
キーポイント検出とマッチングのためのマルチビューConvNetの機能学習

Multi-View ConvNet Feature Learning for Keypoint Detection and Matching

ジュンフン クオン*
Junghyun KWON

ラミア ナラシンハ*
Ramya NARASIMHA

シルビオ サヴァレス**
Silvio SAVARESE

要 旨

キーポイント検出とマッチングに向けたデータ駆動の多視点特徴量学習のアプローチを提案する。我々は検出する対象と、マッチングの手法を同時に畳み込みネットワーク (ConvNet) に学習させることができる4重サイアミーズ・アーキテクチャを開発した。

この4重サイアミーズ・アーキテクチャは4つのConvNetを持ち、それらは2つのサブネットワークに分類される。最初の3つのConvNetは多視点特徴量学習のため、最後の1つはキーポイント検出学習のために用いられる。

我々の多視点特徴量学習は、異なるキーポイント間より、同一キーポイントの類似性をより高く得ることができ、位置精度が高いキーポイントマッチングを実現することができる。

我々が提案する新たなConvNet学習のフレームワークによって、キーポイント検出とマッチング性能が、キーポイント位置の正確性と偽陽性率において最先端手法より良好な結果が得られることを示す。

ABSTRACT

We propose a data-driven multi-view feature learning approach for keypoint detection and matching. We train convolutional networks (ConvNet) using a novel quadruple siamese architecture that allows our ConvNet to learn “what to detect” and “how to match” simultaneously. The quadruple siamese architecture is composed of two sub-networks: the first three for multi-view feature learning and the last one for keypoint detection learning. Our multi-view feature learning allows us to obtain features that are more similar for the same keypoint pairs than the different keypoint pairs even with different views and instances, and thus enable robust keypoint matching while our keypoint detection learning allows robust keypoint detection with high localization accuracy. We experimentally show that by our novel ConvNet learning framework we can achieve keypoint detection and matching performance that outperforms the state-of-the-art method in keypoint localization accuracy and false positive rate.

* リコーイノベーションズコーポレーション
Ricoh Innovations Corporation

** リコーイノベーションズコーポレーション スタンフォード大学
Ricoh Innovations Corporation, Stanford University

1. Introduction

A popular paradigm in object detection and matching is to represent objects as a collection of keypoints (or parts). As demonstrated in early works¹⁻⁴⁾ and more recent works⁵⁻⁸⁾, a part-based representation has the critical advantage of making object detection and matching robust to: (i) occlusions or self-occlusions—if only a portion of the object is visible, fewer keypoints can still be used to successfully identify or match the target object; (ii) intra-class variations—keypoints are typically selected by retaining appearance and shape properties that are shared across the majority of the instances within the same object class. More recent representations^{9,10)} have successfully generalized these ideas so as to capture the intrinsic 3D nature of object categories and enable methods for detecting and matching objects even when they are observed under arbitrary viewpoint conditions.

A key challenge in part-based representations, however, is to guarantee that the “same” part can be detected regardless of the viewpoint while still accounting for intra-class variations. For instance, as Fig. 1 shows, in order to match a taillight from the car on the left with the same taillight from the car on the right, one needs to account for the relative appearance difference due to viewpoint transformations and intra-class variations.

In this paper we present a new approach for detecting and matching keypoints/parts that is based on a novel convolutional network (ConvNet) architecture. Inspired by the recent successful works¹¹⁻¹³⁾ whereby ConvNets are used to detect object keypoints from images using a large training set of annotated keypoints, the goal of our architecture is not only to detect keypoints but also to “learn” how to match corresponding keypoints as they are observed under different viewpoints. At that end, we introduce a novel ConvNet training architecture shown in Fig. 2. The lower convolutional layers are trained by a quadruple siamese network sharing weights whereby the first three networks (a triple siamese architecture) are

dedicated for keypoint matching and the fourth one is dedicated for keypoint detection.

We train a ConvNet using the triple siamese architecture so as to match corresponding keypoints in a discriminative fashion by feeding to the network triplets of input keypoints. Each triplet is composed by two corresponding keypoints and one negative keypoint that is not supposed to be in correspondence with any of the other two. This way the network learns how to establish correspondences in relationship to negative matches. On the other hand, the last subnetwork is trained for the keypoint detection to predict keypoint-like regions from a single input image. By the use of the weight sharing mechanism, the four subnetworks are eventually trained to have the same weights, and it is equivalent to have a single network optimized for both feature learning and keypoint detection.

