FCSS: Fully Convolutional Self-Similarity for Dense Semantic Correspondence - Supplementary Materials -

Anonymous CVPR submission

Paper ID 2965

In this supplemental materials, we provide more detailed analyses and results for the fully convolutional self-similarity (FCSS) descriptor.

- In Sec. 1, we describe the detailed relationship of the FCSS descriptor with conventional local self-similariy (LSS)-based descriptors [12, 8, 9].
- In Sec. 2, we describe the detailed configurations of network architecture in the FCSS descriptor.
- In Sec. 3, we provide the differentiability of convolutional self-similarity (CSS) layer in the FCSS descriptor in detail.
- In Sec. 4, we provide more results in four datasets, including that of Taniai et al. [16], Proposal Flow [5], the PASCAL dataset [2], and Caltech-101 [4].

1. The Relationship of the FCSS Descriptor with Conventional LSS-based Descriptors

In this section, we describe the relationship between the FCSS descriptor with conventional LSS-based descriptors, including local self-similarity (LSS) [12], dense adaptive self-correlation (DASC) [8], and deep self-correlation (DSC) [9]. Generally, LSS-based descriptors aim to represent locally self-similar structure around a given pixel by recording the similarity between certain patch pairs within a local window. Formally, they can be described as a vector of feature values $D_i = \bigcup_l D_i(l)$ for $l \in \{1, ..., L\}$, where the feature values are computed as

$$D_i(l) = \max_{j \in \mathcal{N}_i} \exp\left(-\mathcal{S}\left(P_{j-s_l}, P_{j-t_l}\right)/\lambda\right), \tag{1}$$

where $\mathcal{S}(P_{i-s_l}, P_{i-t_l})$ is a self-similarity distance between two patches P_{i-s_l} and P_{i-t_l} sampled on s_l and t_l, the l^{th} selected sampling pattern, around center pixel i. To alleviate the effects of outliers, the self-similarity responses are encoded by non-linear mapping with an exponential function of a bandwidth λ [1]. For spatial invariance to the position of the sampling pattern, the maximum self-similarity within a spatial window $\mathcal{N}_i$ is computed.

Based on this basic framework, LSS has been formulated in various ways [12, 8, 9], using different self-similarity measures $\mathcal{S}(P_{i-s_l}, P_{i-t_l})$ and sampling strategies (s_l, t_l) for the patch pairs. Firstly, for measuring self-similarities $\mathcal{S}(P_{i-s_l}, P_{i-t_l})$, a simple sum of square differences (SSD) in LSS [12] or an adaptive self-correlation (ASC) in DASC [8] and DSC [9] have been utilized. However, these hand-crafted similarity measure cannot provide a robustness no longer on problems requiring high invariances, *e.g.*, semantic correspondence. Secondly, for sampling patterns (s_l, t_l), center-biased sampling patterns in LSS [12] or randomized sampling patterns in DASC [8] and DSC [9] have been employed. However, it is very challenging to find out optimal sampling patterns for reliably describing structure to non-rigid deformations under intra-class variations. Existing LSS-based methods are formulated with hand-crafted design, thus they have limited performance on semantic correspondences. Unlike these methods, our descriptor formulate LSS in a fully convolutional architecture, where self-similarity measure $\mathcal{S}(P_{i-s_l}, P_{i-t_l})$ and the patch sampling patterns (s_l, t_l) are both learned in a end-to-end manner. Table 1 summarizes the relationship of the FCSS descriptor with conventional LSS-based descriptors.

Methods	Self-Similarity $\mathcal{S}(P_{i-s_l}, P_{i-t_l})$	Sampling Pattern (s_l, t_l)	Pooling Scheme	Feature Dimension	Computational Time
LSS [12]	sum of square differences (SSD)	dense center-biased sampling patterns	max-pooling within local support-window	80 dim.	31s
DASC [8]	adaptive self-correlation (ASC)	sparse randomized sampling patterns	-	128 dim.	2.7s
DSC [9]	adaptive self-correlation (ASC)	dense randomized sampling patterns	max-pooling within local support-window	585 dim.	9.2s
FCSS	convolutional self-similarity	semi-dense learned sampling patterns	max-pooling within local window	192 dim.	1.4s

Table 1. Relationship of the FCSS descriptor with conventional LSS-based descriptors. Computational time is measured in an image with the size of 463×370.

