Learning Camera Localization via Dense Scene Matching

Shitao Tang[1] Chengzhou Tang[1] Rui Huang[2] Siyu Zhu[2] Ping Tan[1]
[1]Simon Fraser University [2]Alibaba A.I Labs
{shitao_tang, chengzhou_tang, pingtan}@sfu.ca {rui.hr, siting.zsy}@alibaba-inc.com

Abstract

Camera localization aims to estimate 6 DoF camera poses from RGB images. Traditional methods detect and match interest points between a query image and a pre-built 3D model. Recent learning-based approaches encode scene structures into a specific convolutional neural network (CNN) and thus are able to predict dense coordinates from RGB images. However, most of them require re-training or re-adaption for a new scene and have difficulties in handling large-scale scenes due to limited network capacity. We present a new method for scene agnostic camera localization using dense scene matching (DSM), where a cost volume is constructed between a query image and a scene. The cost volume and the corresponding coordinates are processed by a CNN to predict dense coordinates. Camera poses can then be solved by PnP algorithms. In addition, our method can be extended to temporal domain, which leads to extra performance boost during testing time. Our scene-agnostic approach achieves comparable accuracy as the existing scene-specific approaches, such as KFNet, on the 7scenes and Cambridge benchmark. This approach also remarkably outperforms state-of-the-art scene-agnostic dense coordinate regression network SANet. The Code is available at https://github.com/Tangshitao/Dense-Scene-Matching.

1. Introduction

Camera Localization aims to estimate a 6-DoF camera pose of an image in a known environment. It is an important module in applications such as mobile navigation, simultaneous localization and mapping (SLAM) and augmented reality (AR). Camera localization methods can be broadly categorized as regression-based and structure-based. Earlier methods [23, 21, 22, 47] directly regress the camera poses from images, which are limited by the nature of image retrieval and generally less accurate [41]. In comparison, structure-based methods [3, 43, 4, 36, 53, 39, 46] gradually become the trend and solve the problem in two stages: first, establishing the correspondences between 2D query image pixels and 3D scene points; second, estimating the desired camera pose by PnP [18] combined with different RANSAC [13] algorithms.

According to how they establish the 2D-3D correspondences, the structure-based methods can be further categorized into two classes: 1) sparse feature matching [36, 38, 39, 46]; 2) scene coordinate map regression [3, 43, 4, 53, 24]. The sparse feature matching methods detect and match handcrafted [27] or CNN-based [10, 38] feature points between a query image and scene images, which is able to handle arbitrary scenes. On the other hand, coordinate map regression methods predict dense 3D coordinates at all image pixels from a random forest [44] or a convolutional neural network (CNN) [3, 43]. The estimated dense coordinate maps can be effectively applied to augmented reality and robotics applications such as virtual object insertion or obstacle avoidance. But these methods are often limited to the scene where the random forests or CNN is trained.

In this paper, we focus on the coordinate map regression approach. Recently, instead of encoding specific scene information in network parameters [44, 3, 43], Yang et.al, propose the first dense coordinate regression network SANet for arbitrary scenes [48]. SANet extracts a scene representation from some scene images and corresponding 3D coordinates by 2D-3D matching. In this way, it can be applied to different scenes without re-training or re-adaption. However, due to the irregular nature of a scene, SANet randomly selects coordinates within a region using ball query and leverages PointNet [32] to regress per-pixel 3D coordinates. This operation undermines the pose accuracy and is computationally-heavy because a shared PointNet is required to make prediction on each pixel individually.

In order to address this problem, we present a new scene-agnostic camera localization network exploiting dense scene matching (DSM), which matches each query image pixel with the scene via a cost volume. With end-to-end training, the cost volume explicitly enforces more accurate scene points to have a higher correlation with the input query pixel. Since the scene structure is irregular, which makes the number of query-scene correlations different for each image pixel, we propose a simple yet effective solu-

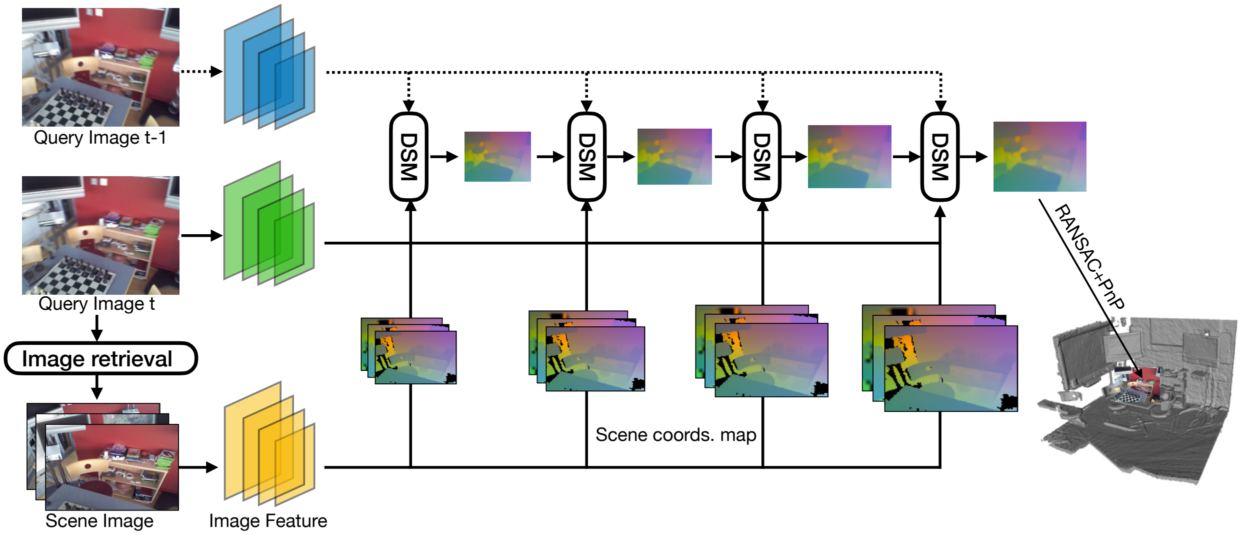


Figure 1: Overview of our framework. Our method predicts dense coordinate maps in a coarse-to-fine manner. The DSM module receives a query image feature map, some scene image feature maps and the corresponding scene coordinates to predict a dense coordinate map for the query image. This predicted scene coordinates are then used to solve camera poses with RANSAC and PnP algorithms.