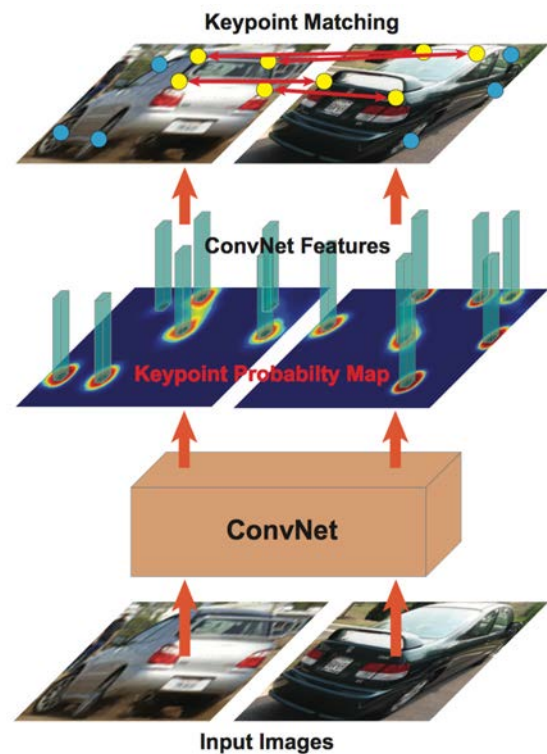


Fig. 1 We first detect keypoints from the keypoint probability map and then match detected keypoints across different views and instances using features learned by our multi-view ConvNet feature learning framework.

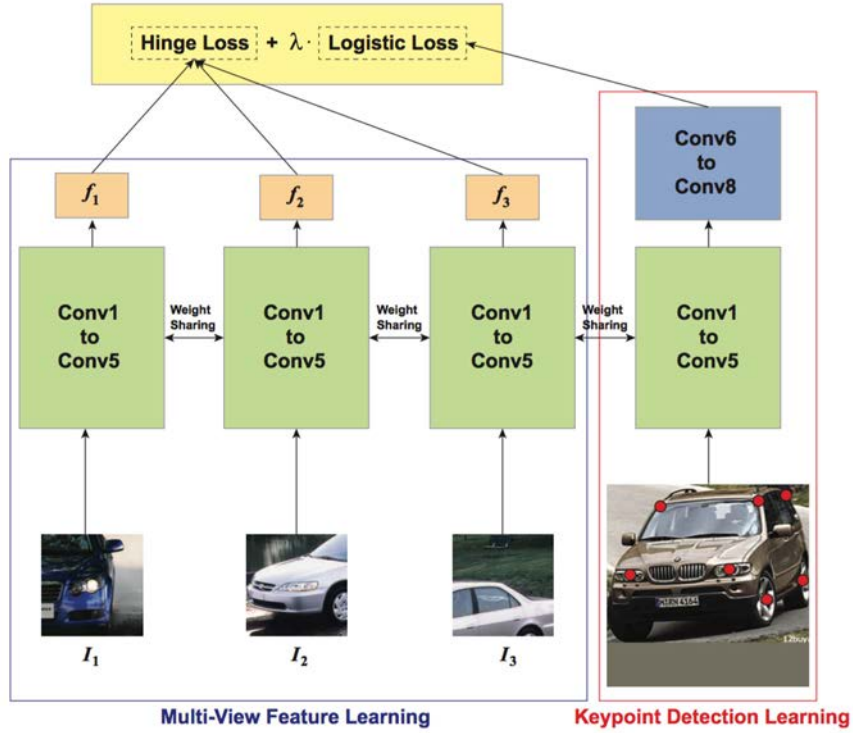


Fig. 2 Our ConvNet architecture for simultaneous multi-view feature learning and keypoint detection learning.

We report an extensive quantitative and qualitative experimental analysis that demonstrates our theoretical findings and shows that our method achieves superior results to the state-of-the-art approach in solving keypoint detection and matching tasks with higher keypoint localization accuracy and lower false positive rate.

1-1 Related Work

ConvNet feature learning using siamese network architectures has been studied well. For example, such networks have been used for signature verification¹⁴⁾, dimensionality reduction¹⁵⁾, and face verification¹⁶⁾. Recently, triple siamese architectures were used for learning fine-grained image similarity¹⁷⁾ and unsupervised ConvNet learning with videos¹⁸⁾. In this paper, we propose a novel quadruple siamese architecture to simultaneously train our ConvNet for both learning features for robust keypoint matching and learning how to detect keypoints.

Object detection and segmentation have been solved well by the ConvNet-based approaches. R-CNN computes

ConvNet features from region proposals and uses them to detect objects¹⁹⁾. Hariharan et al. solved fine-grained segmentation by the use of the hypercolumns²⁰⁾ while Long et al. solved semantic segmentation by classifying every pixel by FCN²¹⁾. Chen et al. combined ConvNet feature maps with the fully connected conditional random fields for accurate semantic segmentation²²⁾. In this paper, instead of trying to segment keypoint regions, we train a ConvNet to generate the spatial keypoint probability map that can be used for localizing keypoints accurately.

In order to match detected keypoints, we use the features extracted from the feature maps of the proposed ConvNet. There have been several attempts to use ConvNet features as off-the-shelf features^{23,24)}. We experimentally show that our learned ConvNet features significantly outperforms the off-the-shelf ConvNet features in keypoint matching.

