testing image $\mathbf{I}_\circ$, then we can transplant $\mathcal{S}_n^\star$ to $\mathcal{S}_\circ$ by projecting each element of $\mathcal{S}_n^\star$ into the corresponding element of $\mathcal{S}_\circ$ by applying a spatially-constrained mapping function.

This method has two key factors. First, it works in the scenario of few (*e.g.*, tens of) training images. Second, it assumes that for every testing image, there exists a training image which has a very similar viewpoint (because the accuracy of semantic matching is not guaranteed in major viewpoint change). To satisfy these two conditions that seem to contradict, we make the **main contribution** of this work, that introduces auxiliary cues from a 3D model $\mathbb{A}$. $\mathbb{A}$ is a virtual model created by computer graphics. Provided a viewpoint $\mathbf{p}$, a rendering function can generate an image $\mathbf{I}'$, possibly with semantic parts if they are present in $\mathbb{A}$.

Now we describe the overall flowchart as follows. In the training stage, we estimate the parameters of $\mathbb{A}$ (*e.g.*, 3D vertices, semantic parts, *etc.*) from the training set, for which we render $\mathbb{A}$ in the viewpoint of each training image and perform semantic matching. In the testing stage, we render $\mathbb{A}$ using the predicted viewpoint of the testing image, and make use of semantic matching to transplant the semantic parts to the testing image. Viewpoint prediction will be described in Section 3.5 in details.

In what follows, we formulate our approach mathematically and solve it using an efficient optimization algorithm.

3.3. Semantic Matching and Viewpoint Consistency

This subsection describes three key modules (a rendering function, a semantic matching function, and a coordinate transformation function) and two loss terms (for geometric consistency and semantic consistency, respectively) that compose of the overall objective function.

We start with defining two key functions. First, a **rendering function**, $\mathbf{R}(\cdot)$, takes the 3D model $\mathbb{A}$, refers to the target viewpoint $\mathbf{p}_n$, and outputs a rendered image:

$$\mathbf{I}'_n = \mathbf{R}(\mathbb{A} \mid \mathbf{p}_n). \tag{1}$$

Throughout the remaining part, a prime in superscript indicates elements in a generated image.

Next, we consider a **semantic matching function** between a rendered image $\mathbf{I}'_n$ and a real image $\mathbf{I}_n$ of the same viewpoint[1]. We assume that both $\mathbf{I}_n$ and $\mathbf{I}'_n$ are represented by a set of regional features, each of which describes the appearance of a patch. In the context of deep learning, this is achieved by extracting mid-level neural responses from a pre-trained deep network [52, 59]. Although it is possible to fine-tune the network with an alternative head for object detection [58], we do not take this option for simplicity. Let an image $\mathbf{I}_n$ have a set, $\mathcal{V}_n$, consisting of L_n regional features, the l-th of which is denoted by $\mathbf{v}_{n,l}$. We assume that all these feature vectors have a fixed length, *e.g.*, all of them are 512-dimensional vectors corresponding to specified positions at the *pool-4* layer of VGGNet [44, 52]. Each $\mathbf{v}_{n,l}$ is also associated with a 2D coordinate $\mathbf{u}_{n,l}$. Based on these features, we build a matching function, $\mathbf{M}(\cdot)$, which takes the form of:

$$\mathcal{M}_n = \mathbf{M}(\mathbf{I}_n, \mathbf{I}'_n) = \left\{ \left(l_{n,w}, l'_{n,w} \right) \right\}_{w=1}^{W}, \tag{2}$$

which indicates that the $l_{n,w}$-th feature in $\mathcal{V}_n$ and the $l'_{n,w}$-th feature in $\mathcal{V}'_n$ are matched. Based on the assumption that $\mathbf{I}_n$ and $\mathbf{I}'_n$ have similar viewpoints[2], we make use of both unary and binary relationship terms to evaluate the quality of $\mathcal{M}_n$ in terms of appearance and spatial consistency, and thus define a **semantic matching loss** of $\mathcal{M}_n$, $\alpha(\mathcal{M}_n)$:

