Learning Camera Localization via Dense Scene Matching

by

Shitao Tang

B.Sc., University of Nottingham, Ningbo, China

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2021

Declaration of Committee

Name:	**Shitao Tang**
Degree:	**Master of Science**
Thesis title:	**Learning Camera Localization via Dense Scene Matching**
Committee:	**Chair:** Jiangchuan Liu Professor, Computing Science
	Ping Tan Supervisor Associate Professor, Computing Science
	Yasutaka Furukawa Committee Member Associate Professor, Computing Science
	Mo Chen Examiner Assistant Professor, Computing Science

Abstract

This thesis presents a method for camera localization. Given a set of reference images with known camera poses, camera localization aims to estimate the 6 DoF camera pose for an arbitrary query image captured in the same environment. It might also be generalized to recover the 6 DoF pose of each video frame of an input query video. Traditional methods detect and match interest points between the query image and a pre-built 3D model, and then solve camera poses accordingly by the PnP algorithm combined with RANSAC. The recent development of deep learning has motivated end-to-end approaches for camera localization. Those methods encode scene structures into the parameters of a specific convolutional neural network (CNN) and thus are able to predict a dense coordinate map for a query image whose pixels record 3D scene coordinates. This dense coordinate map can be used to estimate camera poses in the same way as traditional methods. However, most of these learning-based methods require re-training or re-adaption for a new scene and have difficulties in handling large-scale scenes due to limited network capacity. In this thesis, We present a new method for scene agnostic camera localization which can be applied to a novel scene without retraining. This scene agnostic localization is achieved with our dense scene matching (DSM) technique, where a cost volume is constructed between a query image and a scene. The cost volume is fed to a CNN to predict the dense coordinate map to compute the 6 DoF camera pose. In addition, our method can be directly applied to deal with query videoclips, which leads to extra performance boost during testing time by exploring temporal constraint between neighboring frames. Our method achieves state-of-the-art performance over several benchmarks.

Keywords: Camera localization; deep learning; 3D vision

Acknowledgements

I would like to thank my supervisor Dr. Ping Tan, for his great guidance. I'm also deeply grateful to Dr. Chengzhou Tang, for his wonderful ideas and supports. In the end, I want to thank to Dr. Rui Huang and Dr. Siyu Zhu for their advices.

Table of Contents

Declaration of Committee ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables vii

List of Figures viii

1 Introduction 1

2 Preliminary 4

2.1 Camera Localization Preliminary 4

2.2 Related Work 6

2.2.1 Regression-based localization 6

2.2.2 Structure-based localization 7

2.2.3 Video Localization 9

2.2.4 Cost Volume 9

3 Methods 11

3.1 Overview 11

3.2 Feature and Coordinate Pyramid 11

3.3 Dense Scene Matching 12

3.3.1 Cost Volume Construction 12

3.3.2 Coordinate Regression 15

3.3.3 Confidence Estimation 15

3.4 Archtecture of Net_{coords} and Net_{conf} 15

3.5 Training loss. 15

4 Experiments 18

4.1 Experiment Settings 18

4.2 Localization Accuracy 18
4.3 Scene coordinate accuracy 23
4.4 Efficiency 23
4.5 Ablation Study 23

5 Conclusion **28**
5.1 Discussions 28
5.2 Conclusion 29

Bibliography **30**

List of Tables

Table 4.1 Performance comparison in terms of rotation errors (°) and translation errors (m). (*) indicates scene-specific methods. 19

Table 4.2 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. 19

Table 4.3 Efficiency comparison of SANet and DSM. 23

Table 4.4 Pose accuracy with/without top K correlation sorting. The estimated pose accuracy improves by correlation sorting consistently on all sequences. . . . 25

Table 4.5 Pose accuracy with respect to the number of scene images. The network is trained and tested with the corresponding number of scene images except the one with 10 scene images. The notation ($\star$) means we train the network with 5 scene images instead of 10 scene images. 25

Table 4.6 Pose accuracy with respect to different image resolutions. In our implementation, We resize all images to resolution of 384×512 for better efficiency and performance. 26

Table 4.7 The statistical comparison of median rotation and translation errors of Cambridge landmarks with or without fine-tuning. 26

Table 4.8 Localization accuracy under different 26

List of Figures

Figure 1.1 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. These predicted scene coordinates are then used to solve camera poses with RANSAC and PnP algorithms. 3