2. Network Configurations in the FCSS Descriptor

In this section, we describe the detailed configurations of a network architecture in the FCSS descriptor, consisting of multi-scale convolutional similarity layers, a set of two-stream shifting transformer layers, non-linear gating layers, and max-pooling layers. Detailed configurations of the network architecture are summarized in Table 2.

The convolutional similarity network consists of eight convolutional layers. We used the ImageNet pretrained VGG-Net [14] from the bottom conv1 to the conv3-4 layer, with their network parameters as initial values. Each convolutional layer consists of 3×3 convolutional kernels with different depths. To provide greater discriminativeness, two max-pooling layers are followed with the stride 2 after conv1-2 and conv2-2 convolutional layers. Thus, the spatial resolution of convolutional activation after conv1-2 is the $1/2$ of origial spatial resolution of inputs. The spatial resolution of convolutional activation after conv2-2 is the $1/4$ of origial spatial resolution of inputs. All convolutional layers have non-linear gating with ReLUs, except for last convolutional layer conv3-4.

Three CSS layers are located after conv2-2, conv3-2, and conv3-4. Before each CSS layer, convolutional activations are normalized to have a L_2 norm [15]. Each two-stream shifting transformer layer have 4 network parameters with source and target sampling patterns, where each sampling patterns consists of x- and y-direction shifting parameters. Considering the trade-off between efficiency and robustness, the number of sampling patterns is set to 64, thus the total dimension of the descriptor is $L = 192$. After each two-stream shifting transformer layer, the responses are passed through a non-linear agting layer defined in Eq. (11) to alleviate the effects of outliers. Furthermore, since the pre-learned sampling patterns used in the CSS layers are fixed over an entire image, they may be sensitive to non-rigid deformation as described in [9]. To address this, we perform the max-pooling operation within a spatial window $\mathcal{N}_i$ with the size of 2×2 centered at a pixel i. Finally, since the intermediate activations are of smaller spatial resolutions than the original image resolution, we apply a bilinear upsampling layer [11] after each CSS layer.

	convolutional similarity net.								shifting transform. net.		
	cnv1-1	cnv1-2	cnv2-1	cnv2-2	cnv3-1	cnv3-2	cnv3-3	cnv3-4	sfn1	sfn2	sfn3
kernel	3×3	3×3	3×3	3×3	3×3	3×3	3×3	3×3	4×1	4×1	4×1
channel	64	64	128	128	256	256	256	256	64	64	64
stride	1	2	1	2	1	1	1	1	2	2	2
pad	1	1	1	1	1	1	1	1	1	1	1
pooling	-	max	-	max	-	-	-	-	max	max	max
up-sam.	-	-	-	-	-	-	-	-	bilin.	bilin.	bilin.
non-lin.	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	-	(11)	(11)	(11)

Table 2. Network architecture of the FCSS descriptor.

Algorithm 1 summarizes the FCSS network initialization.

Algorithm 1: Fully Convolutional Self-Similiarity (FCSS) Network

---

Parameters: The number of scales, the number of sampling patterns
/* **ImageNet pretrained VGG-Net initialization** */
1 : Initialize convolutional similarity network $\mathbf{W}_c$ with ImageNet pretrained VGG-Net from the bottom conv1 to the conv3-4 layers.
for $k = 1:3$ **do**
  /* **Convolutional Self-Similarity (CSS) Layer Level-k** */
  2 : Normalize the intermediate convolutional activations with L_2 normalization after $\mathbf{W}_c^k$.
  for $l = 1:64$ **do**
    3 : Build two-stream shifting transformer layers with parameters $\mathbf{W}_s^k$ and $\mathbf{W}_t^k$, with random initialization.
  end for
  4 : Build non-linear gating layer with parameters $\mathbf{W}_\lambda^k$, max-pooling layer, and bilinear up-sampling layer.
  5 : Normalize the responses with L_2 normalization.
end for
6 : Concatenate all three responses after three CSS layers.
7 : Normalize the final responses with L_2 normalization.