tion to unify the size of *all* cost volumes: sorting and selecting the best K candidates and feed them to a convolutional neural network for dense coordinate regression. The cost volume can be further fused with temporal correlations between consecutive query images during inference, so that our method can be extended to video localization.

We have evaluated our method on several benchmark datasets including indoor scenes, 7scenes [44] and large-scale outdoor scenes, Cambridge [23]. We have shown DSM achieves state-of-the-art performance among scene-specific methods including DSAC++ in terms of both pose accuracy and coordinate accuracy, and outperforms scene-agnostic methods, e.g. SANet, by a large margin.

2. Related Work

Direct Pose Regression. The prestigious PoseNet [23] and its varients [23, 22, 5, 21] regress the 6-DoF absolute poses directly from RGB images. These networks are trained in a supervised manner on RGB images with known ground truths by a regression loss of pose errors. Intuitively, these methods train a network to memorize the poses of all RGB images in a database. It has been demonstrated in the work [41] that direct pose regression yields results similar to pose approximation via image retrieval and the pose accuracy is usually inferior to the structure-based approaches, which are further classified into the following two categories.

Sparse Feature Matching. Methods [8, 40, 36, 46, 29, 29] based on sparse feature matching build 2D-3D correspondences by interest point detection [27, 10, 12, 2, 17, 30] and local descriptor matching [36, 10, 38, 12, 27, 6]. Then, poses are estimated by PnP combined with RANSAC. To further improve the localization performance, the subsequent learning-based approaches gradually take a coarse-to-fine methodology [36, 46, 37, 30]. The other methods generally focus on improving the capability of local feature detectors [10, 12, 42, 51], descriptors[1, 52, 10, 12, 51] and correspondence matching[38]. A recent work along this direction is SuperGlue [38] and it achieves strong pose accuracy, especially in large-scale outdoor scenes. However, as limited by local feature descriptors, those methods tend to handle scenes with textureless regions or repeated patterns poorly. Instead, by leveraging global contexts, our method show better robustness on those scenes. Additionally, our method can generate dense coordinate maps which are important to various robotics and augmented reality applications.

Dense coordinate regression. Different from sparse 2D-3D correspondences, these methods directly regress the dense 3D scene coordinates of the query image and obtain the final camera pose by dense 2D-3D correspondences [3, 43, 53, 4, 44, 24]. Shotton et al. [44] proposes to regress the scene coordinates using a Random Forest. Along this direction, DSAC [3] and DSAC++ [43] employ convolutional neural networks to predict a dense coordinate map from a single RGB image. KFNet [53] extends such ideology to the tasks of video sequence localization and embeds coordinate regression into Kalman filter within a deep learning framework. It achieves the top performance on

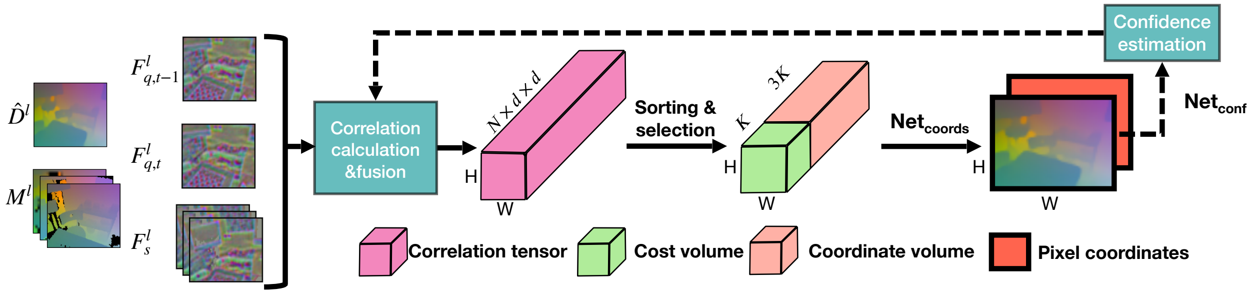


Figure 2: Illustration of Dense Scene Matching (DSM) module. For a specific pyramid level l, DSM takes 1) query image feature maps $\mathbf{F}^l_{q,t}$ at time t and $\mathbf{F}^l_{q,t-1}$ at time $t-1$; 2) scene image feature maps $\mathbf{F}^l_s$ with corresponding scene coordinates; 3) initial coordinate maps $\hat{\mathbf{D}}^l$. Then the DSM module predicts a coordinate map $\mathbf{D}^l$ by cost volume construction and coordinate regression. In the figure, N, H, W is the number of scene images, image heights and image weights respectively. K is the number of scene coordinates selected for regression and d is the window size of candidate scene coordinates.

both single frame and video relocalization tasks. Notably, all of those methods are scene-specific and cannot be generalized to arbitrary novel scenes, which limits their applications in scenarios requiring quick adaption to novel scenes. SANet [48] is the first network proposed to regress coordinates in a scene-agnostic manner. However, it selects feature matches using ball query and uses Point-Net [32, 33] to regress 3D scene coordinates, which largely decreases coordinate accuracy and network efficiency. Our method is also scene agnostic, and we employ cost volumes to evaluate feature matches and compute 3D scene coordinates, which outperforms recent scene-specific and scene-agnostic methods including DSAC [3], DSAC++ [4] and SANet [48].