Figure 2.1 Pinhole camera model. 5

Figure 2.2 Pipeline of common structure-based localization 7

Figure 2.3 Correspondence visualization. 8

Figure 2.4 Visualization of coordinate maps. 9

Figure 3.1 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. 12

Figure 3.2 Demonstration of correlation fusion process. For a specific pixel $\mathbf{q}$ in $\mathbf{F}^l_{q,t}$, 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 $\mathbf{F}^l_s$. Temporal correlation is obtained by projecting the scene coordinates within the searching space in $\mathbf{M}^l$ to $\mathbf{F}^l_{q,t-1}$. Finally, a $N \times d \times d$ correlation tensor is formed for each query pixel. 14

Figure 3.3 Archtecture of Net_{conf} and Net_{coords}. We use residual block for Net_{conf} and dense block for Net_{coords} 16

Figure 4.1 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. 20
Figure 4.2 The comparison of cumulative distribution functions of scene coordinate errors between different localization approaches. 21
Figure 4.3 Coordinate map visualization for SANet, DSAC++ and DSM. 22
Figure 4.4 The average scene coordinate errors of the k-th selected coordinates with respect to ground truth. 24
Figure 4.5 Examples fail in single-frame localization, but succeed in video localization. From the top left, top right, bottom left to the bottom right samples, the error of pose estimation reduces from single-frame localization (30.2°, 0.53m) to video localization (2.2°, 0.06m), (10.4°, 0.34m) to (2.2°, 0.04m), (19.4°, 0.05m) to (0.5°, 0.02m), (15.6°, 0.53m) to (1.4°, 0.07m) respectively. 27

Chapter 1

Introduction

Humans are capable of inferring location and viewpoint from visual information. Given an image of the Statue of Liberty, it is easy to recognize that the picture is taken in New York and predict what viewpoint the picture is taken from. In recent years, localization has become increasingly popular in the research community due to the increasing demand for applications. In order to address the problem, many solutions have been proposed with different sensors and algorithms. Common localization systems can use a variety of inputs, such as WiFi, beacon, or inertial sensors. The most used system is Global Positioning System (GPS), a satellite-based radio navigation system. It can position a GPS receiver anywhere on the Earth. However, GPS-based localization systems alone cannot provide the level of accuracy that the aforementioned applications demand. On the other hand, the supply of large geo-localized image databases and the proliferation of embedded visual acquisition systems, such as smartphone cameras, in recent years make the research community focus on visual localization problems.

Compared to the GPS localization system that can only obtain positions of GPS receivers with large errors, visual localization is able to get the orientations of camera sensors and reach much higher localization accuracy. Thus, it can support a wide range of applications that GPS cannot, such as SLAM, robot navigation, argument reality, etc. In a visual localization system, database images are collected beforehand with RGB or RGBD cameras and the 6 degree-of-freedom camera poses, describing orientations and translations, of those images are obtained using 3D reconstruction algorithms [59, 30] under a world coordinate system. Given a query image, this system aims to recover the camera poses of it with respect to those database images.

Traditional methods adopt the Structure-from-Motion (SfM) [59] algorithm to reconstruct a 3D scene from a set of overlapping 2D database images, which depicts the same scene from different viewpoints. The reconstructed 3D model consists of 3D points with image features and the camera poses of the source imagery. Then, the image features of 3D points are directly matched against those of query images. After obtaining 2D-3D correspondences, the query image poses can then be solved by the perspective-n-point (PnP) [26] algorithms.

With the development of deep learning, learning-based camera localization has drawn attention in the computer vision community. Recent camera localization methods can be broadly catego-

rized as regression-based and structure-based. Earlier methods [33, 31, 32, 64] directly regress the camera poses from images through a convolutional neural network. This approach is faster and effective in obtaining coarse poses, but recent research shows that its performance is limited by the nature of image retrieval and generally less accurate [56]. In comparison, structure-based methods [5, 58, 7, 51, 72, 54, 62] gradually become the trend. and solve the problems in two stages: first, establishing the correspondences between the 2D query image pixels and the 3D scene points with deep learning methods; second, estimating the desired camera pose by PnP [26] combined with different RANSAC [18] 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 [51, 53, 54, 62]; 2) scene coordinate map regression [5, 58, 7, 72, 36]. The sparse feature matching methods first retrieve a set of similar database images via image retrieval approaches [1, 21]. The interest point detection and matching are performed between query images and retrieved scene images via handcrafted [39] or CNN-based [15, 53] features. One of the benefits of sparse feature matching is that it can handle arbitrary scenes. On the other hand, coordinate map regression methods predict dense 3D coordinates at all image pixels from a random forest [60] or a convolutional neural network (CNN) [5, 58]. The estimated dense coordinate maps can be effectively applied to augmented reality and robotics applications, e.g. virtual object insertion or obstacle avoidance. But these methods are often limited to the scene where the random forests or CNN is trained.