There are several recent works to solve keypoint prediction problems by the ConvNet-based keypoint regression in various applications, e.g., facial point

detection¹¹⁾, human pose estimation¹²⁾, and keypoint prediction for general classes¹³⁾. Among these, the work by Tulsiani and Malik¹³⁾ is the most similar to ours as it also detects keypoints based on the spatial keypoint probability map. However, it has important limitations: (i) it needs another ConvNet to provide viewpoint information, (ii) the keypoint localization accuracy is limited by the use of coarse keypoint probability grids, and (iii) it predicts all keypoints regardless of visibility. We experimentally show that our method yields far better keypoint localization accuracy with far less false positives.

2. Overview of Our Approach

Figure 1 summarizes our ConvNet-based approach for keypoint detection and matching. From input images, our ConvNet generates keypoint probability map for keypoint detection and features for keypoint matching. We detect keypoints from the keypoint probability map. These keypoints are then matched across different views and instances using ConvNet features extracted at the detected locations. The main goal here is to simultaneously optimize our ConvNet for the two different tasks: learning features for robust keypoint matching (multi-view feature learning in Section 3) and learning how to detect keypoints (keypoint detection learning in Section 4).

Figure 2 shows the proposed quadruple siamese architecture that can train our ConvNet for multi-view feature learning and keypoint detection learning simultaneously. The main idea is to regard the main roles of each layer differently: we use the lower convolutional layers for both multi-view feature learning and keypoint detection learning while the higher convolutional layers exclusively for keypoint detection learning. The main reason to split the layers in this way is the receptive fields for the lower layers are suitable for multi-view feature learning where the mid-level features are important while the larger receptive fields of the higher layers are suitable

for telling whether a pixel belongs to keypoint regions based on the larger context.

Our quadruple siamese architecture is composed of two sub-networks marked as blue and red boxes in Fig. 2. The first sub-network (blue box) takes triplets of image patches that are composed of the same and different keypoint pairs and is trained for multi-view feature learning while the second sub-network (red box) takes whole object images and is trained for keypoint detection learning along with the higher layers. The overall loss function is the weighted combination of the hinge loss for multi-view feature learning and logistic loss for keypoint detection learning. Note that the higher layers are trained by only the logistic loss as they have no relation with the hinge loss. The lower layers are updated by the weighted loss and optimized for both multi-view feature learning and keypoint detection learning.

We now present the details of our multi-view feature learning and keypoint detection learning in Section 3 and Section 4, respectively. Once keypoints are detected from input images, we can identify them via voting using template images as explained in Section 5 and Fig. 3.

3. Multi-View Feature Learning

The goal of multi-view feature learning is to train our ConvNet to yield features to enable robust keypoint matching across different views and instances. Suppose that there are two images I_1 and I_2 of two different instances taken from different views with several visible keypoints at locations $k_1, k_2 \in \mathcal{R}^2$. We define a function $F(I, t)$ that extracts a feature vector f from an input image I at a certain location $t \in \mathcal{R}^2$. With a similarity function s , what we want for robust matching is

$$s\left(F(I_1, k_{I_1}^i), F(I_2, k_{I_2}^{j=i})\right) \gg s\left(F(I_1, k_{I_1}^i), F(I_2, k_{I_2}^{j \neq i})\right) \quad (1)$$

i.e., the features from the same keypoints are much more similar to each other than those from the different

keypoints are. We directly encode the relationship (1) into our multi-view feature learning using the hinge loss function. In the remainder of the paper, we use f_1 , f_2 and f_3 , to refer to features extracted from $F(I_1, k_{I_1}^i)$, $F(I_2, k_{I_2}^{j=i})$, and $F(I_2, k_{I_2}^{j \neq i})$ respectively.

We use a triple siamese network sharing weights for our multi-view feature learning (see the blue box in Fig. 2). The siamese network takes as an input a triplet of image patches taken from two image I_1 and I_2 that have common visible keypoints. The input patch for the first network is taken from I_1 at $k_{I_1}^i$ while those for the second and third networks are taken from I_2 at $k_{I_2}^{j=i}$ and $k_{I_2}^{j \neq i}$, respectively. After processing the input images by multiple convolutional layers, we extract f_1 , f_2 and f_3 at each keypoint location from the output feature maps of each network. Then the weights are learned by the following hinge loss function

$$L = \max(0, M - d) \quad (2)$$

where $d = s(f_1, f_2) - s(f_1, f_3)$ and M is a margin to ensure that $s(f_1, f_2)$ is much greater than $s(f_1, f_3)$. For the similarity function, we use the cosine similarity.

One can consider the hypercolumns²⁰⁾ extracted at keypoint locations as the feature vector f . We can extend the idea of the hypercolumns by considering the features in the local neighborhood of keypoints. We define our feature f as the hypercolumns extracted from the average pooled feature maps in addition to the original feature maps. The rationale for this is quite intuitive: if all the features within two regions are similar to each other, their average features should be similar as well.