$$\alpha(\mathcal{M}_n) = \lambda_1 \cdot \sum_{w=1}^{W} \left| \mathbf{v}_{n,l_{n,w}} - \mathbf{v}'_{n,l'_{n,w}} \right|^2 + \lambda_2 \cdot \sum_{1 \leqslant w_1 < w_2 \leqslant W} \left| \Delta\mathbf{u}_{l_{n,w_1}, l_{n,w_2}} - \Delta\mathbf{u}'_{n,l'_{n,w_1}, l'_{n,w_2}} \right|^2, \tag{3}$$

[1] In the testing process, $\mathbf{I}_n$ and $\mathbf{I}'_n$ are replaced by $\mathbf{I}_\circ$ and $\mathbf{I}'_\circ$ (the 3D model rendered in the estimated viewpoint of $\mathbf{I}_\circ$), respectively.

[2] In the testing process, we estimate the viewpoint of $\mathbf{I}_\circ$ and render a new image $\mathbf{I}'_\circ$ correspondingly. The estimator may bring in inaccuracy so that the viewpoints of these two images can be slightly different (*e.g.*, [47] reported a medium viewpoint prediction error of less than 10° on the *car* subclass), but most often, this does not harm the matching algorithm.

where Δ denotes the oriented distance between two feature coordinates, *i.e.*, $\Delta\mathbf{u}_{l_{n,w_1},l_{n,w_2}} = \mathbf{u}_{l_{n,w_1}} - \mathbf{u}_{l_{n,w_2}}$ and $\Delta\mathbf{u}'_{n,l_{n,w_1},l_{n,w_2}} = \mathbf{u}'_{n,l_{n,w_1}} - \mathbf{u}'_{n,l_{n,w_2}}$. Thus, the first term on the right-hand side measures the similarity in appearance of the matched patches, and the second term measures the spatial consistency among all matched patch pairs.

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 $\mathbf{I}_n$ to the corresponding region on $\mathbf{I}'_n$:

$$\mathbf{b}'_{n,m} = \mathbf{T}(\mathbf{b}_{n,m} \mid \mathcal{M}_n), \quad s'_{n,m} = s_{n,m}. \tag{4}$$

The annotations collected from all training images form a set, $\mathcal{B}$. Recall that the final goal is to infer the semantic parts of $\mathbb{A}$, which form a subset of vertices denoted by $\mathcal{C}$. To this end, we define the second loss term, a **semantic consistency loss**, denoted by $\beta(\mathcal{B} \mid \mathcal{C})$, which measures the inconsistent pairs of semantic parts on $\mathcal{B}$ and $\mathcal{C}$:

$$\beta(\mathcal{B},\mathcal{C}) = \lambda_3 \cdot \sum_{n=1}^{N}\sum_{m=1}^{M^\star_n} \min_{s_{n,m}=c_{m^*}} \mathrm{dist}\big(\mathbf{b}'_{n,m}, \mathbf{c}_{n,m^*}\big) + \lambda_4 \cdot |\mathcal{C}|, \tag{5}$$

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 viewpoint of $\mathbf{p}_n$. The distance between $\mathbf{b}'_{n,m}$ and $\mathbf{c}_{n,m^*}$, $\mathrm{dist}(\cdot,\cdot)$, 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.

The final objective is to minimize the overall loss function which sums up Eqns (3) and (5) together:

$$\mathcal{L}(\mathbb{A},\mathcal{D}) = \sum_{n=1}^{N}\alpha(\mathcal{M}_n) + \beta(\mathcal{B},\mathcal{C}). \tag{6}$$

3.4. Training and Testing

Both training and testing involve optimizing Eqn (6).