---

3. Differentiability of Convolutional Self-Similarity (CSS) Layer in the FCSS Descriptor

In this section, we provide more details of differentiability of CSS layer in the FCSS Descriptor. The inputs of CSS layer is an intermediate convolutional activation $\mathbf{A}_i$, the ourputs of CSS layer is the self-similarity as $\mathcal{S}(P_{i-\mathbf{W}_s}, P_{i-\mathbf{W}_t}) = \|\mathcal{F}(\mathbf{A}_i;\mathbf{W}_s) - \mathcal{F}(\mathbf{A}_i;\mathbf{W}_t)\|^2 = \|\mathbf{A}_{i-\mathbf{W}_s} - \mathbf{A}_{i-\mathbf{W}_t}\|^2$.

First of all, the derivative of the final loss $\mathcal{L}$ with respect to $\mathcal{S}(P_{i-\mathbf{W}_s}, P_{i-\mathbf{W}_t})$, *i.e.*, $\partial\mathcal{L}/\mathcal{S}(P_{i-\mathbf{W}_s}, P_{i-\mathbf{W}_t})$, can be the inputs of CSS when back-propagating the gradients of the final loss. This gradients can be transfered into two-stream shifting transformer networks such that

$$\frac{\partial\mathcal{L}}{\partial\mathbf{A}_{i-\mathbf{W}_s}} = \frac{\partial\mathcal{L}}{\mathcal{S}(P_{i-\mathbf{W}_s}, P_{i-\mathbf{W}_t})} \cdot 2(\mathbf{A}_{i-\mathbf{W}_s} - \mathbf{A}_{i-\mathbf{W}_t}), \tag{2}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{A}_{i-\mathbf{W}_t}} = \frac{\partial\mathcal{L}}{\mathcal{S}(P_{i-\mathbf{W}_s}, P_{i-\mathbf{W}_t})} \cdot 2(\mathbf{A}_{i-\mathbf{W}_t} - \mathbf{A}_{i-\mathbf{W}_s}), \tag{3}$$

Furthermore, to obtain the derivatives for the convolutional similarity layer and the shifting transformer layers, we compute the Taylor expansion of the shifting transformer activations, under the assumption that $\mathbf{A}_i$ is smoothly varying with respect to shifting parameters $\mathbf{W}_s$:

$$\begin{aligned}\mathbf{A}_{i-\mathbf{W}_s^n} =& \mathbf{A}_{i-\mathbf{W}_s^{n-1}} + (\mathbf{W}_s^n - \mathbf{W}_s^{n-1}) \circ \nabla\mathbf{A}_{i-\mathbf{W}_s^{n-1}} \\ & \mathbf{A}_{i-\mathbf{W}_s^{n-1}} + (\mathbf{W}_{s_\mathbf{x}}^n - \mathbf{W}_{s_\mathbf{x}}^{n-1})\nabla_\mathbf{x}\mathbf{A}_{i-\mathbf{W}_{s_\mathbf{x}}^{n-1}} + (\mathbf{W}_{s_\mathbf{y}}^n - \mathbf{W}_{s_\mathbf{y}}^{n-1})\nabla_\mathbf{y}\mathbf{A}_{i-\mathbf{W}_{s_\mathbf{y}}^{n-1}},\end{aligned} \tag{4}$$

where $\mathbf{W}_s^{n-1}$ represents the sampling patterns at the $(n-1)^{th}$ iteration during training, and $\circ$ denotes the Hadamard product. $\nabla\mathbf{A}_{i-\mathbf{W}_s^{n-1}}$ is a spatial derivative on each activation slice with respect to $\nabla_\mathbf{x}$ and $\nabla_\mathbf{y}$. By differentiating (4) with respect to $\mathbf{W}_{s_\mathbf{x}}^n$, we get the shifting parameter derivatives as

$$\frac{\partial\mathbf{A}_{i-\mathbf{W}_s^n}}{\partial\mathbf{W}_{s_\mathbf{x}}^n} = \nabla_\mathbf{x}\mathbf{A}_{i-\mathbf{W}_s^{n-1}}. \tag{5}$$

By the chain rule, with n omitted, the derivative of the final loss $\mathcal{L}$ with respect to $\mathbf{W}_{s_\mathbf{x}}$ can be expressed as

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{s_\mathbf{x}}} = \frac{\partial\mathcal{L}}{\partial\mathbf{A}_{i-\mathbf{W}_s}}\frac{\partial\mathbf{A}_{i-\mathbf{W}_s}}{\partial\mathbf{W}_{s_\mathbf{x}}}. \tag{6}$$

Similarly, $\partial\mathcal{L}/\partial\mathbf{W}_{s_\mathbf{y}}$, $\partial\mathcal{L}/\partial\mathbf{W}_{t_\mathbf{x}}$, and $\partial\mathcal{L}/\partial\mathbf{W}_{t_\mathbf{y}}$ can be calculated.

