Contents

A. Overview

We include additional implementation details and results in Sec. B. These experiments demonstrate the following. First, the importance of both the distinctiveness score and CoAM to achieve sota results with comparison to other single stage approaches on the challenging YFCC100 dataset. Second, they demonstrate how our approach can use a refinement step to achieve state-of-the-art results for all thresholds >1px on HPatches. Third, they provide additional ablations including the use of a Noise Contrastive (NCE) loss as opposed to the hinge loss presented in the paper. Finally, we provide more results and explanation for the SfM results given in the main paper. We define the metrics used in obtaining the SfM results in Sec. B.3 and provide additional baselines. We also demonstrate that our model is more robust than the one using SIFT on a challenging, new dataset of Henry Moore sculptures. Finally, we provide visualisations of our reconstructed 3D models here and in the attached video.

Additionally, we provide further details of the architectures used in Sec. C for the CD-UNet, CoAM + CAPSNet [31], and the stylization model.

We also provide qualitative samples of the distinctiveness scores learned by our model in Sec. D.2. We provide additional qualitative results for the stylization experiment, HPatches dataset, SfM experiment, and Aachen dataset in Sec. D.3.

B. Additional Experiments

In this section we report the results of additional experiments which consider additional ablations of the grid size chosen at test time (Sec. B.1), a refinement of our model to improve local matching (Sec. B.2) to achieve state-of-the-art results on HPatches, further comparisons of our model on the 3D reconstruction task (Sec. B.4.1), results on the YFCC100M dataset (Sec. B.5), and results using another popular contrastive loss (NCE) (Sec. B.6).

B.1. Further Ablations

In this section we discuss and ablate how we select candidate matches at test time.

In order to compare all descriptor vectors at test time, we operate as follows. We create a $G \times G$ pixel grid and bilinearly interpolate on this grid from both the descriptor maps and distinctiveness scores. (Note that we have to normalize the interpolated descriptors.) We consider all pairs of descriptors as candidate matches and compare all descriptors in one image to those in the other. In practice we use $G = 128$.

At test time, we could use a larger grid size for better granularity. However, this comes with the additional computational cost of performing G^4 comparisons. We tried using a larger grid ($G = 256$) on the Aachen-Day Night dataset in Tab. 4 but obtained comparable results to using $G = 128$. As a result, we continued to use $G = 128$ for all our experiments. Using a larger grid for the larger datasets in SfM (where the number of images are approximately 1500) would have made these experiments intractable using current pipelines.

However, we note that because we only consider points on a grid, we are losing some granularity which presumably impacts the performance on the 3D reconstruction tasks. We discuss a method to refine the correspondences to obtain a finer granularity in the next section and the resulting complications.

B.2. Refining Local Matching

In this section, we demonstrate that our local matching results can be improved by using a simple refinement strategy.

Description of Refinement Strategy. In order to refine the matches obtained using CD-UNet, we use a local neighbourhood to refine the match in the second image. Given a correspondence with location (x_1, y_1) in the first image and (x_2, y_2) in the second image, we look at the similarity score between (x_1, y_1) and the locations in a 3x3 grid centered on (x_2, y_2).

These scores are used to reweight the location in the second image using a normalized weighted sum with one

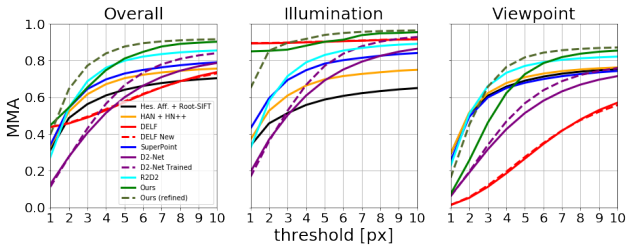


Figure 1: Results on HPatches when refining correspondences. In this experiment, we look at the results of using a refinement strategy based on a local neighbourhood to improve the accuracy of the detected correspondences. We see that using the refinement scheme, we obtain a big boost in performance. In particular, we improve our results for fine-grained pixel thresholds on the viewpoint task. We achieve comparable results with state-of-the-art methods for small pixel thresholds and better performance for larger thresholds ($>$ 4px). On illumination we maintain performance except for very small thresholds ($\leq$ 1px). This small degradation is probably due to limited noise that is introduced with the refinement strategy. In general, using this strategy we improve our results to achieve state of the art performance for all pixel thresholds.

difference. Because the similarity scores are not evenly distribute and cluster around 0.5, they give too much weight to less likely matches. As a result, we subtract the minimum similarity from all scores in a local neighbourhood before computing the normalized weighted sum.

Results. The results are given in Fig. 1. As can be seen, this simple refinement scheme gives a large boost in performance. In particular, using this simple modification gives superior results to state of the art approaches for pixel thresholds $>$ 4px for viewpoint and all thresholds $>$ 2px for illumination. Overall, our model with the refinement scheme achieves state-of-the-art results for all thresholds $>$ 1px.

Discussion. While we could achieve high quality performance on HPatches using this simple modification, we note that it is not straightforward to apply this to the camera localization or 3D reconstruction tasks. This is because both of these tasks require a single point to be tracked over multiple images in order to perform 3D reconstruction (the camera localization performs 3D reconstruction as part of the evaluation pipeline). 3D reconstruction pipelines assume a detect and describe pipeline for extracting matches, which implicitly have this assumption baked in to their setup, as they match the same detected points across different images.

However, this assumption is not implicit to our approach, as we only find correspondences between pairs of images at a time. Further refining the points means that the location of a refined point from one pair of images will not necessarily match that of another pair, invalidating the track and negatively affecting the 3D reconstruction. Incorporating these refined points would require rethinking how we incorporate refined correspondences in a 3D reconstruction pipeline. For example, we could use a reference image against which further images are compared and incorporated. Once a reference image has been exhausted, a new reference image would be chosen and so on. We leave such an investigation to future work, but the boost in performance demonstrates the further potential of our setup.

B.3. SfM Terminology

Here we define the metrics used in reporting the SfM results. Note that these metrics are only *indicative* of the quality of the 3D model; please look at the reconstructed models in the zipped video for a qualitative estimate as to their respective quality.