Training determines the semantic parts on $\mathbb{A}$ based on those annotated in $\mathcal{D}$, while testing applies the transplants the learned semantic parts from $\mathbb{A}$ to an unseen image. For simplicity, we assume that all the vertices of $\mathbb{A}$ are fixed, which means that a pre-defined 3D model is given. With more powerful 3D matching techniques in the future, this assumption can be relaxed, allowing the 3D model itself to be inferred by the dataset. Based on this, the overall optimization process is partitioned into four steps. During optimization, we replace parameters λ_1 through λ_4 with empirically set thresholds, as described below. Our algorithm is not sensitive to these parameters.

The **first** step involves Eqn (1) which renders all virtual images $\mathbf{I}'_n$ according to the ground-truth viewpoint $\mathbf{p}_n$ (of $\mathbf{I}_n$). We purchased a high-resolution model from the Unreal Engine Marketplace, and use UnrealCV [40], a public library based on Unreal Engine, to synthesize all images. The rendering function $\mathbf{R}(\cdot)$ is implemented with standard rasterization in a game engine. We place the 3D model in regular background with *road* and *sky*, and use two directional light sources to reduce shadow in the rendered images (this improves image matching performance). Some typical examples are displayed in Figure 1.

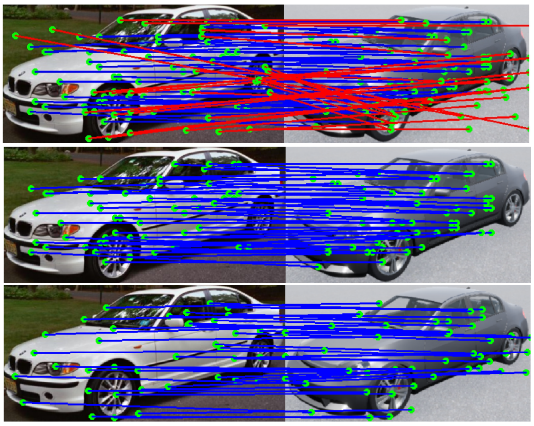

Figure 3. The matching process between real and synthesized images. The first row depicts the result using the matched regional features. The second represents the filtered matching pairs with geometric constraints. The third is the final matched key points after using *pool-3* features to refine.

In the **second** step, we match each training image $\mathbf{I}_n$ to the corresponding synthesized image $\mathbf{I}'_n$. This is to compute a function $\mathcal{M}_n$ that minimizes the semantic matching loss $\alpha(\mathcal{M}_n)$, as defined in Eqns (2) and (3). We use a fixed and approximate algorithm for this purpose. Given an image, either real or synthesized, we extract features by rescaling each image so that the short axis of the object is 224-pixel long [52], followed by feeding it into a pre-trained 16-layer VGGNet [44] and extracting all 512-dimensional feature vectors at the *pool-4* layer. All feature vectors are ℓ_2-normalized. There are $L_n \times L'_n$ feature pairs between $\mathbf{I}_n$ and $\mathbf{I}'_n$. For each of them, (l, l'), we compute the ℓ_2 distance between $\mathbf{v}_{n,l}$ and $\mathbf{v}'_{n,l'}$ and use a threshold ξ to filter them. On the survived features, we further enumerate all quadruples (l_1, l_2, l'_1, l'_2) with l_1 matched to l'_1 and l_2 matched to l'_2, compute $\Delta\mathbf{u}_{l_1,l_2}$ and $\Delta\mathbf{u}'_{l'_1,l'_2}$, and again filter them with a threshold ζ. Finally, we apply the Bron-Kerbosch algorithm to find the max-clique on both images that are matched with each other. An example of the matching algorithm is shown in Figure 3. In practice, ξ and ζ are determined empirically, and the matching performance is not sensitive to these parameters.