Finally, the derivative of the final loss $\mathcal{L}$ with respect to $\mathbf{A}_i$ can be formulated as

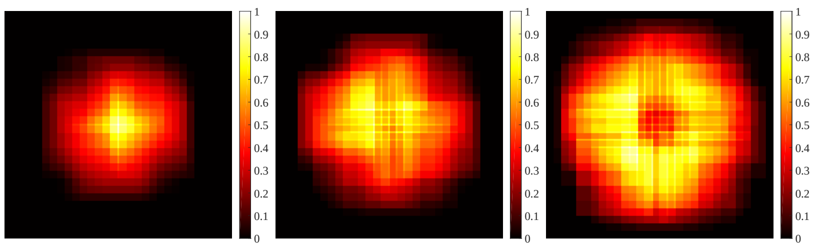
$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{A}_i} &= \frac{\partial\mathcal{L}}{\partial\mathbf{A}_{i-\mathbf{W}_s}}\frac{\partial\mathbf{A}_{i-\mathbf{W}_s}}{\partial\mathbf{A}_i} + \frac{\partial\mathcal{L}}{\partial\mathbf{A}_{i-\mathbf{W}_t}}\frac{\partial\mathbf{A}_{i-\mathbf{W}_t}}{\partial\mathbf{A}_i} \\ &= \frac{\partial\mathcal{L}}{\partial\mathbf{A}_{i-\mathbf{W}_s}} + \frac{\partial\mathcal{L}}{\partial\mathbf{A}_{i-\mathbf{W}_t}},\end{aligned} \tag{7}$$

since $\partial\mathbf{A}_{i-\mathbf{W}_s}/\partial\mathbf{A}_i$ is 1 on the pixel $i-\mathbf{W}_s$. In this way, the derivatives for the CSS layer can be computed.

4. More Results

In this section, we first represent the visualization of learned sampling patterns used in experiments, and then provide the additional results for our FCSS descriptor compared to state-of-the-art handcrafted descriptors and recent CNNs-based feature descriptors on Taniai et al. [16], Proposal Flow [5], the PASCAL dataset [2], and Caltech-101 [4].

Visualization of Learned Sampling Patterns Fig. 1 shows learned sampling patterns in convolutional self-similarity (CSS) layer of FCSS descriptor. For an effective visualization, we followed the practice used in [3]. We stacked all sampling patterns learnt from the Caltech-101 dataset [4] excluding testing image pairs used in experiments. A set of histogram bins corresponding to the patch of sampling patterns are incremented by one, and they are finally normalized with the maximum value. In low scale-level in Fig. 1(a) derived from the shallower convolutional layers, the density of sampling patterns tends to be concentrated on the center, which provides the precise localization ability. In high scale-level in Fig. 1(c) derived from the deeper convolutional layers, the sampling patterns can conver more large receptive fields within a support window, which provides high robustness for intra-class appearance variations. In shows that the optimal sampling patterns on each scale are learned in the FCSS descriptor.



(a) Scale 1 (b) Scale 2 (c) Scale 3

Figure 1. Visualization of learned sampling patterns in convolutional self-similarity (CSS) layer of FCSS descriptor.

Additional Results on Various Benchmarks Fig. 2 shows qualitative results compared to state-of-the-art correspondence techniques on the Taniai benchmark [16]. Fig. 3 and Fig. 4 show comparison of dense correspondence for various feature descriptor with fixed SF optimization [10] on the Taniai benchmark [16]. Fig. 5 show comparison of dense correspondence for various feature descriptor with fixed SF optimization [10] on the Proposal Flow benchmark [5]. Fig. 6 show comparison of dense correspondence for various feature descriptor with fixed SF optimization [10] on the PASCAL dataset [2]. Fig. 7 show comparison of dense correspondence for various feature descriptor with fixed SF optimization [10] on Caltech-101 [4].