1. $\uparrow$ # Reg. Ims: **The number of registered images.** This is the number of images that are able to be put into correspondence and for which cameras were obtained when doing the 3D reconstruction. A higher values means more images were registered, implying a better model.

2. $\uparrow$ # Sparse Pts: **The number of sparse points.** This is the number of sparse points obtained after performing the 3D geometry estimation. The higher the number indicates a better model, as more correspondences were able to be triangulated.

3. $\uparrow$ Track Len: **The track length.** How many images a given 3D point is seen in on average. If this is higher, it indicates that the model is more robust, as more images see that 3D point.

4. $\downarrow$ Reproj err: **The repreojection error.** This is the average pixel error between a 3D point and its projection in the images. If this is lower, it indicates the 3D points are more accurate.

5. $\uparrow$ # Dense Points: **The number of dense points.** This is the number of dense points in the final 3D model. The higher this is, the more of the 3D structure was able to be reconstructed.

B.4. Further Results for SfM

We include additional baselines on the Local Feature Evaluation Benchmark of the original paper. We additionally include results in a challenging scenario where the dataset contains fewer ($\approx 10 - 100$ images) images of the same scene and where the object (a sculpture) may differ in material, location, and context.

B.4.1 Further Baselines on Local Feature Evaluation Benchmark

In this section we compare our 3D reconstruction on the Local Feature Evaluation Benchmark [27] to two additional

Table 1: SfM. We compare our approach to three baselines on 3D reconstruction for three scenes with a large number of images. Our method obtains superior performance across the metrics except for reprojection error, despite using coarse correspondences and a single scale. In particular, our method registers more images and obtains more sparse 3D points. ↑ denotes higher is better. ↓ denotes lower is better.

LMark	Method	↑ # Reg. Imgs	↑ # Sparse Pts	↑ Track Len	↓ Reproj. Err	↑ # Dense Pts
Madrid Metropolis 1344 images	RootSIFT [10]	500	116K	6.32	**0.60px**	**1.82M**
	GeoDesc [13]	495	144K	5.97	0.65px	1.56M
	D2 MS [3]	495	144K	**6.39**	1.35px	1.46M
	Ours	**702**	**256K**	6.09	1.30px	1.10M
Gendarmen-markt 1463 images	RootSIFT [10]	1035	338K	5.52	**0.69px**	**4.23M**
	GeoDesc [13]	1004	441K	5.14	0.73px	3.88M
	D2 MS [3]	965	310K	5.55	1.28px	3.15M
	Ours	**1072**	**570K**	**6.60**	1.34px	2.11M
Tower of London 1576 images	RootSIFT [10]	804	239K	**7.76**	**0.61px**	**3.05M**
	GeoDesc [13]	776	341K	6.71	0.63px	2.73M
	D2 MS [3]	708	287K	5.20	1.34px	2.86M
	Ours	**967**	**452K**	5.82	1.32px	1.81M

Table 2: SfM. We compare our approach to using SIFT features on 3D reconstruction on a challenging new SFM dataset of sculptures. This dataset contains images from the web containing large variations in illumination and viewpoint. These metrics are a proxy for 3D reconstruction quality, so we encourage the reader to view the reconstructions in the supplementary. X: failure. ↑: higher is better. ↓: lower is better.

		Sculpture Dataset								
	Landmark:	HM1	HM2	HM3	HM4	HM5	HM6	HM7	HM8	HM9
	# Images:	12	124	250	266	78	31	358	238	74
# Reg. Ims ↑	**SIFT [10]:**	X	103	**198**	212	61	22	266	**201**	53
	Ours:	**12**	**108**	194	**215**	**67**	**25**	**284**	**201**	**57**
# Sparse Pts ↑	**SIFT [10]:**	X	48K	70K	102K	28K	9K	128K	**99K**	**23K**
	Ours:	**2.9K**	**63K**	**83K**	**121K**	**40K**	**10K**	**190K**	**99K**	21K
Track Len ↑	**SIFT [10]:**	X	**5.33**	**5.92**	**5.80**	**4.54**	**4.73**	**4.46**	**5.24**	4.75
	Ours:	**3.60**	5.03	5.43	5.61	4.32	4.00	4.23	**5.24**	**4.77**
Reproj Err (px) ↓	**SIFT [10]:**	X	1.31	1.32	**1.28**	1.30	1.33	**1.22**	**1.30**	**1.32**
	Ours:	**1.33**	**1.30**	**1.30**	1.29	**1.29**	**1.26**	1.23	**1.30**	**1.32**
# Dense Pts ↑	**SIFT [10]:**	X	160K	143K	**307K**	73K	**46K**	174K	333K	**54K**
	Ours:	**0.2K**	**188K**	**156K**	296K	**82K**	44K	**187K**	**333K**	53K

baselines [3, 13] in Tab. 1. These results were not included in the paper as these additional baselines perform similarly to SIFT [10] and there was limited space. However, we note that both of these baselines use learned descriptors, yet they do not perform any better than SIFT in terms of the number of registered images and sparse 3D points. Our method performs significantly better across all three large scale datasets for obtaining sparse 3D points and registering images.

B.4.2 Further Results on a Sculpture SFM Benchmark

We use images from the Sculpture dataset [5], which consists of images of the same sculpture downloaded from the web. As an artist may create the same sculpture multiple times, a sculpture's material (e.g. bronze or marble), location, or context (e.g. the season) may change in the images (refer to the supplementary for examples). In particular, we evaluate on nine sculptures by the artist Henry Moore. These sets of images contain large variations and the sculpture itself is often smooth, leading to less texture for finding repeatably detectable regions.

We report the results in Tab. 2 and visualise samples in Fig. 10. While these metrics are proxies for reconstruction accuracy, our approach is able to consistently obtain more 3D points than the others for each image set. These results validate that our approach does indeed make our model robust in this context and it performs as well if not better than the SIFT baseline method.

B.5. The YFCC100M Dataset

Here we report results for our model and ablations on the YFCC100M [30] dataset. This dataset further demonstrates the superiority of our approach to other detect and describe setups and the utility of each component of our model (i.e. the distinctiveness score and CoAM).