In the **third** step, we compute the transformation function $\mathbf{T}(\cdot)$ defined in Eqn (4). This is to determine how each coordinate in one image is mapped to that in the other. We use the nearby matched features to estimate this function. For each semantic part, a weighted average of

its neighboring features' relative translation is applied to the annotation, where the weights are proportional to the inverse of the 2D Euclidean distances between the semantic part and the features in the source image. The basis of our approach is the feature vectors extracted from a pre-trained deep network. However, these features, being computed at a mid-level layer, often suffer a lower resolution in the original image plane. For example, the *pool-4* features of VGGNet [44] used in this paper have a spatial stride of 16, which leads to inaccuracy in feature coordinates and, consequently, transformed locations of semantic parts. To improve matching accuracy, we extract two levels of regional features. The idea is to first use higher-level (*e.g.*, *pool-4*) features for semantic matching, and then fine-tune the matching using lower-level (*e.g.*, *pool-3*) features which have a smaller spatial stride for better alignment.

The **fourth** and final step varies from training and testing. In the *training* stage, we collect semantic parts from annotated real images, transform them to the virtual model, and determine the set $\mathcal{C}$ by minimizing Eqn (5). This is approximately optimized by a standard K-Means clustering on all transformed semantic parts, *i.e.*, the set of $\mathcal{B}$. After clustering is done, we enumerate the number of semantic parts, *i.e.*, $|\mathcal{C}|$, and choose the maximum clustering centers accordingly. This is to say, the final position of each semantic part is averaged over all transformed semantic parts that are clustered to the same center. In practice, we take the center of the bounding box annotations and find the closest viewable vertex on the 3D model. In the **testing** stage, the reversed operation is performed, transforming the learned semantic parts back to each real image individually. This can introduce 3D prior to overcome the issue of occlusion (see experiments).

3.5. Similarity-based Viewpoint Prediction

From diagnostic experiments, we can see that the accuracy of semantic part detection relies on that of viewpoint estimation. Except from using either ground-truth or an off-the-shelf method such as Click-Here-CNN [47], here we present an approach of using the quality of semantic matching itself to measure whether two images are similar enough, and thus provide an alternative to viewpoint estimation.

We start with the maximum clique computed on the matched features between two images. For the details of feature matching, graph construction and maximum clique computation, please refer to Section 3.4.

After the maximum clique algorithm, we obtain a shared sub-graph which is composed of the matched parts in two images. Let N and M be the number of vertices and edges in the graph defined by the maximum clique. Based on this graph, we design an energy function to describe the viewpoint similarity:

$$E = \lambda \sum_{n=1}^{N} V_n + \frac{\mu}{N} \sum_{n=1}^{N} d_n + \frac{\gamma}{M} \sum_{m=1}^{M} \cos \langle \mathbf{v}_{\mathbf{A}}^m, \mathbf{v}_{\mathbf{B}}^m \rangle. \quad (7)$$

This is to say, the quality of matching is determined by the similarity of matched parts, their absolute position and their relative position. The **first** term represents 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. Note that by summing up all feature pairs, we are actually taking the number of the matched parts into consideration, *i.e.*, the more matched feature pairs, the higher similarity. The **second** term restricts the absolute position of these features: d_n is the Euclidean distance between the 2D coordinates of two matched features (of two objects) in the cropped images. This is based on an intuitive assumption that, for example, if two cars have the same viewpoint, their corresponding key points should be located in approximately the same position in the image. The **third** term confines the relative position of these parts by calculating the cosine distance $\cos \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ of the adjacent pairs of matched features. For a pair of features, $\mathbf{p}$ and $\mathbf{q}$, we calculate the cosine distance between $\mathbf{v}_{\mathbf{A}}$ and $\mathbf{v}_{\mathbf{B}}$, which point from $\mathbf{p}$ to $\mathbf{q}$ in $\mathbf{A}$ and $\mathbf{B}$. λ, μ and γ are parameters to balance different terms.

Based on this function, the matching algorithm works as follows. Given a real image in the testing stage, we first enumerate 8 synthetic images with the azimuth angles of $\{0, 45, 90, \dots 315\}$. After finding the most similar case, we narrow down the searching range to a region of 90 degrees, and enumerate at every 10 degrees to find the best match. 10 degrees is the amount that preserves most feature matches.