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 2. Qualitative results compared to state-of-the-art correspondence techniques on the Taniai benchmark [16]: (a) source image, (b) target image, (c) DFF [18], (d) DSP [7], (e) Zhou *et al.* [21], (f) Taniai *et al.* [16], (g) Proposal Flow [5], and (h) FCSS w/PF [5]. The source images were warped to the target images using correspondences.

Figure 3. Comparison of dense correspondence for FG3DCar on the Taniai benchmark [16]. The results consist of warped target images and correspondence flow fields overlaid with source images. (from top to bottom) source and target image pairs, SIFT [10], DAISY [17], LSS [12], DASC [8], DeepD. [13], DeepC. [20], MatchN. [6], LIFT [19], VGG [14], VGG w/S-CSS, VGG w/M-CSS, and FCSS.

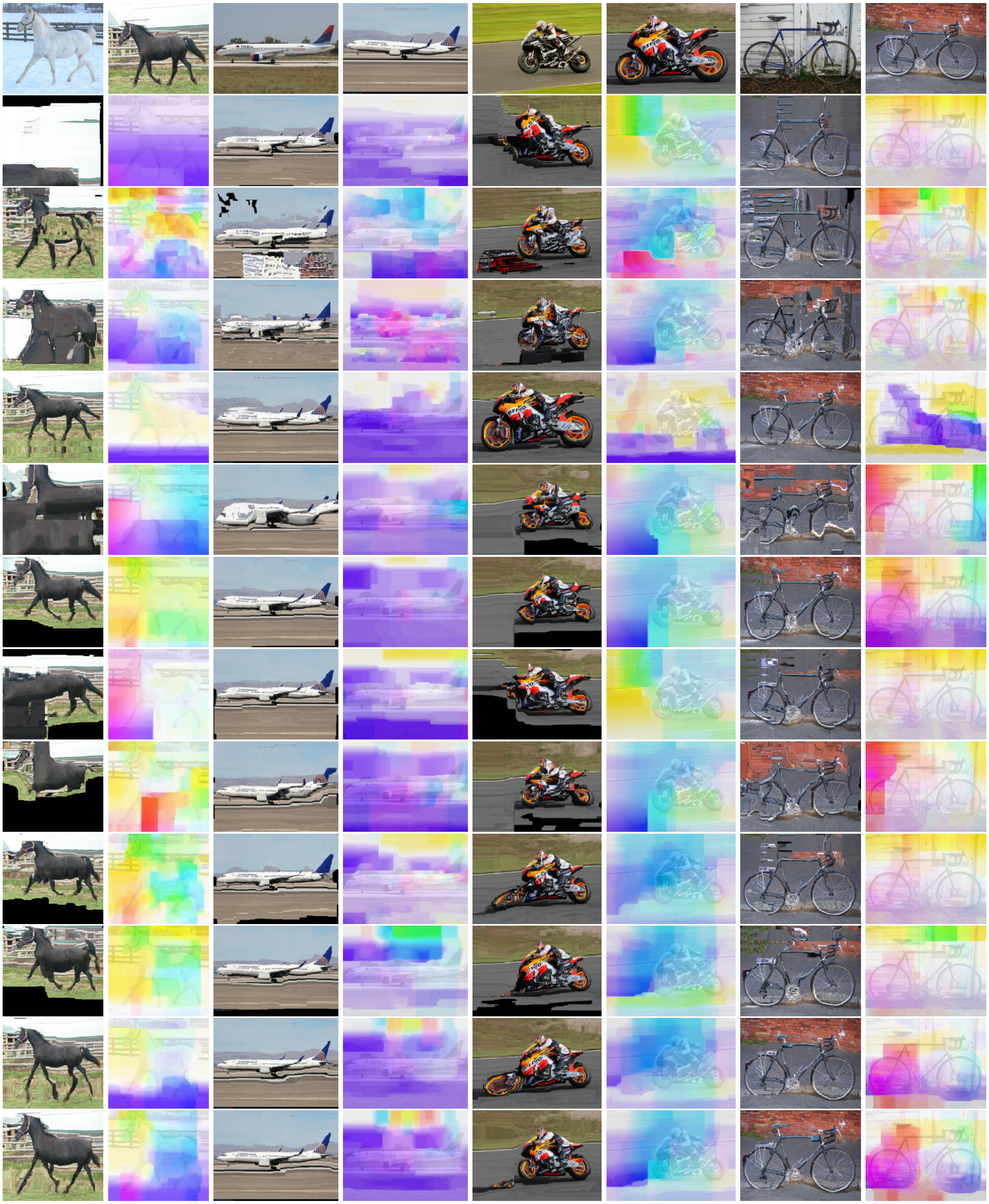


Figure 4. Comparison of dense correspondence for JODS and PASCAL on the Taniai benchmark [16]. The results consist of warped target images and correspondence flow fields overlaid with source images. (from top to bottom) source and target image pairs, SIFT [10], DAISY [17], LSS [12], DASC [8], DeepD. [13], DeepC. [20], MatchN. [6], LIFT [19], VGG [14], VGG w/S-CSS, VGG w/M-CSS, and FCSS.