Setup. The task of this dataset is to perform two view geometry estimation on four scenes with 1000 pairs each. Given a pair of images, the task is to use estimated corre-

Table 3: Results on the YFCC dataset [30]. Higher is better. Our approach outperforms all other detect and describe approaches (e.g. all but RANSAC-Flow) that operate directly on features for smaller angle errors and performs competitively for larger angle errors. Additionally, this dataset clearly demonstrates the utility of CoAM and the distinctiveness score. †Note that RANSAC-Flow [28] is a multi stage approach that iteratively registers images. Such an approach could be added on top of ours.

Method	mAP@5°	mAP@10°
SIFT [10]	46.83	68.03
Contextdesc [12]	47.68	69.55
Superpoint [2]	30.50	50.83
PointCN [17, 33]	47.98	-
PointNet++ [21, 34]	46.23	-
N^3Net [20, 34]	49.13	-
DFE [22, 34]	49.45	-
OANet [34]	52.18	-
RANSAC-Flow† [28]	64.68	73.31
Ours (w/o conf)	31.60	40.80
Ours (w/o cond)	53.43	65.13
Ours	55.58	66.79
Ours (E-B1)	**57.23**	**68.39**

spondences to predict the essential matrix using the known intrinsic matrices. The essential matrix is decomposed into the rotation and translation component [7]. The reported error metric is the percentage of images that have the rotation and translation error (in degrees) less than a given threshold.

To run CD-UNet on this dataset, we first use CD-UNet to extract high quality matches for each pair of images. We use the known intrinsics to convert these to points in camera space. We then use RANSAC [4] and the 5-point algorithm in order to obtain the essential matrix [7].

Results. The results are given in Tab. 3. They demonstrate that our model achieves superior performance for low error thresholds to other methods that directly operate on extracted features. The results further demonstrate that the distinctiveness score and CoAM are crucial for good performance, validating our model design.

Finally, our model does a bit worse than RANSAC-Flow [28]. However, we note [28] uses a segmentation model to restrict correspondences to only regions in the foreground of the image (e.g. the segmentation model is used to remove correspondences in the sky). Additionally, this method first registers images under a homography using predetected correspondences and then trains a network on top of the transformed images to perform fine-grained optical flow. As a result, this method performs as well as the underlying correspondences. Considering that our method has consistently been demonstrated to perform comparably or better than previous approaches for obtaining correspondences, we note that this method could be used on top of ours for presumably further improved performance. However, as this work was only recently published, we leave this for future work.

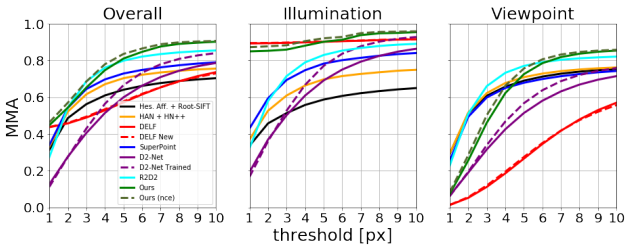


Figure 2: Results on HPatches using a NCE loss. In this experiment, we look at the results of using a NCE loss as opposed to a hinge loss. We see that using a NCE loss, we still achieve high quality results, demonstrating the robustness of our approach.

Table 4: Results on Aachen Day-Night using a NCE loss. We can see that training with NCE is slightly worse than our hinge loss, but it is competitive with other state-of-the-art methods. Higher is better. * indicates the method was trained on the Aachen dataset.

Method	Type	Threshold Accuracy		
		0.25m (2°)	0.5m (5°)	5m (10°)
Upright RootSIFT [10]	Spa	36.7	54.1	72.5
DenseSFM [25]	Den	39.8	60.2	84.7
Han+, HN++ [15, 14]	Spa	39.8	61.2	77.6
Superpoint [2]	Spa	42.8	57.1	75.5
DELF [18]	Spa	39.8	61.2	85.7
D2-Net [3]	Spa	**44.9**	66.3	**88.8**
R2D2* [23]	Spa	**45.9**	66.3	**88.8**
Ours (nce)	Den	42.9	62.2	87.8
Ours ($G = 256$)	Den	**44.9**	68.4	87.8
Ours	Den	**44.9**	70.4	**88.8**

B.6. An InfoNCE Loss

In the paper we demonstrate the robustness of our approach to the precise choice of architecture (e.g. we can achieve impressive results using a ResNet [8] or EfficientNet [29] backbone).

Here, we consider using the CPC objective [19], which is inspired by Noise-Contrastive Estimation (NCE) approaches [6, 16], and so is called an InfoNCE loss. While this is also a contrastive loss, similarly to the hinge loss in the main paper, the implementation is different. We find that we can still achieve impressive results with this loss, demonstrating the robustness of the approach to the precise choice of loss.

B.6.1 Implementation

We follow the implementation of [19], except that because we use normalized features, we add a temperature τ. This is essential for good performance.

The setup is the same as that described in the paper, except for the implementation of the loss. Assume we have two descriptor maps D^1 and D^1 corresponding to the two input

images I^1 and I^2. At a location i in D^1, we obtain the descriptor vector $d_i^1 \in \mathbb{R}^c$. To compare descriptor vectors, we first normalize and then use cosine similarity to obtain a scalar matching score:

$$s(d_i^1, d_j^2) = \left(\frac{d_i^1}{||d_i^1||_2} \right)^T \frac{d_j^2}{||d_j^2||_2}. \quad (1)$$

If the score is near 1, this is most likely a match. If it is near -1, it is most likely not a match.

Again, as in the paper, given two images of a scene I^1 and I^2 with a known set of correspondences from MegaDepth [9], we randomly select a set $\mathcal{P}$ of L true correspondences. For each positive correspondence p, we additionally select a set $\mathcal{N}_p$ of N negative correspondences. The loss $\mathcal{L}_{\texttt{nce}}$ is then

$$-\log \frac{1}{L} \sum_{p=(x,y)\in\mathcal{P}} \frac{e^{(\tau * s(d_x^1, d_y^2)}}{e^{(\tau * s(d_x^1, d_y^2))} + \sum_{(x,\hat{y})\in\mathcal{N}_p} e^{(\tau * s(d_x^1, d_{\hat{y}}^2))}} \quad (2)$$

where $\tau = 20$ is a temperature.