Cost volume. The proposed method in this paper is inspired by the ideology of cost volume which has been widely adopted in computer vision tasks, e.g. optical flow[11, 45, 19, 34], stereo matching [7, 28, 31] and multi-view stereo [49, 16, 50]. Recent learning-based methods for optical flow or stereo matching extract feature pyramid, build cost volumes and make predictions in a coarse-to-fine manner [45, 7]. Since stereo matching or optical flow construct the cost volumes between image pairs, the number of costs for each pixel is fixed and can be arranged into a regular volume. On the other hand, multi-view stereo (MVS) build a dense 3D regular cost volume between images and the 3D space, with respect to a fixed number of depth or disparity hypothesis planes. However, the 3D dense cost volume used in MVS is infeasible to construct in our problem since it requires to sample a large number of hypothesis points, which makes the cost volume too large to process. Therefore, to build a regular 2D cost volume between a query image and a 3D scene efficiently, we propose a straightforward sorting strategy. The final dense coordinate maps are then obtained from the constructed cost volume. Thanks to the cost volume based formulation, we can easily fuse temporal information to deal with video input.

3. Method

3.1. Overview

The overall framework of our system is illustrated in Fig.1. The pipeline takes a single image or a video sequence as query input. For each query image, we first retrieve N nearest scene images with corresponding coordinate maps via deep image retrieval [15]. Next, we extract a L-level feature pyramid for each query and scene image via the Feature Pyramid Network [25]. In a coarse-to-fine manner, we then design a Dense Scene Matching (DSM) module at each pyramid level to regress the dense coordinate maps of gradually higher resolution and accuracy. Finally, the camera pose is estimated from the finest coordinate map by the standard RANSAC+PnP algorithm.

3.2. Feature and Coordinate Pyramid

Given one query image $\mathbf{Q}_t$ at time t and multiple reference scene images $\{\mathbf{S}_i | i = 1, ..., n\}$, we generate a L-level pyramid of feature maps $\{\mathbf{F}^l | l = 1, ..., L\}$ for each of them by ResNet50-FPN [25]. We denote the query feature maps as $\mathbf{F}^l_q$ and the scene feature maps as $\mathbf{F}^l_s$. The feature vectors in $\mathbf{F}^l_q$ are referred as $\mathbf{f}^l_q$, and those in $\mathbf{F}^l_s$ are $\mathbf{f}^l_s$. The spatial size of feature maps at level l is $H^l \times W^l$.

For each scene image with known 3D coordinates, we also build a L-level coordinate pyramid $\{\mathbf{M}^l | l = 1, ..., L\}$. The spatial size of each coordinate map is the same as that of the feature map $\mathbf{F}^l_s$. In order to deal with scenes at different scales, we transform the 3D scene coordinates to a local coordinate system, where the coordinates are normalized to zero-mean and unit standard deviation at all x, y, z channels.

We estimate the coordinate map in a coarse-to-fine manner. After initializing the coarsest level, the coordinate map $\hat{\mathbf{D}}^l$ at level l is initialized by upsampling from $\mathbf{D}^{l+1}$.

3.3. Dense Scene Matching

The overview of the DSM module is shown in Fig. 2. At a specific level l, the input of DSM module includes: 1) query image feature maps $\mathbf{F}_q^l$; 2) scene image feature maps $\mathbf{F}_s^l$ and corresponding scene coordinate maps M^l; 3) initial coordinate maps $\hat{\mathbf{D}}^l$ upsampled from $\mathbf{D}^{l+1}$. The DSM module predicts the coordinate map $\mathbf{D}^l$ with more details from the initial $\hat{\mathbf{D}}^l$. Specifically, DSM consists of two steps, namely cost volume construction and coordinate regression. It first constructs a cost volume which measures the correlations between 2D query pixels and scene points (with known coordinates). It then regresses a dense coordinate map of the query image from the cost volume.

3.3.1 Cost Volume Construction

This section explains the details of cost volume construction, which involves two processes, namely the scene correlation and temporal correlation. The scene correlation measures similarity between query image pixels and scene points, while the temporal correlation measures the similarity between query image pixels from two neighboring frames in the query video clip. Our network only uses scene correlation in training, and fuses both correlations at testing time.

Scene correlation. The scene correlation is defined as cosine similarity between the features of query pixels and the ones of 3D scene points. We adopt a coarse-to-fine strategy in order to avoid the computation between all 2D-3D pairs. For the coarsest level, we compute the correlation between each query pixel and every 3D scene point since its initial depth is unknown. For the other levels, as shown in Fig.3, for an pixel $\mathbf{q}$ in the query feature map $\mathbf{F}_q^l$, we obtain its 3D coordinate from the initial coordinate map $\hat{\mathbf{D}}^l$. After that, we project the 3D coordinates to each scene image. Suppose the projected position is $\mathbf{p}$, we consider a $d \times d$ search window centered at $\mathbf{p}$ and compute the cosine similarity between the feature vector at $\mathbf{q}$ and those feature vectors for the pixels within the search window. In this way, we obtain a correlation vector of size $d \times d$ at the query pixel at $\mathbf{q}$. We initialize the correlation value as 0 if the corresponding position is out of the image. Given N reference scene images, we obtain a $N \times d \times d$ scene correlations per pixel, which aggregate to a $H^l \times W^l \times (N \times d \times d)$ tensor, named correlation tensor.

Temporal correlation. If the query input is a video sequence, we can leverage the result at the previous frame and the correlation between neighboring video frames to enhance the result. Basically, if the camera pose is known, we can project a scene point $\mathbf{p}$ into the query video frame $\mathbf{Q}_{t-1}$ at $\mathbf{q}'$. Then the correlation between $\mathbf{p}$ and the query pixel $\mathbf{q}$ in video frame $\mathbf{Q}_t$ can be evaluated by the correlation between the two query pixels $\mathbf{q}'$ and $\mathbf{q}$.