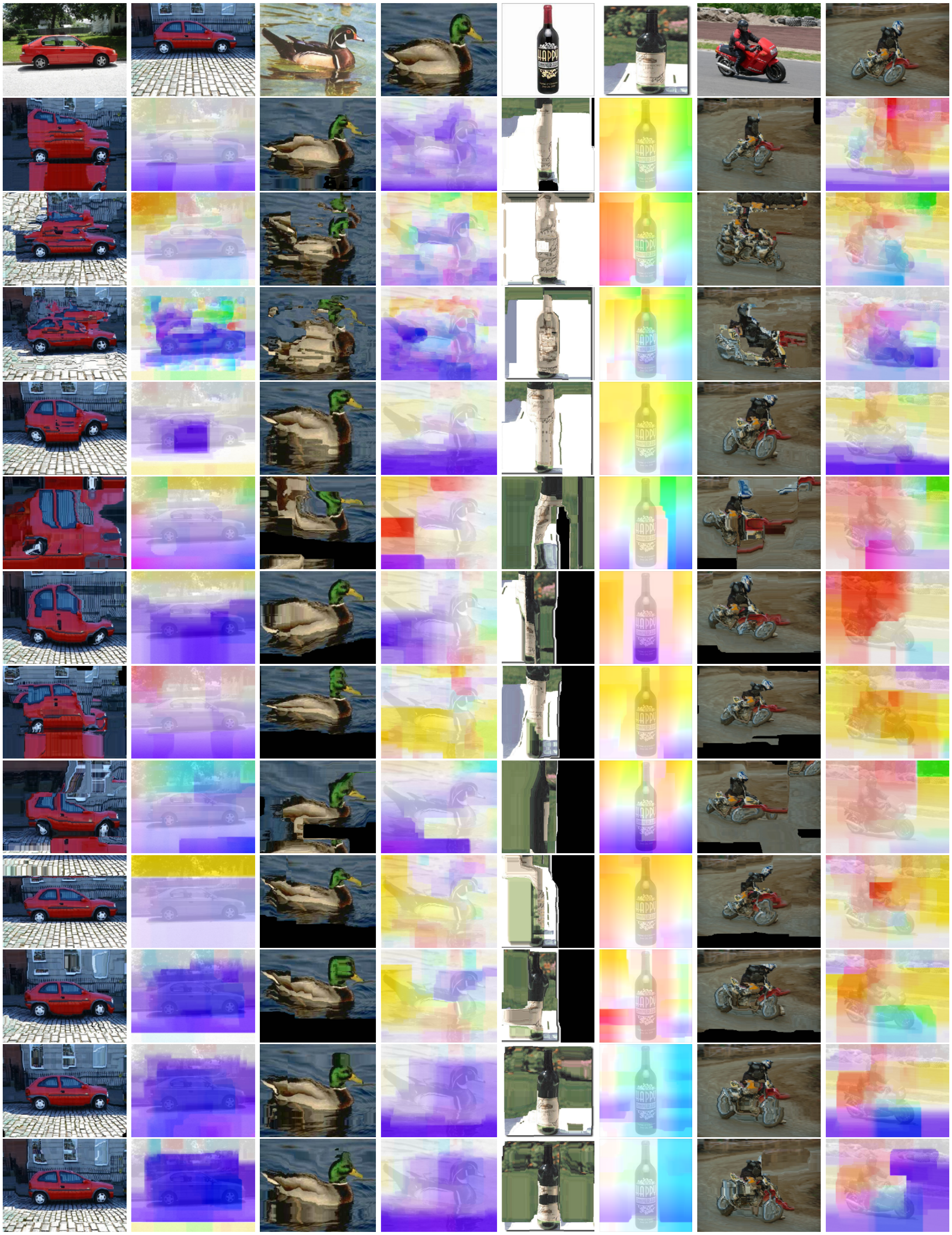


Figure 5. Comparison of dense correspondence on the Proposal Flow benchmark [5]. The results consist of warped target images and correspondence flow fields overlaid with source images. (from top to bottom) source and target image pairs, SIFT [10], DAISY [17], LSS [12], DASC [8], DeepD. [13], DeepC. [20], MatchN. [6], LIFT [19], VGG [14], VGG w/S-CSS, VGG w/M-CSS, and FCSS.