B.6.2 Experiments

We train the InfoNCE model using the $\mathcal{L}_{\texttt{nce}}$ in the same manner as in the paper and evaluate it on two of the datasets discussed in the paper: HPatches [1] and Aachen [25, 26].

HPatches. The results are given in Fig. 2. From here we see that our model with an NCE loss performs competitively on this dataset, obtaining superior results to that of the model in the paper.

Aachen Day-Night. The results are given in Tab. 4. These results demonstrate that using an NCE loss with our backbone achieves results competitive with other state-of-the-art approaches but it performs a bit worse than the hinge loss used in the paper.

Discussion. These experiments have shown that we can achieve high quality results when using a different but effective contrastive loss. As a result, our approach is robust to not only the backbone architecture (as shown in the paper) but also the precise choice of the contrastive loss.

C. Architectures

C.1. Architecture for CD-UNet

The components of the main model are described in the main text. Here we give further details of the different components. The encoder is a ResNet50 model, except that we extract the features from the last two blocks to obtain feature maps f_S^i and f_L^i. The details are given in Tab. 5.

The features f_L^i and f_S^i are projected in order to reduce the number of channels using linear layers. There are four linear layers (one for each of f_L^1, f_L^2, f_S^1, f_S^2). The linear layers operating at the larger resolution (f_L^i) project the features from 2048 size vectors to 256. The linear layers

Table 5: Encoder of CD-UNet. The encoder is a Resnet50 [8] encoder. The convolutions column denotes the convolutional and max-pooling operations. Implicit are the BatchNorm and ReLU operations that follow each convolution.

layer name	output size	convolutions
conv 1	128×128	$7 \times 7, 64$, stride 2
conv2_x	64×64	3×3 max pool, stride 2 $\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 3$
conv3_x	32×32	$\begin{pmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{pmatrix} \times 4$
conv4_x (f_L^i)	16×16	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 6$
conv5_x (f_S^i)	8×8	$\begin{pmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{pmatrix} \times 3$

Table 6: Decoder of CD-UNet. The decoder is a UNet [24] variant. The convolutions column denotes the convolutional operations. Implicit are the BatchNorm and ReLU operations that follow each convolution as well as the bi-linear upsampling operation that resizes features from the previous layer before the convolutional blocks.

layer name	inputs	output size	convolutions
deconv_5	conf5_x ($\times 2$), $\hat{f}_S^i$	16×16	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix}$
deconv_4	deconv_5, conv4_x, $\hat{f}_L^i$	32×32	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix}$
deconv_3	deconv_4, conv3_x	64×64	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix}$
deconv_2	deconv_3, conv2_x	128×128	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix}$
deconv_1	deconv_2, conv1_x	256×256	$\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix}$

Table 7: Encoder of CAPSNet [31] variant. The encoder is a ResNet34 [8] encoder. The convolutions column denotes the convolutional and max-pooling operations. Implicit are the BatchNorm and ReLU operations that follow each convolution.

layer name	output size	convolutions
conv 1	240×320	$7 \times 7, 64$, stride 2
conv2_x	120×160	3×3 max pool, stride 2 $\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix} \times 2$
conv3_x (f_L^i)	60×80	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix} \times 2$
conv4_x (f_S^i)	30×40	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix} \times 2$

Table 8: Decoder of CAPSNet [31]. The decoder is a UNet [24] variant. The convolutions column denotes the convolutional operations. Implicit are the BatchNorm and ReLU operations that follow each convolution as well as the bi-linear upsampling operation that resizes features from the previous layer before the convolutional blocks.

layer name	inputs	output size	convolutions
deconv_3	$\hat{f}_S^2, f_S^1$	60×80	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 512 \end{pmatrix}$
deconv_2	$\hat{f}_L^2, f_L^1$, deconv_3	120×160	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 512 \\ 3 \times 3, 256 \end{pmatrix}$
deconv_1	deconv_2, conv2_x	120×160	$\begin{pmatrix} 3 \times 3, 512 \\ 3 \times 3, 128 \end{pmatrix}$

operating at the smaller resolution (f_S^i) project the features from 1024 size vectors to 128.

The decoder consists of a sequence of decoder layers. A layer takes the bi-linearly upsampled features from the previous layer, the corresponding encoded features, and optionally the attended features. The details are given in Tab. 6. Finally, the unnormalized features are passed to a MLP which regresses the distinctiveness score. The MLP consists of three blocks of linear layer (with no bias) and batch normalization followed by a sigmoid layer. The channel dimensions are $64 \rightarrow 1 \rightarrow 1$.

C.2. Architecture for CoAM + CAPSNet [31]

Here we further describe the CAPSNet architecture and how we incorporate CoAMs into the architecture. We use the ResNet34 encoder (in order to fit a batch size of 6 on our GPUs). We extract the features from the 2nd and 3rd blocks to obtain feature maps f_L^i (at a fine level) and f_S^i (at a coarse level). The details are given in Tab. 7.

The features f_L^1, f_L^2, f_S^1, and f_S^2 are projected as above. The linear layers operating at the larger resolution (f_L^i) project the features from 128 size vectors to 16. The linear layers operating at the smaller resolution (f_S^i) project the features from 256 size vectors to 32.

The decoder operates as above. The details are given in Tab. 8. This gives the final set of descriptors of size 128D. We find that using just the fine features performs (output of deconv_1) better than usin a concatenation of the coarse (at a resolution 60×80 and fine features).

C.3. Architecture for Stylization

In Fig. 3, we illustrate further our method for stylizing images using our initial set of dense correspondences. Given two images I_v and I_S, the task is to generate an image with the viewpoint of I_v and the style of I_S. In brief, we first use the dense correspondences to sample from I_S to obtain the initial image. We then use a refinement network to fix errors and fill in missing regions. We do this in two stages. The first stage fixes errors and can be trained with an L1 loss. However, we wish to use a discriminator loss, but using this directly on the intermediary image causes information to leak and the generated image to match I_v, which is input to the network. To use a discriminator loss without leaking information, we use a second network (a sequence of ResNet blocks) which is trained with both an L1 and discriminator loss. Crucially, we do not allow gradients to flow from the second network (the set of ResNet blocks) to the first.