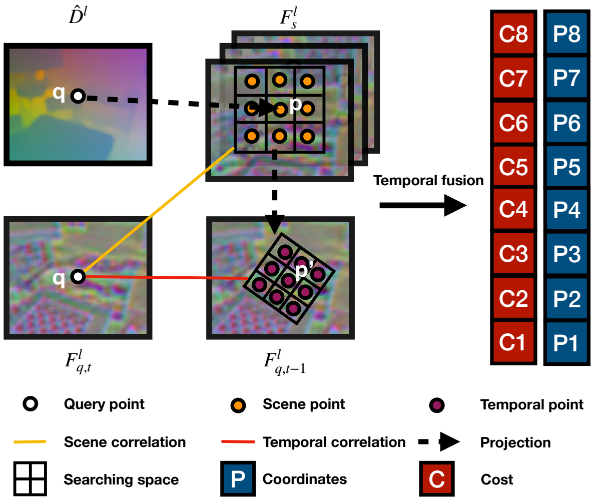


Figure 3: Demonstration of correlation fusion process. For a specific pixel $\mathbf{q}$ in $\mathbf{F}_{q,t}^l$, we obtain its scene correlation by projecting its corresponding 3D coordinate predicted in $\hat{\mathbf{D}}^l$ to a searching space of a $d \times d$ window in retrieved N scene feature maps $\mathbf{F}_s^l$. Temporal correlation is obtained by projecting the scene coordinates within the searching space in $\mathbf{M}^l$ to $\mathbf{F}_{q,t-1}^l$. Finally, a $N \times d \times d$ correlation tensor is formed for each query pixel.

Specifically, we project all scene points to the query image $\mathbf{Q}_{t-1}$ according to the camera pose at $t-1$. Subsequently, we can compute the correlation between the feature vector at a query pixel in $\mathbf{Q}_t$ and the feature vectors of these projected pixels in $\mathbf{Q}_{t-1}$. In this way, for each query pixel, we also obtain a correlation vector of size $N \times d \times d$ by temporal correlation.

Correlation fusion. The final correlation score between a query pixel and a scene point is then computed from the scene correlation and temporal correlation by the equation, $Corr = \alpha Corr_s + (1-\alpha) Corr_t$, where $Corr_s$ stands for scene correlation and $Corr_t$ is temporal correlation. The parameter α balances $Corr_s$ and $Corr_t$. The hyper-parameter α is derived from the confidence score by $\alpha = min(s + 0.4, 1)$, s is the confidence score, which will be introduced in Sec. 3.3.3. Note that the fusion is applied to each of the N reference scene images. At the end of this fusion, we obtain a fused correlation tensor of size $H^l \times W^l \times (N \times d \times d)$.

Cost volume. We construct a cost volume by sorting the correlation values and selecting the top K ($K = 16$ in our implementation) scene coordinates. Although the sorting operation is not differentiable, the gradients can still be passed by the correlation values in the backward propagation during training. Intuitively, a higher correlation score means more accurate match between query pixels and scene points.

After sorting, we obtain a cost volume of size $H^l \times W^l \times$

K. We further concatenate this cost volume with the 3D scene coordinates of corresponding scene points to form a $H \times W \times 4K$ (1 for correlation and 3 for scene coordinates) cost-coordinate volume. This cost-coordinate volume is then processed by CNN to produce a dense coordinate map.

3.3.2 Coordinate Regression

We design a network namely Net_{coords} to estimate the final scene coordinate map by taking the input of cost volume, coordinate volume and image features. The cost-coordinate volume are first fed into a network consisting of 1×1 convolutional layers and produce a coordinate feature map. This coordinate feature map are concatenated with image feature map and fed into another network consisting of 3×3 convolutional layers to predict the final coordiante map. The detailed architecture is illustrated in the supplementary material.

3.3.3 Confidence Estimation

In order to fuse the temporal correlations and scene correlations, we estimate a confidence value s as the weighting parameter, as discussed in Sec. 3.3.1. We predict a certainty score for each pixel, which measures how accurate the coordinate prediction is. As illustrated in Fig.2, after predicting the coordinate map from only scene correlation, we concatenate it with the corresponding 2D pixel coordinates and feed to Net_{conf} which outputs certainty scores. The architecture of Net_{conf} is explained in supplementary material. We treat the certainty score estimation as a ranking problem. Coordinates with higher certainty scores are supposed to have smaller reprojection errors. This relation can be measured by the average precision metric. Therefore, we label each pixel as correct if its reprojection error of the estimated coordinate is smaller than a threshold (1 pixel in implementation) or incorrect otherwise and use average precision loss [35] to optimize Net_{conf}. The final confidence s is the average certainty score over all pixels, and then the fusion score α can be computed. After fusing scene correlation and temporal correlation, we still use Net_{coords} to predict the final coordinate map.

3.4. Training loss.

The total loss is the summation of the regression loss, $L_{regress}$, for coordinate regression and the average precision loss [35], L_{AP}, for training certainty scores. For coordinate regression, we use L_1 distance errors between predicted coordinates and ground truth coordinates as training loss.

$$L_{regress} = ||Y_{coords} - \overline{Y}_{coords}||$$

$$L = L_{AP} + \frac{1}{n}\sum_{i=0}^{n}(L_{regress})$$

Where Y_{coords} is the absolute coordinate predicted from Net_{coords}, $\overline{Y}$ stands for the ground truth and n is the number of query pixels.