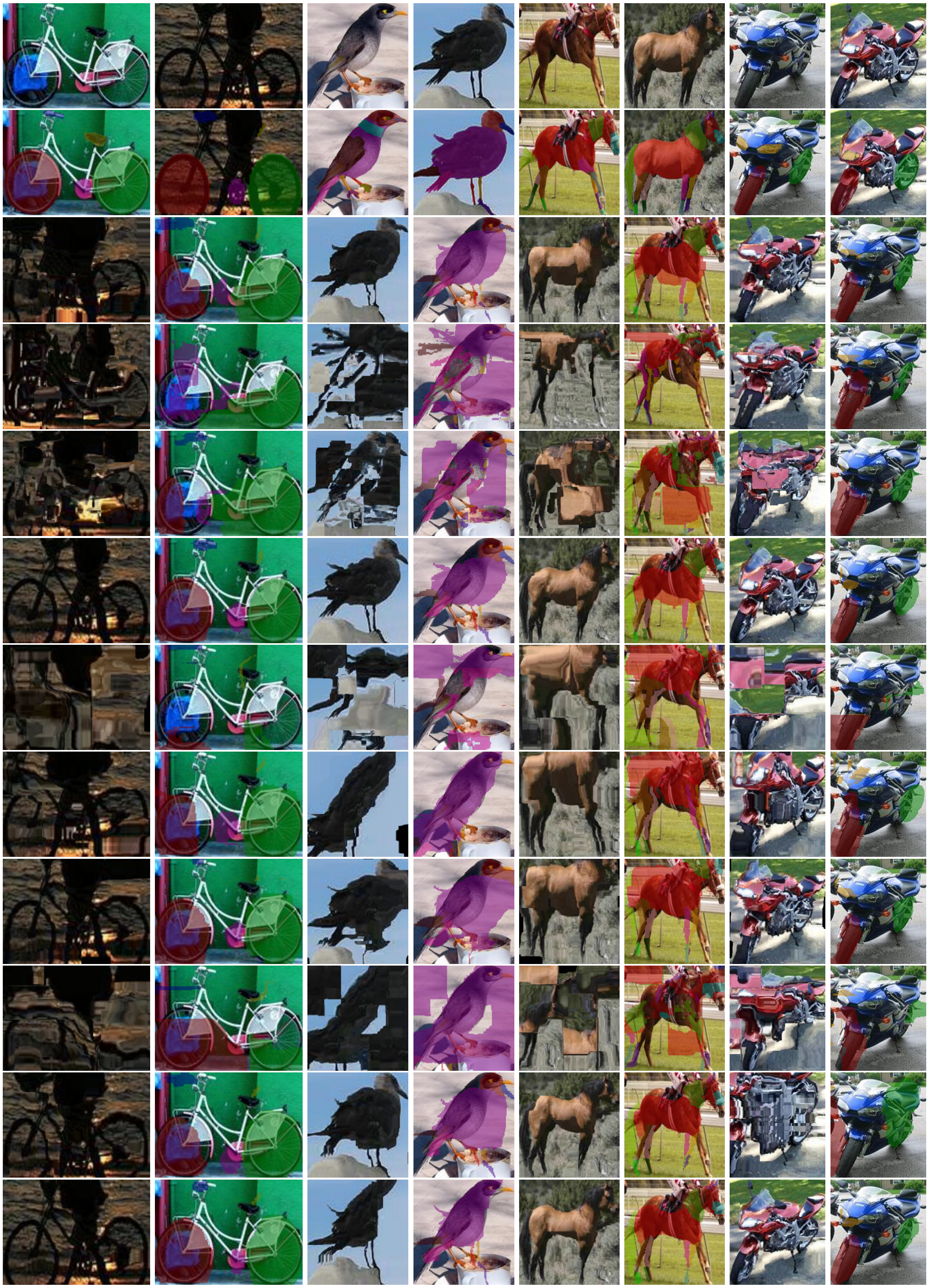


Figure 6. Comparison of dense correspondence with color-coded part segments on the PASCAL-VOC part dataset [2]. The results consist of warped target images and trasfered part segments overlaid with source images. (from top to bottom) source and target image pairs, source and target part segment image, SIFT [10], DAISY [17], LSS [12], DASC [8], DeepD. [13], DeepC. [20], MatchN. [6], LIFT [19], VGG [14], and FCSS.

Figure 7. Comparison of dense correspondence with mask transfer on the Caltech-101 dataset [4]. The results consist of warped target images and transfered mask overlaid with source images. (from top to bottom) source and target image pairs, source and target part segment image, SIFT [10], DAISY [17], LSS [12], DASC [8], DeepD. [13], DeepC. [20], MatchN. [6], LIFT [19], VGG [14], and FCSS.

References

[1] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. IP*, 7(3):421–432, 1998. 2

[2] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasum, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. *In: CVPR*, 2014. 1, 5, 9

[3] B. Fan, Q. Kong, T. Trzcinski, and Z. Wang. Receptive fields selection for binary feature description. *IEEE Trans. IP*, 23(6):2583–2595, 2014. 5

[4] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. PAMI*, 28(4):594–611, 2006. 1, 5, 10

[5] B. Ham, M. Cho, C. Schmid, and J. Ponce. Proposal flow. *In: CVPR*, 2016. 1, 5, 8

[6] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. *In: CVPR*, 2015. 6, 7, 8, 9, 10

[7] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. *In: CVPR*, 2013. 5