4. Keypoint Detection Learning

The goal of keypoint detection learning is to train our ConvNet to yield the spatial probability map of keypoints from input images as shown in Fig. 1. In order to achieve this goal while optimizing our ConvNet also for multi-view feature learning, we augment more convolutional

layers on top of the convolutional layers used for multi-view feature learning (see the red box in Fig. 2). The lower convolutional layers are shared for multi-view feature learning and keypoint detection learning while the augmented higher layers are used only for keypoint detection learning. We use the logistic loss with ground truth (GT) keypoint label images to train our ConvNet for keypoint detection learning.

We obtain the keypoint probability map by applying a logistic function to the 1-channel feature map from the last convolutional layer upsampled to the input image sizes. We then find blobs from the keypoint probability map and regard their centroids as detected keypoints.

5. Keypoint Identification by Voting

After detecting keypoints from input images, we can identify them by assigning labels. We use template images with keypoint annotations from datasets such as the PASCAL3D+²⁵⁾ to perform this task. Figure 3 illustrates our voting approach for keypoint identification with template images. The first step is to select the top-k candidates similar to an input image in terms of the keypoint feature similarity. We perform the nearest neighbor matching of detected keypoints to keypoints of each template image using our learned features. We then find the top-k candidate templates based on the summation of the cosine similarity scores of the matched template keypoints.

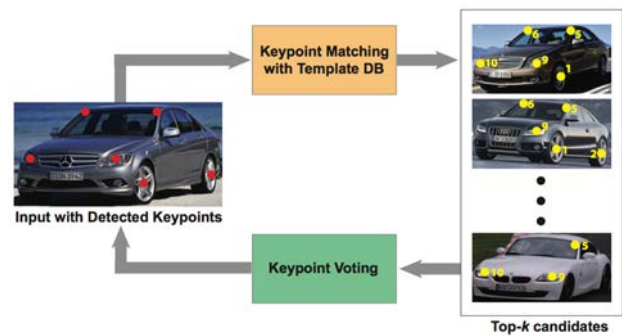


Fig. 3 Our voting approach for keypoint identification.

The second step is to find IDs (i.e. keypoint labels) of the detected keypoints based on the voting with the cosine similarity scores of the matched template keypoints from the selected top- k candidates. We build a $N \times M$ table where N is the number of detected keypoints and M is the number of all keypoint IDs. We then accumulate the cosine similarity scores of each matched template keypoint in the table cells corresponding to the detected keypoint and matched keypoint IDs. The ID of each detected keypoint is finally determined as the one that gives the highest accumulated cosine similarity score. This voting approach is simple and straightforward and we can get quite satisfactory keypoint identification results. This is because of superior keypoint matching ability of our features learned by the proposed multi-view feature learning framework.

6. Implementation Details

6-1 ConvNet Architecture

Our ConvNet architecture is based on the 16-layer model of the VGG architecture (VGG16)²⁶⁾. We use the Conv1 to Conv5 layers for multi-view feature learning while all layers for keypoint detection learning.

For multi-view feature learning, we compute 5×5 average pooled feature maps of the three sub-layers of Conv5 and extract our 3072-dimensional feature f from their original and average pooled features maps. We empirically set $M = 0.3$ for the hinge loss (2).

In order to increase keypoint localization accuracy, in addition to the feature map from Conv8, we generate 1-channel feature maps using 1×1 kernels from feature maps from the lower layers whose spatial resolution is higher than Conv8 by following the work by Long et al.²¹⁾. We use the last sub-layers of Conv5 and Conv4 instead of Pool4 and Pool3 used in the work by Long et al.²¹⁾ because the highest layer among those with the same spatial

resolution has more discriminative power due to more feature abstraction.

6-2 ConvNet Training

We trained our ConvNet by the stochastic gradient descent algorithm using the Caffe framework²⁷⁾. For initial weights, we used the VGG16 pre-trained for the ImageNet classification tasks²⁸⁾ except for the problem-specific Conv8 layer initialized with zero weights.

For training data, we used the PASCAL3D+²⁵⁾ that has keypoint and viewpoint annotations for 12 rigid object categories of the PASCAL VOC²⁹⁾ and ImageNet²⁸⁾. We used the whole ImageNet images and PASCAL train set for our ConvNet training. We generated 480×480 training samples by cropping the GT bounding box regions and resizing them to make the longer side length 480. We filled the void regions with the mean values of each color channel of the ImageNet images.

6-3 Training Data for Feature Learning

In order to generate the triplet training samples suitable for our multi-view feature learning, we first split all training samples into 16 azimuth angle bins, $a_i = 0, \frac{1}{8}\pi, \frac{2}{8}\pi, \dots, \frac{15}{8}\pi$, based on the PASCAL3D+ annotations. From the nearby bins, we generated pairs of images to be used for generating triplets. We regarded a_{i+1} and a_{i+2} as the nearby bins of a_i . The images were paired only when the azimuth and elevation differences are less than 60° and 15° , respectively, and the subtypes are the same. Finally, triplets are generated from each pair by considering all possible combinations of the same and different keypoint pairs. Our goal with these training triplets is to obtain features that are consistent for the same keypoint pairs even with appearance variations expected from such different views and different instances.