D. Additional Results

D.1. Further Visualisation of CoAM's Attention

In Fig. 4 we illustrate additional predicted attention maps from our CoAM when trained with CAPSNet [31] in a self-supervised framework. Again, it is not clear apriori what the model *should* attend to. However, in these results, we can see how the attention varies as a function of the query location and seems to attend to relevant regions.

D.2. Visualising the Distinctiveness Score

In Fig. 5 we illustrate the predicted distinctiveness score and in Fig. 6 the similarity score. Here we can see that the most confident parts of both images are regions that exist in *both* images. We further test this by looking at what the distinctiveness scores look like when we keep one input view the same but change the other in Fig. 7. We can see that the distinctiveness score changes depending on the input images: the output is indeed dependent on *both* input images.

To visualise the similarity score, we proceed as follows in Fig. 6. For a query point k in I_1, we display the correspondence map c_{kl}. The point in the sky matches a region around the building; a point on the arch matches a point on the arch (shown by the bright spot)

D.3. Qualitative Results

In Fig. 8 we show random matches obtained by our method on random samples from the Aachen Day-Night test set and similarly in Fig. 9 for HPatches, Fig. 10 for 3D reconstruction using SfM and Fig. 11 for the stylization task.

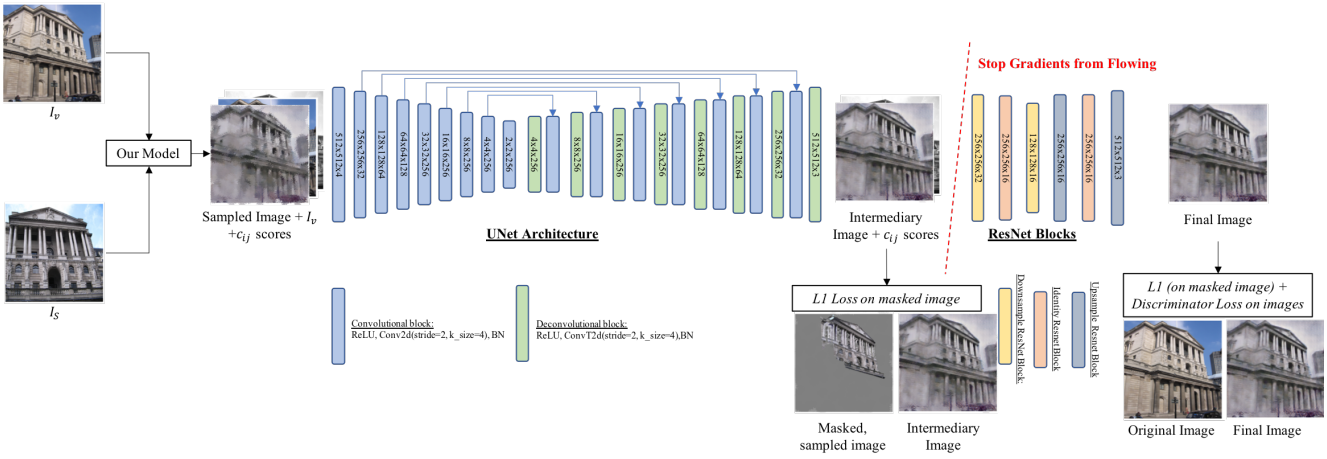


Figure 3: Illustration of the architecture used for the downstream stylization task. We first use our model to predict correspondences to transform the input image I_S into the position of I_v. The next step is to learn how to fill in and fix errors with a discriminator. However, we additionally can train the portion of the generated image visible in both input images to match the true transformed image. We also can use high frequency information in I_v when performing this transformation. However, if we train end-to-end then information will leak from I_v to the generated image. As a result, we train in two stages. We first generate an intermediary image using a UNet [24] which is trained using an L1 loss on the generated intermediary image for regions that are in common between the two images (and for which we can determine what the true pixel colour should be). We input this intermediary image to a set of ResNet blocks to refine the original prediction: this is trained with both a discriminator (pix2pixHD [32]) and L1 loss.

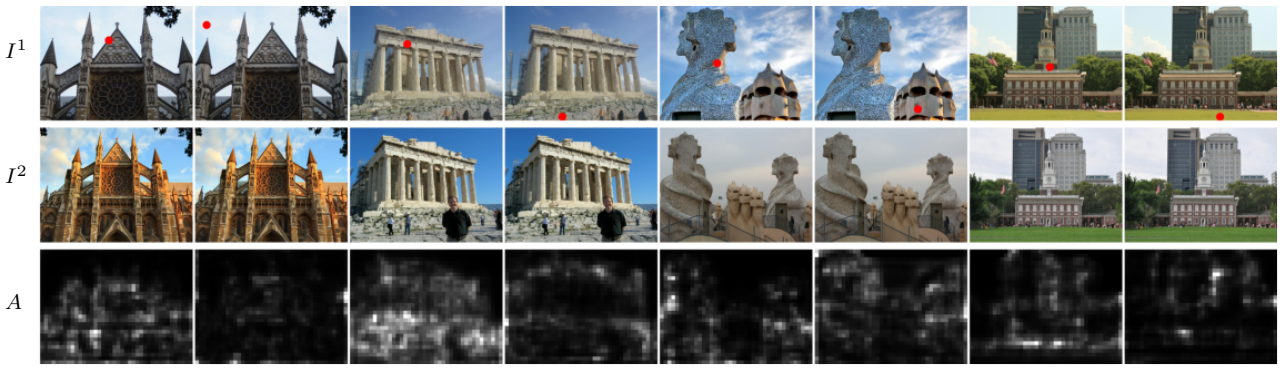


Figure 4: CoAM Attention. Here we visualize more samples of CoAM's predicted attention for sample image pairs. As in the main paper, the red dot in I^1 denotes the point for which we compute the attention. It is not clear apriori what the attention module should do but it does attend to relevant, similar regions in the other image and is dependent on the query location.