[8] S. Kim, D. Min, B. Ham, S. Ryu, M. N. Do, and K. Sohn. Dasc: Dense adaptive self-correlation descriptor for multi-modal and multi-spectral correspondence. *In: CVPR*, 2015. 1, 2, 6, 7, 8, 9, 10

[9] S. Kim, D. Min, S. Lin, and K. Sohn. Deep self-correlation descriptor for dense cross-modal correspondence. *In: ECCV*, 2016. 1, 2, 3

[10] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Trans. PAMI*, 33(5):815–830, 2011. 5, 6, 7, 8, 9, 10

[11] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *In: CVPR*, 2015. 3

[12] E. Schechtman and M. Irani. Matching local self-similarities across images and videos. *In: CVPR*, 2007. 1, 2, 6, 7, 8, 9, 10

[13] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. *In: ICCV*, 2015. 6, 7, 8, 9, 10

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In: ICLR*, 2015. 3, 6, 7, 8, 9, 10

[15] H. O. Song, Y. Xiang, S. Jegelk, and S. Savarese. Deep metric learing via lifted structured feature embedding. *In: CVPR*, 2016. 3

[16] T. Taniai, S. N. Sinha, and Y. Sato. Joint recovery of dense correspondence and cosegmentation in two images. *In: CVPR*, 2016. 1, 5, 6, 7

[17] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Trans. PAMI*, 32(5):815–830, 2010. 6, 7, 8, 9, 10

[18] H. Yang, W. Y. Lin, and J. Lu. Daisy filter flow: A generalized discrete approach to dense correspondences. *In: CVPR*, 2014. 5

[19] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. *In: ECCV*, 2016. 6, 7, 8, 9, 10

[20] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. *In: CVPR*, 2015. 6, 7, 8, 9, 10

[21] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. *In: CVPR*, 2016. 5