Because of the spatial dimension of the Conv5 layer's feature maps, we have to constraint the input dimension to multiples of 16. Therefore, we extract 208×208 patches centered at keypoint locations. This is closest to

and larger than the 196×196 receptive field size of Conv5. With this size of input patches, the spatial dimension of the feature maps of the Conv5 sub-layers are 13×13 and our features are extracted at the center of those feature maps. Figure 4 shows examples of triplet training samples for a “Car” category.



Fig. 4 Examples of triplet training samples for a “Car” category used in our multi-view feature learning.

6-4 Training Data for Keypoint Detection

For keypoint detection learning, it is not effective to use only GT keypoint locations as positive samples because they are too few compared to numerous negative samples. In order to solve this problem, we generated GT masks of keypoint regions by drawing circles centered at keypoint locations. We adjusted the radii of circles using the object distance to the camera with the minimum and maximum limits. Figure 5 shows examples of our training samples and their keypoint region GT masks for a “Car” category. By using these keypoint region masks, we can train our ConvNet to generate higher probabilities for regions that can contain keypoints.



Fig. 5 Examples of training samples and GT keypoint masks for a “Car” category for keypoint detection learning.

Figure 6 shows examples of detected keypoints by finding blobs from the keypoint probability map of our

trained ConvNet. Note that we can detect keypoints correctly despite clutter and low image quality. We can control the sensitivity of detection by changing the probability threshold p ; the results in Fig. 6 are with $p = 0.15$ that yields a good balance of precision and recall.

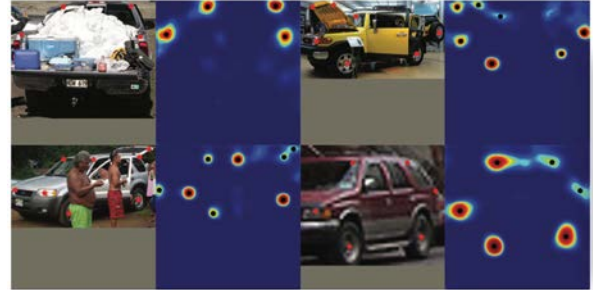


Fig. 6 Examples of keypoint detection results for a “Car” category.

7. Experiments

In this section, we first demonstrate the keypoint matching performance of our learned ConvNet features. We then demonstrate the performance of our keypoint detection and matching via comparison with the work by Tulsiani and Malik¹³⁾ (VPKP). We use the “Car” category of the PASCAL validation set for experiments.

7-1 Experiment 1: Keypoint Matching with Known Keypoint Locations

We first test the keypoint matching performance of our learned ConvNet features. We generated image pairs from the “Car” category of the PASCAL validation set as done in Section 6.3. The total number of image pairs are 1028 and the total number of GT keypoint matches are 4703.

We consider the following features for comparison: hypercolumns from the original feature maps (“HC”), hypercolumns from the original and average pooled feature maps (“HC+Pool”), our learned features from the original feature maps (“Ours (HC)”), and our learned features from the original and average pooled feature maps (“Ours (HC+Pool)”). We used the Conv5 feature maps for all cases. “HC” and “HC+Pool” are obtained

from the VGG16 pre-trained for the ImageNet tasks. We also consider the geometric blur descriptor³⁰⁾ (“GB”) as a baseline. We extract each feature from the known keypoint locations and perform keypoint matching by the nearest neighbor matching with cross check. We use the cosine similarity for all features except “GB”. We use the normalized cross correlation for “GB” as suggested in the work of Berg and Malik³⁰⁾.

Table 1 summarizes the matching performance of the compared features. It is clear that our learned features yielded significantly better keypoint matching performance than any other compared features. The increases in precision and recall of “Ours (HC+Pool)” compared to “HC+Pool” are 0.249 and 0.344, respectively. It is also notable that features using both original and average pooled feature maps consistently outperform those using only the original feature maps. Also note that the geometric blur descriptor shows far inferior matching performance compared to ours.

Table 1 Keypoint matching results with known keypoint locations.

| | GB | HC | HC+Pool | Ours (HC) | Ours (HC+Pool) |
|-----------|-------|-------|---------|-----------|----------------|
| Precision | 0.288 | 0.558 | 0.576 | 0.811 | 0.825 |
| Recall | 0.186 | 0.375 | 0.391 | 0.715 | 0.735 |

7-2 Experiment 2: Keypoint Detection, Identification, and Matching

We now test the performance of our overall approach for keypoint detection and matching via comparison with VPKP in three different aspects: (i) keypoint detection without identification, (ii) keypoint prediction (i.e., keypoint detection with identification), and (iii) keypoint detection and matching. We use the GT bounding box annotations for this comparison. Note that we used the PASCAL3D+ keypoint annotations²⁵⁾ for training and evaluation while VPKP used the PASCAL keypoint dataset⁷⁾. We believe that the overall difficulties in keypoint prediction using both annotations are quite

similar as the test images are the same and number of keypoints are similar: 12 in the PASCAL3D+²⁵⁾ and 14 in the PASCAL keypoint dataset⁷⁾ for the “Car” category.