Figure 5: Random pairs associated with their predicted distinctiveness score on the MegaDepth test set. Top row shows the input pairs, bottom row the associated distinctiveness scores.

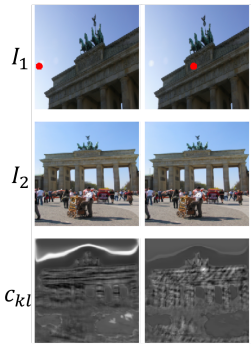


Figure 6: Pairs of images with their similarity scores on the MegaDepth test set. For a query point k in I_1, we display the correspondence map c_{kl}. The point in the sky matches a region around the building; a point on the arch matches a point on the arch (shown by the bright spot).

Figure 7: Random pairs associated with their predicted distinctiveness score on the MegaDepth test set. Top: Pairs of image of the same scene. Bottom: Pairs with one image from the top associated with another from a different scene. Notice that the distinctiveness score changes between the two cases and becomes irrelevant.

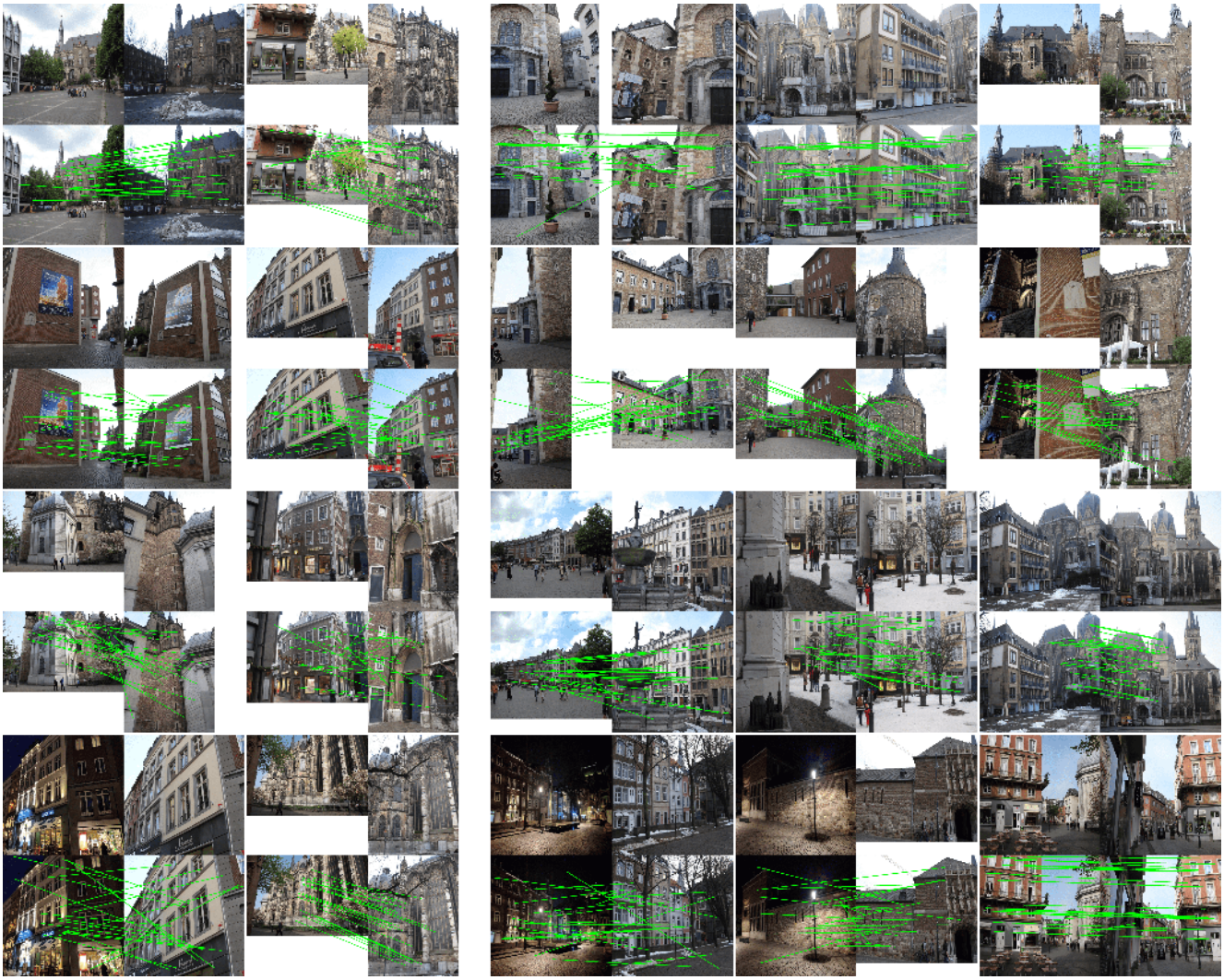


Figure 8: Random sample matches found on the Aachen Day-Night test set. We show the original image pairs top and the pairs with a random subset of located correspondences overlaid below.