3.6. Discussions

Compared to prior methods on object detection [42] or parsing [58], our approach enjoys a higher explainability as shown in experiments (see Figure 6). Here, we inherit the argument that low-level or mid-level features can be learned by deep networks as they often lead to better local [57] or regional [43] descriptions, but the high-level inference stage should be semantically meaningful so that we can manipulate either expertise knowledge or training data for improving recognition performance or transferring the pipeline to other tasks. Moreover, our approach requires much fewer training data to be optimized, and applies well in novel viewpoints which are not seen in training data.

The training process of our approach can be largely simplified if we fix $\mathbb{S}$, *e.g.*, manually labeling semantic parts on each 3D model $\mathbb{I}$. However, the amount of human labor required increases as the complexity of annotation as well as the number of 3D models. Our approach serves as a good balance – the annotation on each 2D image can be

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	**55.43**	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	**42.25**	**27.44**	**23.87**	**14.03**	**44.06**	**28.29**	**23.76**	**14.58**	45.39	**33.93**	**25.24**	**16.75**

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.

transferred to different 3D models. In addition, 2D images are often annotated by different users, which provide complementary information by crowd-sourcing [5]. Therefore, learning 3D models from the ensemble of 2D annotations is a safer option.

4. Experiments

In experiments, we first compare our performance on detecting semantic parts of *sedan* with Faster-RCNN and DeepVoting. In this comparison, we studied the impact of the amount of data and occlusion level. Then, we explore the potential of our approach in various challenges, including its sensitivity to viewpoint prediction error, ability of working on unseen viewpoints, and ability of being applied to other prototypes. Finally, we apply our model to two more vehicle types, namely *bicycle* and *motorbike*, which have fewer available training data.

4.1. Settings and Baselines

We perform experiments on the VehicleSemanticPart (VSP) dataset [52]. We start with *sedan* the most popular prototype of *car*, then later shows only using a *sedan* CAD model, our approach can generalize to other prototypes (*e.g.*, *minivan*) without the need of a CAD model for each prototype. There are 395 training and 409 testing images, all of which come from the Pascal3D+ dataset [53], and the authors of [52] manually labeled 39 semantic parts, covering a large fraction of the surface of each *car* (examples in Figure 2). There are 9 semantic parts related to *wheel*, 1 at the center and other 8 around the rim. We only consider the center one as the others are less consistent in the annotation.

We use the ground-truth azimuth angle to categorize training images into 8 bins, centered at $0°, 45°, \ldots, 315°$, respectively. We randomly sample $N' \in \{2, 4, 8\}$ images in each bin, leading to three training sets with 16, 32 and 64 images, which are much smaller than the standard training set (395 images). Following the same setting of [58], we evaluate our approach on both clean object and occluded objects, where different levels of occlusion are tested.

The competitors of our approach include DeepVoting [58], a recent approach towards explainable semantic part detection, as well as Faster R-CNN [42] (also used in [58]), an end-to-end object detection algorithm. Other approaches (*e.g.*, [52] and [51]) are not listed as they have been verified weaker than DeepVoting.

4.2. Quantitative Results

We first investigate the scenario that the ground-truth viewpoint (quantized into the 8 bins) is provided. Though obtaining extra information, we point out that annotating the viewpoint of an object often requires less than 3 seconds – in comparison, labeling all semantic parts costs more than one minute. In later experiments, we also provide parsing results with predicted viewpoints and study the sensitivity of our approach to viewpoint accuracy.

Results are summarized in Table 1. One can observe that our approach outperforms both DeepVoting and Faster R-CNN, especially in the scenarios of (i) fewer training data and (ii) heavier occlusion.

Since our approach is viewpoint-aware, we also equip baseline methods with ground-truth viewpoint information. More specifically, we train 8 individual models (for both Faster R-CNN and DeepVoting), each of which takes charge of objects within one bin. On the clean testing images (occlusion level L0), the accuracy of Faster R-CNN is almost unchanged, and that of DeepVoting is improved by 1%–6%, but still much lower than our results.