4. Experiments

4.1. Experiment Settings

Dataset. We evaluate our method on both the indoor dataset 7scenes [14] and the outdoor dataset Cambridge Landmarks [23]. For 7scenes, it contains 7 different scenes with raw RGB-D video sequences captured by a handheld Kinect RGB-D camera. It also provides camera poses and a dense 3D model for each scene generated by KinectFusion [20]. Cambridge Landmarks dataset contains 6 different outdoor scenes with RGB video frames labelled with full 6-DOF camera poses. We train our network using ScanNet dataset [9], which is a RGB-D video dataset consisting of 2.5M views in 1513 scenes annotated with 3D camera poses and dense depth maps.

Data processing. All the images of the 7scenes [14], Cambridge Landmarks [23] and ScanNet [9] datasets are downsized to 384 × 512. To form the training data, we first randomly sample about 160k images from ScanNet dataset as query images. For each query image, we retrieve 5 and 10 corresponding scene images in the same video sequence for training and testing respectively by the learning-based image retrieval approach [15]. In order to encourage query-scene image pairs with different viewing angles, we only keep the scene images of the same video sequence that are at least 50 frames away from a given query image. We follow the multi-view stereo reconstruction method adopted in the DSAC [3] to obtain dense 3D coordinates of the Cambridge Landmarks.

Training. We only use Scannet as training data for the inference on 7scenes dataset. As for a specific scene of the outdoor dataset Cambridge Landmarks, we fine-tune our pretrained model with the other 5 scenes. ResNet50-FPN [25] is regarded as our backbone network for all the following experiments. Our model is trained with an AdamW optimizer [26], whose base learning rate is 0.0005, and a batch size of 16 in a single RTX TITAN GPU for 50000 iterations.

4.2. Localization Accuracy

In this section, we mainly compare our approach with two classes of methods, namely sparse feature matching [39, 46, 36] and dense coordinate regression methods [3, 43, 53, 48]. We measure localization accuracy in

	7scenes (Indoor)	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
Sparse	Active Search	1.96°, 0.04m	1.53°, 0.03m	1.45°, 0.02m	3.61°, 0.09m	3.10°, 0.08m	3.37°, 0.07m	2.22°, **0.03m**
	InLoc	1.05°, 0.03m	1.07°, 0.03m	0.16°, 0.02m	1.05°, 0.03m	1.55°, 0.05m	1.31°, 0.04m	2.47°, 0.09m
	HLoc	**0.79**°, **0.02m**	**0.87**°, **0.02m**	**0.92**°, **0.02m**	**0.91**°, **0.03m**	**1.12**°, **0.05m**	**1.25**°, **0.04m**	**1.62**°, 0.06m
Dense	DSAC(*)	0.7°, 0.02m	1.0°, 0.03m	1.3°, 0.02m	1.0°, 0.03m	1.3°, 0.05m	1.5°, 0.0.5m	49.4°, 1.9m
	DSAC++(*)	**0.5**°, **0.02m**	0.9°, **0.02m**	**0.8**°, **0.01m**	0.7°, **0.03m**	1.1°, **0.04m**	1.1°, 0.04m	2.6°, 0.09m
	KFNet(*)	0.65°, **0.02m**	0.9°, **0.02m**	0.82°, **0.01m**	**0.69**°, **0.03m**	**1.02**°, **0.04m**	1.16°, 0.04m	**0.94**°, **0.03m**
	SANet	0.88°, 0.03m	1.10°, 0.03m	1.48°, 0.02m	1.03°, 0.03m	1.32°, 0.05m	1.4°, 0.04m	4.59°, 0.16m
	Ours (Single)	0.71°, 0.02m	0.85°, 0.02m	0.85°, 0.01m	0.84°, 0.03m	1.16°, 0.04m	1.17°,0.04m	1.33°, 0.05m
	Ours (Video)	0.68°, **0.02m**	**0.80**°, **0.02m**	**0.80**°, **0.01m**	0.78°, **0.03m**	1.11°, **0.04m**	**1.11**°,**0.03m**	1.16°, 0.04m

	Cambridge (outdoor)	Great Court	King's College	Old Hospital	Shop Facade	St. Mary's Church	Street
Sparse	Active Search	0.6°, 1.20m	0.6°, 0.42m	1.0°, 0.44m	0.4°, 0.12m	0.5°, 0.19m	**0.8**°, 0.85m
	InLoc	0.62°, 1.20m	0.82°, 0.46m	0.96°, 0.48m	0.50°, 0.11m	0.63°, 0.18m	2.16°, 0.75m
	HLoc	**0.21**°, **0.38m**	**0.31**°, **0.17m**	**0.39**°, **0.23m**	**0.37**°, **0.07m**	**0.29**°, **0.10m**	1.32°, **0.62m**
Dense	DSAC(*)	1.5°, 2.8m	0.5°, 0.30m	0.6°, 0.33m	0.4°, 0.09m	1.6°, 0.55m	
	DSAC++(*)	0.2°, **0.40m**	0.3°, 0.18m	0.3°,0.2m	**0.3**°, 0.06m	0.4°, 0.13m	
	KFNet(*)	0.21°, 0.42m	**0.27**°, **0.16m**	**0.28**°, **0.18m**	0.35°, **0.05m**	0.35°, 0.12m	
	SANet	1.95°, 3.28m	0.42°, 0.32m	0.53°, 0.32m	0.47°, 0.10m	0.57°, 0.16m	12.64°, 8.74m
	Ours (Single)	0.23°, 0.44m	0.36°, 0.19m	0.39°, 0.24m	0.38°, 0.07m	0.35°, 0.12m	1.71°, 0.68m
	Ours (Video)	**0.19**°, 0.43m	0.35°, 0.19m	0.38°, 0.23m	**0.30**°, 0.06m	**0.34**°, **0.11m**	**1.53**°, **0.61m**

Table 1: Performance comparison in terms of rotation errors (°) and translation errors (m). (*) indicates scene-specific methods.