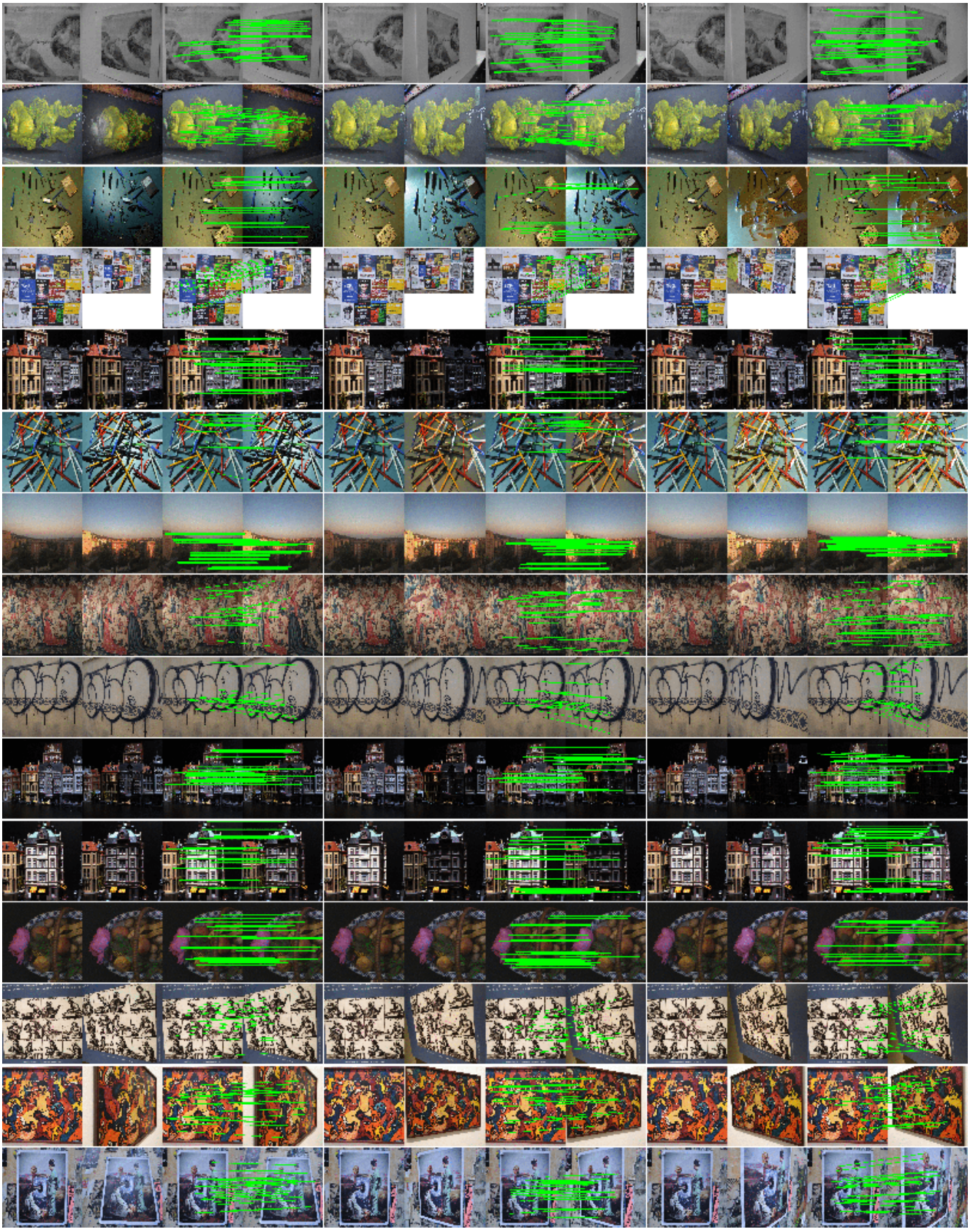
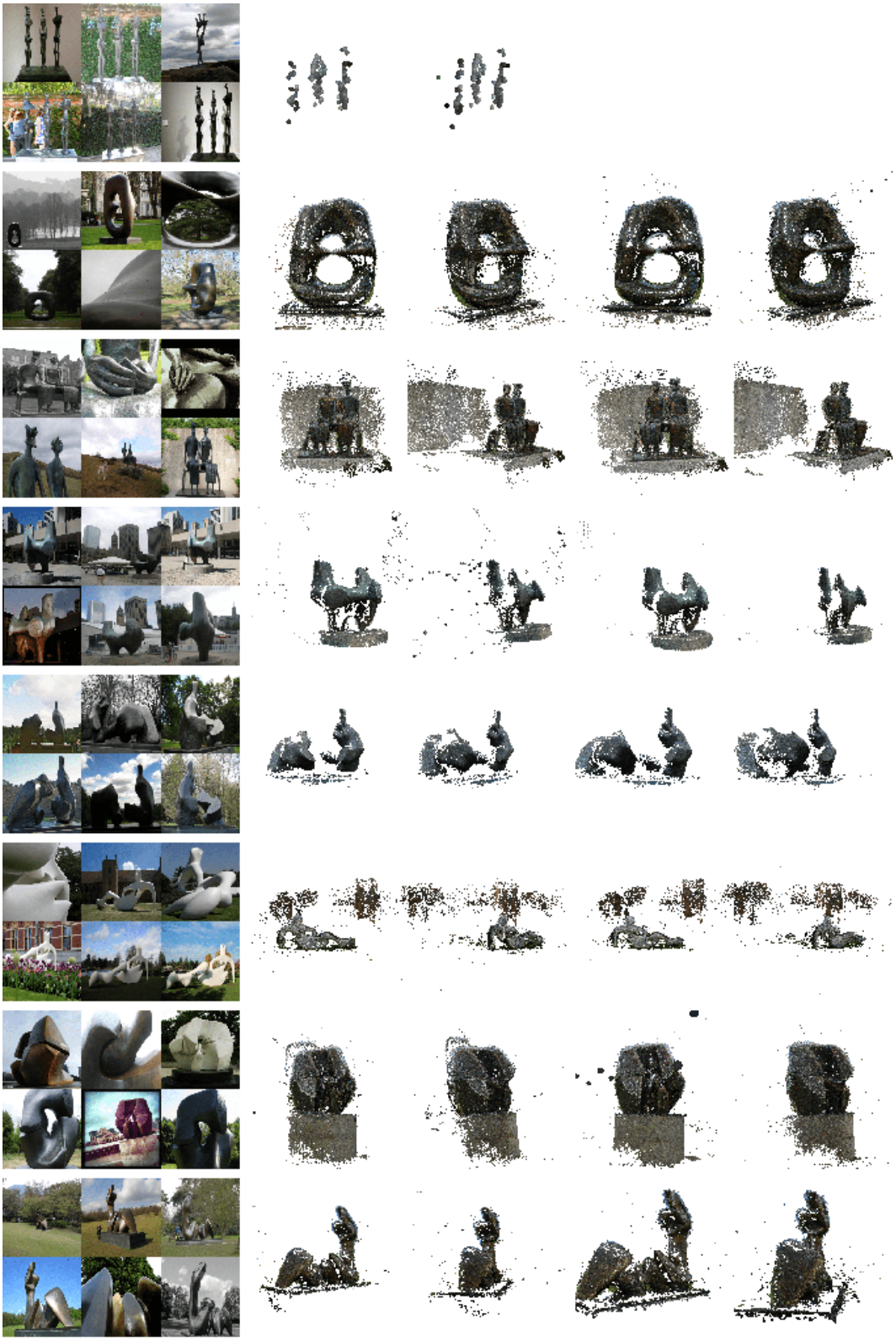


Figure 9: Random sample matches found on the HPatches test set. Rows show progressively harder illumination or viewpoint changes. We first show the original image pairs followed by the pairs with a random subset of located correspondences overlaid.