4.2.1 Impact of the Amount of Training Data

One major advantage of our method is the ability to learn from just a few training samples by preserving the 3D geometry consistency for images from different viewpoints. As shown in Table 1, when using only 16 training images, which means no more than 6 training samples for most semantic parts, our method still gives reasonable predictions and outperforms other baseline method by a large margin. By increasing the training sample number, our method also benefits from learning more accurate annotations on the 3D model, resulting to higher mAP in the 2D detection task. By contrast, both Faster R-CNN and DeepVoting fail easily given small number of training.

4.2.2 Ability of Dealing with Occlusion

To evaluate the robustness of our method to occlusion, we apply the models learned from the occlusion-free dataset to images with different levels of occlusion. Different from DeepVoting [58] which learns the spatial relationship between semantic parts and their characteristic deep features in occlusion-free images, our method directly models the spatial relationship of parts by projecting them to the 3D

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

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

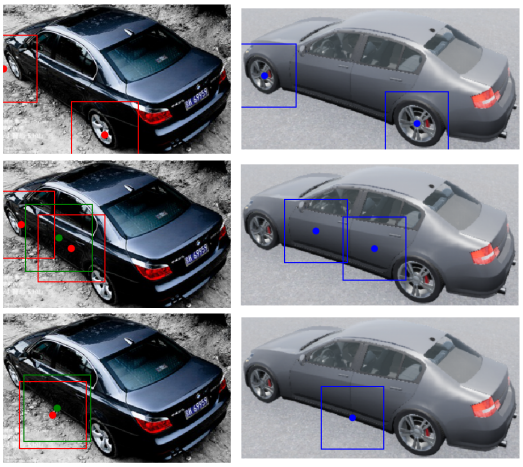

Figure 4. Our approach also works on unseen viewpoints (training on an elevation angle of 0° and testing on 20°). It is worth noting that the *wheel* has been completely self-occluded in the elevation angle of 0°, but our model can still detect it. Left: a testing image with transferred annotations (red) and ground-truth (green). The ground-truth for *wheels* is missing, which is an annotation error. Right: the synthetic image in the same viewpoint with learned semantic parts.

space, and then matches them back to the occluded objects. On light occlusions, our method consistently beats the baselines. In the scenarios of heavier occlusion, due to the deficiency of accurately matched features, the performance of our method deteriorates. As expected, Faster R-CNN lacks the ability of dealing with occlusion and its performance drops quickly, while DeepVoting is less affected.

It is interesting to see that the robustness to fewer training data and occlusion is negatively related to the number of extra parameters. For example, DeepVoting has less than 10% parameters compared to Faster R-CNN, and our approach, being a stage-wise one, only requires some hyper-parameters to be set. This largely alleviates the risk of over-fitting (to small datasets) and the difficulty of adapting a model trained on non-occluded data to occluded data.

4.2.3 Robustness on Unseen Viewpoints

To show that our approach has the ability of working on unseen viewpoints, we train the models using *sedan* images with various azimuth angle and 0° elevation angle, then test them on *sedan* with elevation angle equal or larger than 10°. The results are shown in Table 2. Our method maintains roughly the same mAP as tested on all viewpoints, while Faster R-CNN and DeepVoting deteriorate heavily. In Figure 4, we show predictions made by our method on one sample with unseen viewpoint (elevation angle equals 20°). The predicted locations are very close to the annotated ground truth and may help to fix the annotation error in the dataset.

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

Table 3. Results of applying a model trained with *sedan* images to other prototypes (*SUV*, *minivan*, and *hatchback*). Our model can generalize better to unseen prototypes.

4.2.4 Transfer Across Different Prototypes

In order to evaluate how sensitive our method is to the *car* prototype used during training, we transfer the model trained with *sedan* images to other prototypes of *cars*, including *minivan*, *SUV* and *hatchback*. Results are summarized in Table 3. As expected, our method generalizes well to prototypes with similar appearance (e.g., *SUV*). For *minivan* and *hatch-back*, due to the variation of the 3D structures and semantic part definitions, the performance drops more. Similar results are observed for Faster R-CNN and DeepVoting, and DeepVoting seems slightly more robust to the prototypes.