		Single frame localization			Video localization		
	Acc. thresh	Median	Mean	Acc.	Median	Mean	Acc.
Chess	5°, 0.05	0.713°, 0.021	0.824°, 0.024	94.5	0.684°, 0.020	0.795°, 0.023	96.1 (+1.6)
Fire	5°, 0.05	0.856°, 0.021	1.025°, 0.027	93.8	0.802°, 0.020	0.878°, 0.020	94.5 (+0.7)
Heads	5°, 0.05	0.846°, 0.013	1.369°, 0.023	96.4	0.802°, 0.013	0.957°, 0.016	99.5 (+3.1)
Office	5°, 0.05	0.843°, 0.028	0.983°, 0.037	82.3	0.782°, 0.026	0.937°, 0.034	84.2 (+1.9)
Pumpkin	5°, 0.05	1.164°, 0.043	2.224°, 0.112	57.0	1.113°, 0.043	1.823°, 0.083	57.2 +(0.2)
Kitchen	5°, 0.05	1.165°, 0.038	3.145°, 0.082	68.7	1.115°, 0.034	1.358°, 0.044	69.2 (+0.5)
Stairs	5°, 0.05	1.356°, 0.045	3.424°, 0.197	53.9	1.157°, 0.037	1.553°, 0.069	69.9 (+16.0)
Great Court	5°, 1.0	0.209°, 0.444	6.043°, 5.624	68.5	0.193°,0.428	4.023°, 4.017	76.7 (+8.2)
King's College	5°, 0.5	0.358°, 0.194	0.574°, 0.424	82.9	0.353°, 0.188	0.522°, 0.367	84.6 (+1.7)
Old Hospital	5°, 0.3	0.388°, 0.243	0.387°, 0.502	41.2	0.382°, 0.228	0.372°, 0.498	43.7 (+2.5)
Shop Facade	5°, 0.2	0.375°, 0.074	0.623°, 0.131	84.2	0.303°, 0.061	0.574°, 0.112	86.4 (+2.2)
St. Mary's Church	5°, 0.3	0.353°, 0.118	1.146°, 0.374	91.4	0.342°, 0.111	0.845°, 0.264	93.7 (+2.3)
Street	5°, 2.0	1.711°, 0.684	22.551°, 27.111	62.2	1.523°, 0.609	20.756°, 25.862	64.8 (+2.6)

Table 2: Comparison of single frame based localization and video-based localization. For 7scenes, we use common threshold (5°, 0.05m) to calculate accuracy. We can see video-based has significantly lower mean errors and higher accuracy.

terms of median errors in translation and rotation. As shown in Table. 1, the proposed DSM approach achieves state-of-the-art performance among both sparse matching and dense regression methods.

Compared with sparse matching methods, the pose accuracy of our approach is superior to that of Active Search [39] and InLoc [46]. HLoc [36], upgraded with SuperPoint [10] for feature detection and Superglue [38] for feature correspondence matching, is considered and such upgrade brings higher relocalization accuracy compared with the original HLoc approach [36] as reported in the work [38]. We can see that DSM outperforms HLoc in 7scenes, and it is slightly inferior to HLoc in outdoor Cambridge Landmarks dataset which contains much more salient texture for sparse feature matching.

When comparing with scene-specific dense coordinate regression methods, the proposed scene-agnostic approach DSM outperforms DSAC [3] by a large margin and obtains slightly superior performance than DSAC++ [43]. Even for KFNet [53] with the top performance on single frame and video localization tasks, our approach achieves comparable performance. In comparison with the scene-agnostic SANet [48], DSM shows obvious superior performance.

Table.2 shows the detailed comparison of metrics of median errors, mean errors and the pose accuracy falling within certain accuracy threshold (*Acc. thresh*) between single frame localization and video localization methods. As shown, after applying the temporal fusion, the localiza-

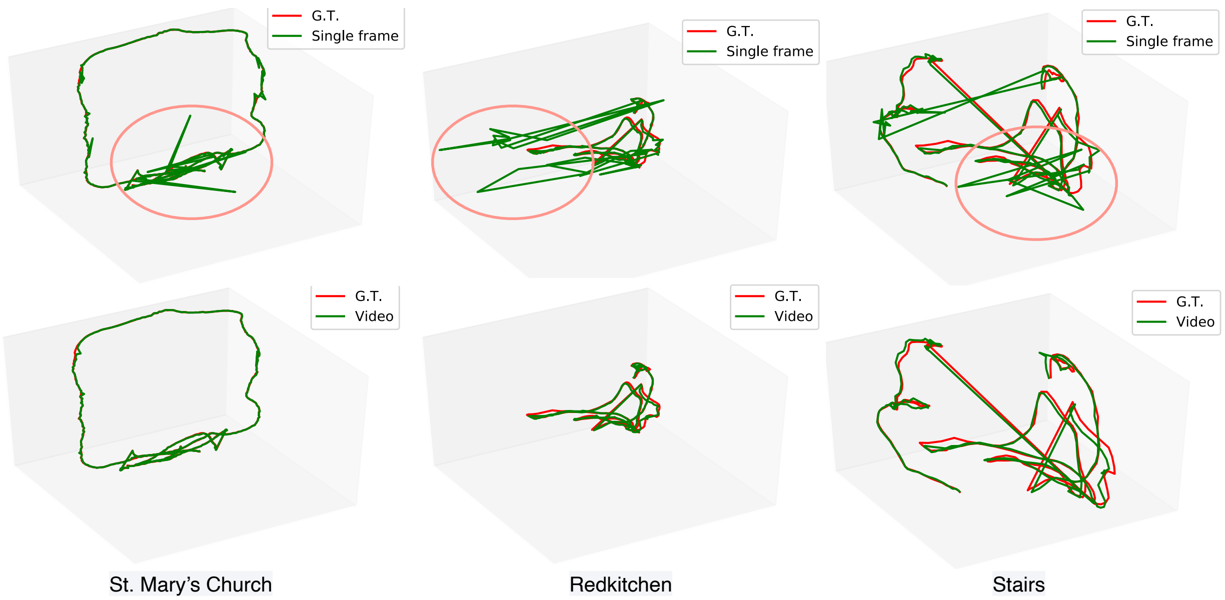


Figure 4: The comparison of camera trajectories between the single frame (first row) and video localization (second row) via the proposed dense scene matching network. The visualized results are respectively *Redkitchen* and *Stairs* in 7-scenes dataset, and *St. Mary's Church* sequence in Cambridge Landmarks dataset. In the first row, the outliers are shown in the red circles.