7-2-1 Keypoint Detection without Identification

We first compare the keypoint detection performance of our method and VPKP without keypoint identification. In order to identify true positives from the detected keypoints, we greedily match the detected keypoints to the GT keypoints by (i) performing the nearest neighbor matching with cross check using the pixel distance and (ii) repeating it until there are no unmatched GT or detected keypoints. The detected keypoints are determined as true positives if they are matched to one of the GT keypoints and the distance between them is less than $\alpha \cdot \max(w, h)$ where α is the detection threshold and w and h are width and height of the GT bounding box.

Figure 7 shows our PR curves for the keypoint detection $\alpha = 0.1$ obtained with different keypoint probability thresholds p . Ours with $p = 0.15$ yielded much higher precision (+0.308) with slightly lower recall (-0.0284). The main reason of poor precision of is it predicts all keypoints regardless of visibility.

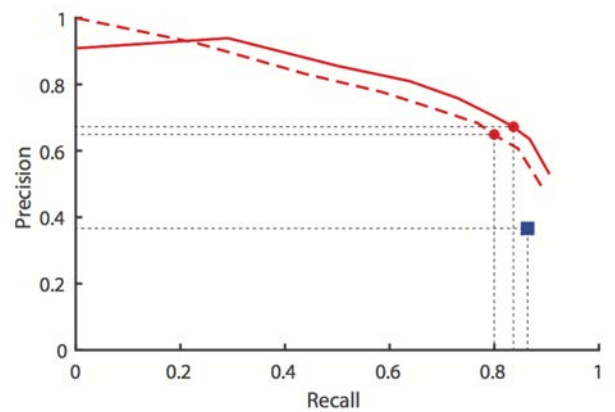


Fig. 7 Our PR curves for keypoint detection with $\alpha = 0.1$ using original (solid) and jittered (dashed) GT bounding boxes. The red dots represent our operating points with the keypoint probability threshold $p = 0.15$. The blue square is the precision and recall of VPKP²⁷⁾.

In order to check the robustness of our keypoint detection to errors in object bounding boxes, we repeated the experiment with the jittered GT bounding boxes. The dashed line in Fig. 7 shows our PR curve obtained with the jittered GT bounding boxes. We can see that our method is quite robust to errors in object bounding boxes: the decreases in precision and recall with $p = 0.15$ are only 0.0251 and 0.0356, respectively.

In order to better quantify the keypoint localization accuracy, we computed precision and recall with different values of α as shown in Table 2. Our precision and recall are with $p = 0.15$. Note that recall of VPKP significantly decreases with stricter detection thresholds unlike ours. Our recall with $\alpha = 0.03$ is 0.642 while that of VPKP is only 0.370. The significant decrease in recall of VPKP with stricter detection thresholds is due to the fact that it predicts keypoints based on coarse keypoint probability grids. Contrarily, our ConvNet generates the keypoint probability map of the same size as input images and can localize keypoints far more accurately than VPKP.

Table 2 Comparison of keypoint detection performance with different values of α .

| α | | 0.1 | 0.07 | 0.05 | 0.03 |
|-----------|------|--------------|--------------|--------------|--------------|
| Precision | [27] | 0.365 | 0.330 | 0.269 | 0.156 |
| | Ours | 0.673 | 0.653 | 0.616 | 0.517 |
| Recall | [27] | 0.864 | 0.782 | 0.637 | 0.370 |
| | Ours | 0.836 | 0.811 | 0.766 | 0.642 |

7-2-2 Keypoint Prediction

We now compare our keypoint prediction (i.e., keypoint detection with identification) performance with that of VPKP. We use as the performance measures the probability of correct keypoint (PCK), which measures the fraction of images where each keypoint is correctly localized³¹⁾, together with precision and recall. We used the keypoint detection results obtained with $p = 0.15$ and used the top-25 candidates for keypoint identification.

Table 3 summarizes the keypoint prediction performance by our method and VPKP with the different

values of α . As the keypoint prediction performance also depends on the keypoint localization accuracy, both mean PCK and recall of VPKP significantly decrease with stricter detection thresholds. The mean PCK and recall of VPKP with $\alpha = 0.03$ are only 0.292 and 0.275, respectively, while ours are 0.511 and 0.503, respectively.