4.2.5 Extending to Other Vehicle Types

For vehicle, most of computer vision research focused on *car*, which has a lot of annotated images. *Bicycle* ,and *motorbike*, while being equally important objects on the road, attracted much fewer attention, partially due to the lack of data. 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 [39, 12, 27] which only focused on *car* parsing.

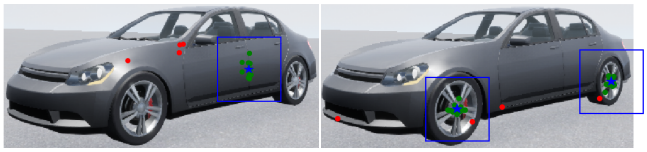

Figure 5. Two examples of how viewpoint consistency improves the semantic part annotation on the 3D model. The red dots represent incorrectly transferred semantic part annotations that get eliminated during our aggregation process using 3D geometry constraints. The green dots are the reasonable annotations 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.

4.3. Qualitative Studies

4.3.1 Viewpoint Consistency in Training

In Figure 5, we show examples of how viewpoint consistency improves the stability of the training stage. Although we have applied 2-D geometry coherence as one of the criteria during matching individual training samples to their viewpoint-paired synthetic images, it is possible to get wrong matched features at inaccurate positions. Therefore, the semantic part annotations transferred from an individual training image could be far off the ground truth area (e.g., outliers shown by the red circles). With viewpoint consistency, the incorrect annotations are eliminated during aggregation, and our method stably outputs the right position for the targeted semantic parts (e.g., final annotation shown by blue stars) based on the reasonably transferred annotations (e.g., inliers shown by green circles).

4.3.2 Interpreting Semantic Part Detection

Next, we provide some qualitative results to demonstrate the explainability of our approach. In Figure 6, we show examples of how we locate the semantic parts in two image pairs. Each pair includes one testing image and its viewpoint-matched synthetic image.The star represents the location of the semantic parts in each image (learned from training in the synthetic images and got transferred to in the testing images), and the color represents their identity. The transformation is learned using nearby matched features, which are shown by green circles (matched features are linked by red lines). For better visualization, we only display the nearest three features for each semantic part in the figure. This explains what features are used to transfer the annotation from synthetic images to testing images, and helps us understand what is going on during the inference process.

5. 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 (*e.g.*, tens of) training images, on which other competitors [42, 58] often heavily over-fit the data and generalize badly.

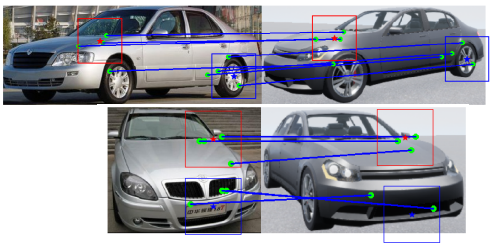

Figure 6. Interpret semantic part detection. The matching result is evidence for our approach to detect semantic parts. By visualizing the matching result, we can understand why the approach makes correct or incorrect predictions. Frames represent the detected semantic parts that are transferred from the synthetic image (right) to the testing image (left). Each semantic part is located based on nearby feature matching (shown in blue lines).

Our approach provides an alternative solution to object parsing, which has three major advantages: (i) it can be trained on a limited amount of data and generalized to unseen viewpoints; (ii) it can be trained on a subset of viewpoints and then transferred to novel ones; and (iii) it can be assisted by virtual data in both training and testing. However, it still suffers from the difficulty of designing parameters, which is the common weakness of stepwise methods compared to the end-to-end learning methods.

Researchers believe that 3D is the future direction of computer vision. In the intending research, we will try to learn one or more 3D models directly from 2D data, or allow the 3D model to adjust slightly to fit 2D data. More importantly, it is an intriguing yet challenging topic to generalized this idea to non-rigid objects, which will largely extend its area of application.