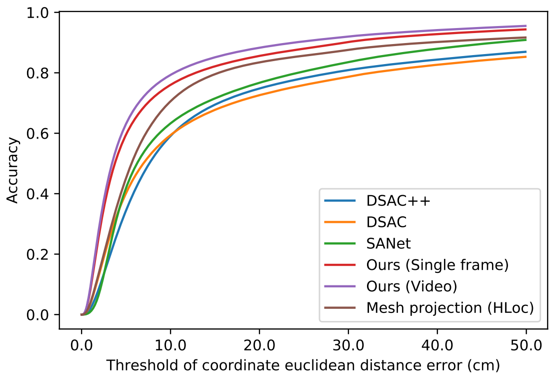


Figure 5: The comparison of cumulative distribution functions of scene coordinate errors between different localization approaches.

tion accuracy notably increases indeed. In addition, Fig. 4 shows the trajectories of *Redkitchen* and *Stair* sequence of 7-scenes dataset and *St. Mary's Church* sequence of Cambridge dataset. We can see that the trajectories of our single frame localization contains some outliers while our video localization is able to remove most of them.

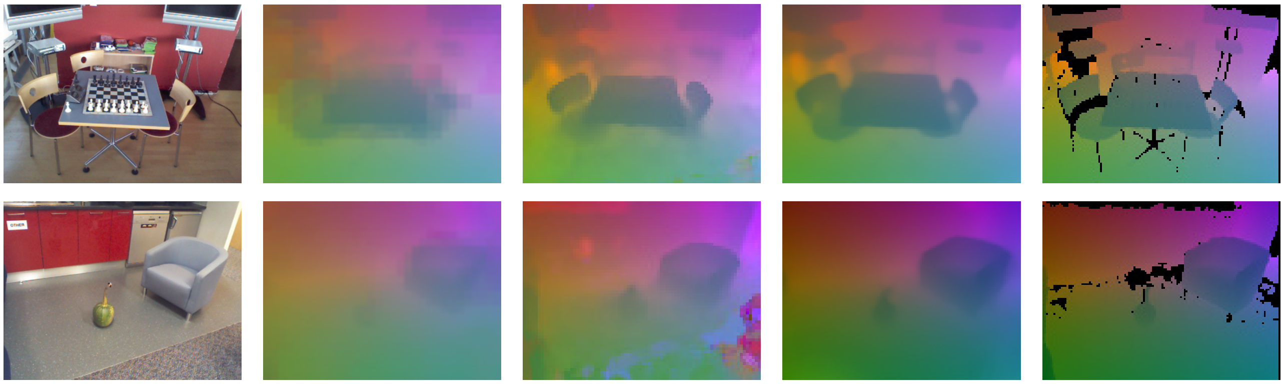
4.3. Scene coordinate accuracy

In terms of scene coordinate accuracy, we compare our method with SANet [48], DSAC [3], DSAC++ [43] and HLoc [36] on the whole 7scenes dataset. Since HLoc cannot directly output a dense coordinate map, we first get dense depth maps by projecting reconstructed mesh to its predicted poses and compute coordinates by back-projection. We calculate the coordinate accuracy under different euclidean distance error threshold and plot cumulative distribution function in Fig.5. We can see that the accuracy of coordinate maps from our network outperforms SANet, DSAC and DSAC++ by a large margin. More specifically, we surpass SANet by 16% and DSAC++ by 20% when the threshold is set to 10 cm. The projected coordinates of HLoc is more accurate than DSAC, DSAC++ and SANet, but is under-performed by DSM. In addition, our temporal-based coordinate map regression boosts accuracy compared with our single frame prediction.

	Run time	GPU memory usage
SANet	0.33s	5GB
Ours	0.21s	2.7GB

Table 3: Efficiency comparison of SANet and DSM.

We also visualize coordinate map in Fig.6 for DSM, SANet and DSAC++. In general, the coordinate map produced by DSM has higher quality and preserves more details than SANet and DSAC++. SANet randomly sample coordinates from search space with ball query, and the best match may be dropped due to this operation. As a result, its coordinate maps contain a large number of artifacts.



(a) Query (b) SANet (c) DSAC++ (d) DSM (e) G.T.

Figure 6: Coordinate map visualization for SANet, DSAC++ and DSM.

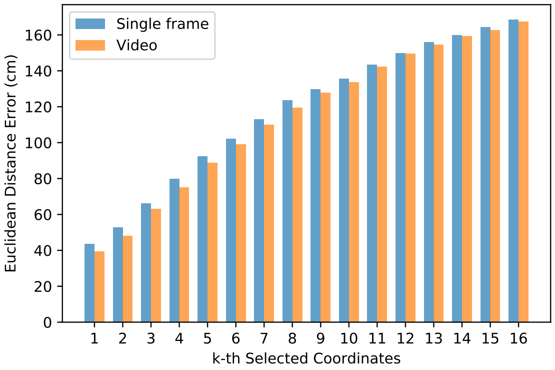


Figure 7: The average scene coordinate errors of the k-th selected coordinates with respect to ground truth.