Table 3 Comparison of keypoint prediction performance with different values of α .

| α | | 0.1 | 0.07 | 0.05 | 0.03 |
|-----------|------|--------------|--------------|--------------|--------------|
| Mean PCK | [27] | 0.756 | 0.669 | 0.530 | 0.292 |
| | Ours | 0.618 | 0.604 | 0.579 | 0.511 |
| Precision | [27] | 0.303 | 0.268 | 0.210 | 0.113 |
| | Ours | 0.498 | 0.486 | 0.464 | 0.405 |
| Recall | [27] | 0.741 | 0.653 | 0.512 | 0.275 |
| | Ours | 0.619 | 0.603 | 0.576 | 0.503 |

Figure 8 shows examples of keypoint prediction results by our method and VPKP. It is clear that the keypoint localization accuracy of VPKP is far inferior to ours because of the coarse keypoint probability grids. Also note that our method generated far less false positives.

7-2-3 Keypoint Detection and Matching

We finally compare the keypoint detection and matching performance of ours and VPKP. For this comparison, we used the same image pairs used in Section 7.1. We detect keypoints with $p = 0.15$ and match them using the nearest neighbor matching with cross check. In order to be counted as correct matches, both of matched keypoints should be true positives and have the same GT IDs. For keypoint matching performance of VPKP, we match their predicted keypoints across image pairs based on their predicted IDs. In this case, in order to be counted as correct matches, both of matched keypoints should be true positives.

Table 4 shows precision and recall of ours and VPKP for the keypoint detection and matching with the different values of α . The results are quite similar to the previous experiments: compared to VPKP, our precision is consistently much higher and our recall is far better with

stricter detection thresholds because of superior localization accuracy.

Table 4 Comparison of keypoint detection and matching performance with different values of α .

| α | | 0.1 | 0.07 | 0.05 | 0.03 |
|-----------|------|--------------|--------------|--------------|--------------|
| Precision | [27] | 0.381 | 0.342 | 0.239 | 0.0882 |
| | Ours | 0.615 | 0.601 | 0.579 | 0.488 |
| Recall | [27] | 0.820 | 0.737 | 0.514 | 0.190 |
| | Ours | 0.542 | 0.529 | 0.510 | 0.430 |



Fig. 8 Examples of keypoint prediction results by our method and VPKP¹³. Red and cyan circles represent true and false positives ($\alpha = 0.1$), respectively.

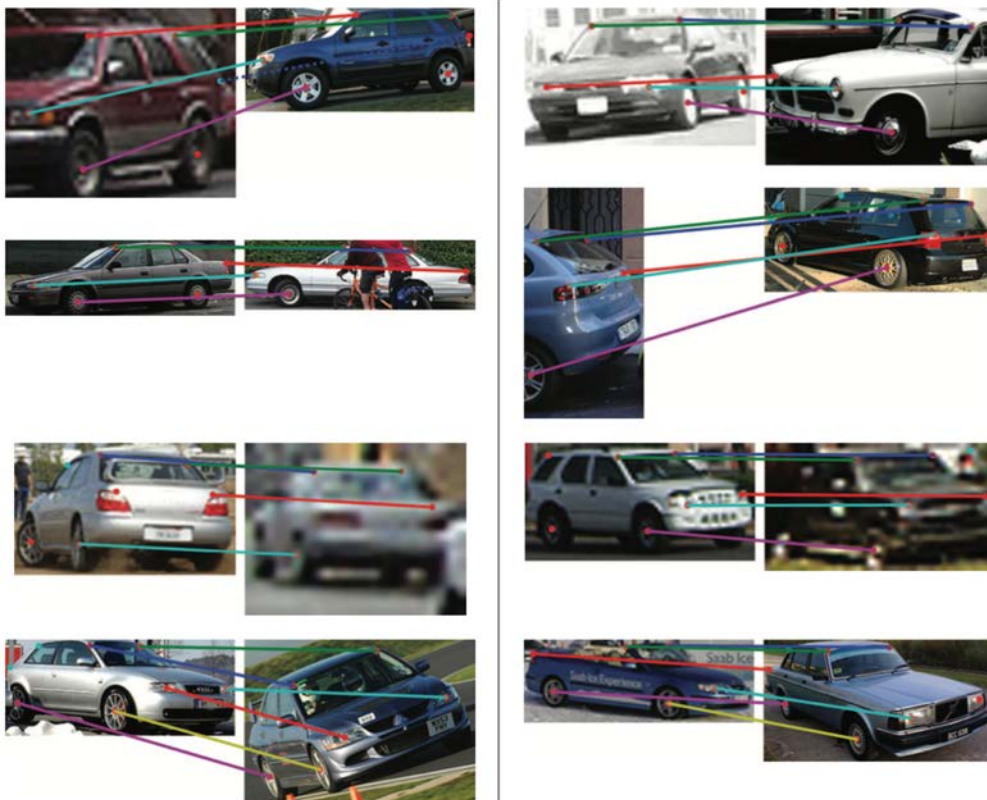


Fig. 9 Examples of keypoint detection and matching results by our method with the probability threshold of $p = 0.15$. Red and cyan circles are true and false positives ($\alpha = 0.1$), respectively. Solid and dashed lines are correct and incorrect keypoint matches, respectively. Note that all incorrect matches shown here are due to missing ground truth keypoint annotations.