Acknowledgement

This research was supported by ONR grant N00014-18-1-2119 and iARPA DIVA with grant D17PC00342. We thank Huiyu Wang, Yingwei Li and Wei Shen for instructive discussions.

References

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012. 2

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 2

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 1, 2

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. 2005. 2

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 6

[6] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014. 2

[7] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 2

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 1

[9] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009. 1, 2

[10] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1, 2

[12] Daniel Glasner, Meirav Galun, Sharon Alpert, Ronen Basri, and Gregory Shakhnarovich. aware object detection and continuous pose estimation. *Image and Vision Computing*, 30(12):923–933, 2012. 7

[13] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017. 2

[14] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3475–3484, 2016. 2

[15] Kai Han, Rafael S Rezende, Bumsub Ham, Kwan-Yee K Wong, Minsu Cho, Cordelia Schmid, and Jean Ponce. Scnet: Learning semantic correspondence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1831–1840, 2017. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[17] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4507–4515, 2017. 2

[18] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, 2012. 2

[19] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2

[21] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 2

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[23] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018. 2

[24] Seungryong Kim, Dongbo Min, Bumsub Ham, Sangryul Jeon, Stephen Lin, and Kwanghoon Sohn. Fcss: Fully convolutional self-similarity for dense semantic correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6560–6569, 2017. 2

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015. 2

[27] Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. Deep supervision with shape concepts for occlusion-aware 3d object parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5465–5474, 2017. 7

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[29] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2010. 2

[30] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2368–2382, 2011. 2

[31] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2

[33] Jiayi Ma, Weichao Qiu, Ji Zhao, Yong Ma, Alan L Yuille, and Zhuowen Tu. Robust $l\_\{2\}e$ estimation of transformation for non-rigid registration. *IEEE Transactions on Signal Processing*, 63(5):1115–1129, 2015. 2

[34] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. 2

[35] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 2

[36] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 2

[37] David Novotny, Diane Larlus, and Andrea Vedaldi. Anchornet: A weakly supervised network to learn geometry-sensitive features for semantic matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5277–5286, 2017. 2

[38] Masatoshi Okutomi and Takeo Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):353–363, 1993. 2

[39] Mustafa Ozuysal, Vincent Lepetit, and Pascal Fua. Pose estimation for category specific multiview object localization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785. IEEE, 2009. 7

[40] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016. 2, 4

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2, 3, 5, 6, 8

[43] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 2, 5

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 3, 4, 5

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 2

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[47] Ryan Szeto and Jason J Corso. Click here: Human-localized keypoints as guidance for viewpoint estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1595–1604, 2017. 3, 5

[48] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5916–5925, 2017. 2

[49] Nikolai Ufer and Bjorn Ommer. Deep semantic feature matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6914–6923, 2017. 2

[50] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 2

[51] Jianyu Wang, Cihang Xie, Zhishuai Zhang, Jun Zhu, Lingxi Xie, and Alan Yuille. Detecting semantic parts on partially occluded objects. *arXiv preprint arXiv:1707.07819*, 2017. 6

[52] Jianyu Wang, Zhishuai Zhang, Cihang Xie, Vittal Premachandran, and Alan Yuille. Unsupervised learning of object semantic parts from internal states of cnns by population encoding. *arXiv preprint arXiv:1511.06855*, 2015. 2, 3, 4, 6

[53] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82. IEEE, 2014. 6

[54] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378, 2017. 2

[55] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015. 2

[56] Hongsheng Yang, Wen-Yan Lin, and Jiangbo Lu. Daisy filter flow: A generalized discrete approach to dense correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3406–3413, 2014. 2

[57] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016. 5

[58] Zhishuai Zhang, Cihang Xie, Jianyu Wang, Lingxi Xie, and Alan L Yuille. Deepvoting: a robust and explainable deep network for semantic part detection under partial occlusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1372–1380, 2018. 1, 2, 3, 5, 6, 8

[59] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014. 2, 3

[60] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016. 2

[61] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2