DSAC++ is able to produce coordinate map with more details, but artifacts exist in some regions as well.

4.4. Efficiency

Table.3 shows the running time and GPU memory usage to localize a single query frame with 5 scene images. We list the statistics of SANet since it is the only localization pipeline that predict dense coordinate map for an arbitrary novel scenes. Here, the image retrieval time is not included. Compared with SANet, Our network reduces the time consumption by 33% and memory consumption by 46%. The efficiency can be further improved by adapting light-weight backbones.

4.5. Analysis of correlation

Our proposed approach assumes that high query-scene correlations lead to more accurate corresponding scene coordinates for query pixels. To verify this argument, we evaluate the relationship between the correlation and scene coordinate errors with respect to ground truth. For each pixel in the query image, we select the top K scene coordinate candidates in 5^{th} level of coordinate pyramid. Then for each ranking index k, we take the average of the euclidean distance error between selected scene coordinates and ground truths over all query pixels. Finally, We define the k^{th} average scene coordinate error as $e_k = \frac{1}{n}\sum_{i=1}^{n}\sqrt{||Y_k^i - \overline{Y}||_2}$, where n is the number of pixels in a query image, for the i^{th} query pixel, Y_k^i is the k^{th} corresponding scene coordinate and $\overline{Y}$ is the ground truth. We summarize the statistics in 7scenes dataset and plot e_k in Fig.7. It can be seen that the scene coordinate error gradually becomes larger when correlation becomes smaller. In other words, high correlation stands for more accurate scene coordinate selection for a specific query pixel. In addition, we also include the evaluation for temporal-based model, which obtains consistently lower euclidean distance errors than single frame model, indicating that correlation fusion further improves the accuracy of selected scene coordinates.

5. Conclusion

In this paper, we present dense scene matching (DSM) for visual localization. DSM is able to estimate dense coordinate maps for arbitrary novel scenes. First, DSM builds a cost volume between a query image and a scene by sorting and selecting the top K highest correlations per pixel. Then, the cost volume with the corresponding coordinates are feed into CNN for dense coordinate regression and a temporal fusion module is introduced to further improve the accuracy of the dense coordinate map. Finally, the camera poses are then estimated by PnP together with RANSAC algorithms. We demonstrated the effectiveness of DSM on both indoor and outdoor datasets. This scene-agnostic method yields comparable accuracy among all scene-specific methods and outperforms scene-agnostic methods in terms of both localization and coordinate accuracy.

References

[1] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Bmvc*, volume 1, page 3, 2016. 2

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. 2

[3] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017. 1, 2, 3, 5, 6, 7

[4] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7525–7534, 2019. 1, 2, 3

[5] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2018. 2

[6] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010. 2

[7] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018. 3

[8] Wentao Cheng, Weisi Lin, Kan Chen, and Xinfeng Zhang. Cascaded parallel filtering for memory-efficient image-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1032–1041, 2019. 2

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 5

[10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018. 1, 2, 6

[11] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 3

[12] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *arXiv preprint arXiv:1905.03561*, 2019. 2

[13] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1

[14] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time rgb-d camera relocalization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179. IEEE, 2013. 5

[15] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European conference on computer vision*, pages 241–257. Springer, 2016. 3, 5

[16] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015. 3

[17] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988. 2

[18] Joel A Hesch and Stergios I Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390. IEEE, 2011. 1

[19] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 3

[20] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 5

[21] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016. 1, 2

[22] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5974–5983, 2017. 1, 2

[23] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 1, 2, 5

[24] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11983–11992, 2020. 1, 2

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3, 5

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1, 2

[28] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 3

[29] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-dof localization on mobile devices. In *European conference on computer vision*, pages 268–283. Springer, 2014. 2

[30] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004. 2

[31] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 887–895, 2017. 3

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 3

[33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 3

[34] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4161–4170, 2017. 3

[35] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5107–5116, 2019. 5

[36] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. 1, 2, 5, 6, 7

[37] Paul-Edouard Sarlin, Frédéric Debraine, Marcin Dymczyk, Roland Siegwart, and Cesar Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. *arXiv preprint arXiv:1809.01019*, 2018. 2

[38] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020. 1, 2, 6

[39] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In *European conference on computer vision*, pages 752–765. Springer, 2012. 1, 5, 6

[40] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1744–1756, 2016. 2

[41] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3302–3312, 2019. 1, 2

[42] Nikolay Savinov, Akihito Seki, Lubor Ladicky, Torsten Sattler, and Marc Pollefeys. Quad-networks: unsupervised learning to rank for interest point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1822–1830, 2017. 2

[43] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *ICML*, volume 2, page 6. Citeseer, 2000. 1, 2, 5, 6, 7

[44] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013. 1, 2

[45] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018. 3

[46] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018. 1, 2, 5, 6

[47] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 627–637, 2017. 1

[48] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. Sanet: Scene agnostic network for camera localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 42–51, 2019. 1, 3, 5, 6, 7

[49] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018. 3

[50] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *The journal of machine learning research*, 17(1):2287–2318, 2016. 3

[51] Linguang Zhang and Szymon Rusinkiewicz. Learning to detect features in texture images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6325–6333, 2018. 2

[52] Hao Zhou, Torsten Sattler, and David W Jacobs. Evaluating local features for day-night matching. In *European Confer-*

ence on Computer Vision, pages 724–736. Springer, 2016. 2

[53] Lei Zhou, Zixin Luo, Tianwei Shen, Jiahui Zhang, Mingmin Zhen, Yao Yao, Tian Fang, and Long Quan. Kfnet: Learning temporal camera relocalization using kalman filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4919–4928, 2020. 1, 2, 5, 6