Input Images | Our 3D Reconstruction | SIFT [10] 3D Reconstruction

Figure 10: Additional SfM results. Randomly selected input images and 3D models reconstructed using our matches and those obtained using the SIFT [10] baseline. This figure demonstrates the variety of the input images and their *scene shift* as well as that both our and [10] are of similar quality for these image sets. However, unlike [10], our model is able to determine that there are 3 statues in the first example.

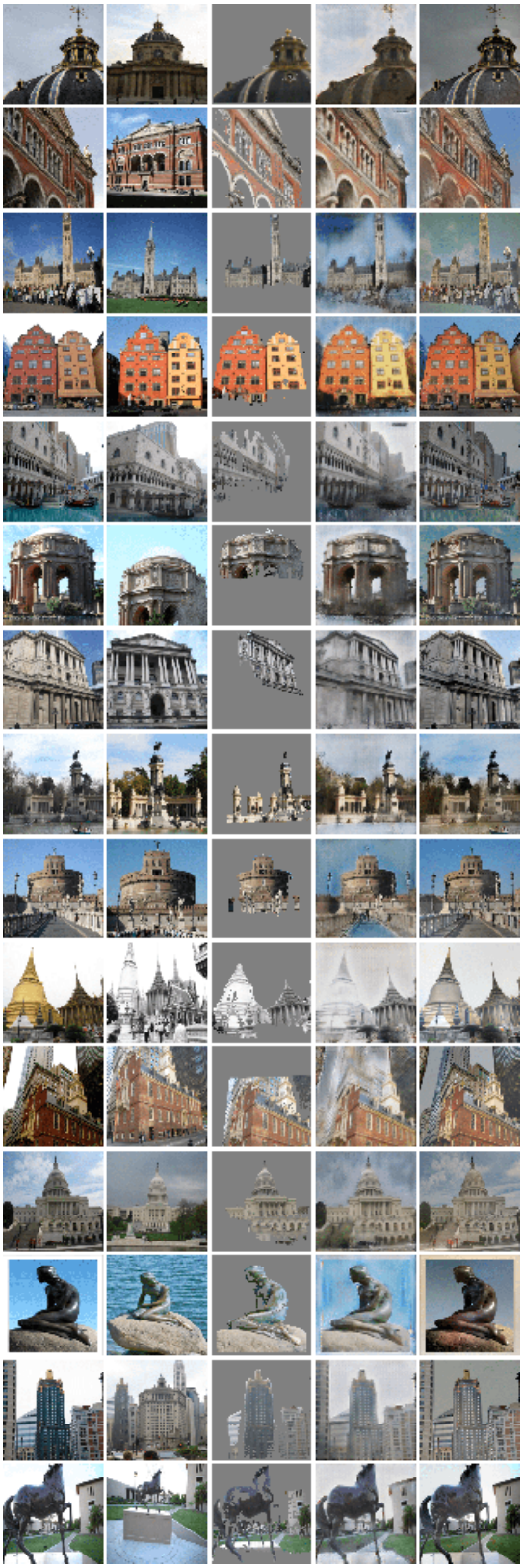


Figure 11: Additional stylization results. These results are random sampled from the test set of MegaDepth.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatch: A benchmark and evaluation of hand-crafted and learned local descriptors. In *Proc. CVPR*, 2017. 5

[2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPR workshops*, 2018. 4

[3] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A trainable cnn for joint description and detection of local features. In *Proc. CVPR*, 2019. 3, 4

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981. 4

[5] D. F. Fouhey, A. Gupta, and A. Zisserman. 3d shape attributes. In *Proc. CVPR*, 2016. 3

[6] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. AISTATS*, 2010. 4

[7] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000. 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proc. CVPR*, 2016. 4, 5

[9] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proc. CVPR*, 2018. 5

[10] David Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 3, 4, 12

[11] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *Proc. CVPR*, 2017. 13

[12] Zixin Luo, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. ContextDesc: Local descriptor augmentation with cross-modality context. In *Proc. CVPR*, 2019. 4

[13] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan. GeoDesc: Learning local descriptors by integrating geometry constraints. In *Proc. ECCV*, 2018. 3

[14] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *NeurIPS*, 2017. 4

[15] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Repeatability is not enough: Learning affine regions via discriminability. In *Proc. ECCV*, 2018. 4

[16] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NeurIPS*, 2013. 4

[17] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proc. CVPR*, 2018. 4

[18] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *Proc. ICCV*, 2017. 4

[19] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 4

[20] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *NeurIPS*, 2018. 4

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 4

[22] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *Proc. ECCV*, 2018. 4

[23] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2D2: Repeatable and reliable detector and descriptor. In *NeurIPS*, 2019. 4

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241. Springer, 2015. 5, 6, 7

[25] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl Kahl, and Tomas Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proc. CVPR*, 2018. 4, 5

[26] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *Proc. BMVC*, 2012. 5

[27] Johannes L Schönberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Proc. CVPR*, 2017. 2

[28] Xi Shen, François Darmon, Alexei A Efros, and Mathieu Aubry. RANSAC-Flow: generic two-stage image alignment. *Proc. ECCV*, 2020. 4

[29] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *Proc. ICML*, 2019. 4

[30] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: the new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 3, 4

[31] Qianqian Wang, Xiaowei Zhou, Bharath Hariharan, and Noah Snavely. Learning feature descriptors using camera pose supervision. In *Proc. ECCV*, 2020. 1, 5, 6

[32] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proc. CVPR*, 2018. 7

[33] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *Proc. ICCV*, 2019. 4

[34] Linguang Zhang and Szymon Rusinkiewicz. Learning to detect features in texture images. In *Proc. CVPR*, 2018. 4