8. Conclusion

In this paper, we proposed a novel quadruple siamese architecture to simultaneously train our ConvNet for both (i) learning features for robust keypoint matching using triplet training samples composed of the same and different key- point pairs, and (ii) learning how to detect keypoints. We detect keypoints from the keypoint probability map generated by our trained ConvNet and match them across different views and instances using our learned ConvNet features. We showed that our novel ConvNet learning framework can yield keypoint detection and matching performance that outperforms the state-of-the-art method in the keypoint localization accuracy and false positive rate via extensive quantitative and qualitative experimental comparisons.

References

- 1) P. Felzenszwalb, D. P. Huttenlocher: Pictorial structures for object recognition, *International Journal of Computer Vision*, 61(1):55–79 (2005).
- 2) R. Fergus, P. Perona, A. Zisserman: Object class recognition by unsupervised scale-invariant learning, In *CVPR* (2003).
- 3) B. Leibe, A. Leonardis, B. Schiele: Combined object categorization and segmentation with an implicit shape model, In *ECCV Workshops* (2004).
- 4) X. Ren, A. C. Berg, J. Malik: Recovering human body configurations using pairwise constraints between parts, In *ICCV* (2005).
- 5) L. Bourdev et al.: Detecting people using mutually consistent poselet activations, In *ECCV* (2010).
- 6) L. Bourdev, S. Maji, J. Malik: Describing people: A poselet-based approach to attribute classification, In *ICCV* (2011).
- 7) L. Bourdev, J. Malik: Poselets: Body part detectors trained using 3d human pose annotations, In *ICCV* (2009).
- 8) P. Felzenszwalb, D. McAllester, D. Ramanan: A discriminatively trained, multiscale, deformable part model, In *CVPR* (2008).
- 9) B. Pepik et al.: 3d2pm–3d deformable part models, In *ECCV* (2012).
- 10) S. Savarese, L. Fei-Fei: 3d generic object categorization, localization and pose estimation, In *ICCV* (2007).
- 11) Y. Sun, X. Wang, X. Tang: Deep convolutional network cascade for facial point detection, In *CVPR* (2013).
- 12) A. Toshev, C. Szegedy: Deeppose: Human pose estimation via deep neural networks, In *CVPR* (2014).
- 13) S. Tulsiani, J. Malik: Viewpoints and keypoints, In *CVPR* (2015).
- 14) J. Bromley et al.: Signature verification using a siamese time delay neural network, *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688 (1993).
- 15) R. Hadsell, S. Chopra, Y. LeCun: Dimensionality reduction by learning an invariant mapping, In *CVPR* (2006).
- 16) S. Chopra, R. Hadsell, Y. LeCun: Learning a similarity metric discriminatively, with application to face verification, In *CVPR* (2005).
- 17) J. Wang et al.: Learning fine-grained image similarity with deep ranking, In *CVPR* (2014).
- 18) X. Wang, A. Gupta: Unsupervised learning of visual representations using videos, *arXiv preprint arXiv:1505.00687* (2015).
- 19) R. Girshick et al.: Rich feature hierarchies for accurate object detection and semantic segmentation, In *CVPR* (2014).
- 20) B. Hariharan et al.: Hypercolumns for object segmentation and fine-grained localization, In *CVPR* (2015).

- 21) J. Long, E. Shelhamer, T. Darrell: Fully convolutional networks for semantic segmentation, In *CVPR* (2015).
- 22) L.-C. Chen et al.: Semantic image segmentation with deep convolutional nets and fully connected crfs, In *ICLR* (2015).
- 23) Y. Hou, H. Zhang, S. Zhou: Convolutional neural network-based image representation for visual loop closure detection, *arXiv preprint arXiv:1504.05241* (2015).
- 24) A. S. Razavian et al.: CNN features off-the-shelf: an astounding baseline for recognition, In *CVPR Workshops* (2014).
- 25) Y. Xiang, R. Mottaghi, S. Savarese: Beyond pascal: A benchmark for 3d object detection in the wild, In *WACV* (2014).
- 26) K. Simonyan, A. Zisserman: Very deep convolutional networks for large-scale image recognition, In *ICLR* (2015).
- 27) Y. Jia et al.: Caffe: Convolutional architecture for fast feature embedding, *arXiv preprint, arXiv:1408.5093* (2014).
- 28) O. Russakovsky et al.: ImageNet large scale visual recognition challenge, *International Journal of Computer Vision*, pp. 1–42 (2014).
- 29) M. Everingham et al.: The PASCAL visual object classes challenge: A retrospective, *International Journal of Computer Vision*, 111(1):98–136 (2015).
- 30) A. C. Berg, J. Malik: Geometric blur for template matching, In *CVPR* (2001).
- 31) Y. Yang, D. Ramanan: Articulated human detection with flexible mixtures of parts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2878–2890 (2013).