Therefore, in this thesis, we focus on the coordinate map regression approach. Most of the dense coordinate regression methods are scene-specific, which means the models are typically learned from images and 3D data for a specific scene and require retraining or adaptation before they can be applied to a novel scene. As a result, it is unknown how such a model can be quickly adapted to a different scene, which limits their applications when the scene is novel or progressively updated. Recently, instead of encoding specific scene information in network parameters [60, 5, 58], Yang et al. propose the first dense coordinate regression network, SANet, for arbitrary scenes [66]. The 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 [46] 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 solution 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.

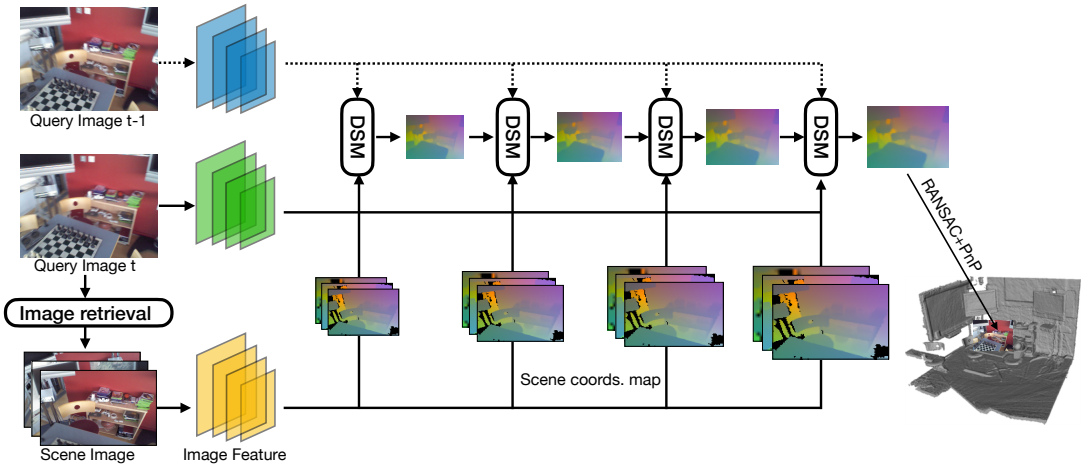


Figure 1.1: Overview of our framework. Our method predicts dense coordinate maps in a coarse-to-fine manner. The DSM module receives a query image feature map, some scene image feature maps and the corresponding scene coordinates to predict a dense coordinate map for the query image. These predicted scene coordinates are then used to solve camera poses with RANSAC and PnP algorithms.

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 [60] and large-scale outdoor scenes, Cambridge [33]. 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.

Chapter 2

Preliminary

2.1 Camera Localization Preliminary

Pinhole camera model. We first introduce the pinhole camera model, which serves as the foundation of the camera localization problem. As shown in Fig. 2.1, the pinhole camera model establishes a one-to-one mapping between points on the 3D object and the plane. The result is that the plane gets exposed by an "image" of the 3D object by means of this mapping. Let $\mathbf{p} = [x_w, x_w, x_w, 1]^T$ be a point on some 3D object under a world coordinate system $\mathcal{W}$ visible to the pinhole camera, then it is mapped to $\mathbf{x} = [u, v, 1]^T$. The notation $\mathbf{p}$ is the homogenous coordinate of a 3D scene point and $\mathbf{x}$ is the image coordinate. They satisfy the following equation,

$$\mathbf{x} \sim \mathbf{KTp}, \tag{2.1}$$

$$\mathbf{T} = [\mathbf{R}|\mathbf{t}], \tag{2.2}$$

where $\mathbf{K}$ is the 3×3 camera intrinsic matrix and $\mathbf{T}$ is the six degree of freedom camera transformation matrix. The notation $\sim$ implies that the left and right hand sides are equal up to a non-zero scalar multiplication. The translation $\mathbf{t}$ is a 3×1 vector describing the camera position. The rotation $\mathbf{R}$ is a 3×3 matrix describing the camera orientation and satisfying the following equation,

$$\mathbf{R}\mathbf{R}^T = \mathbf{I}, \tag{2.3}$$

where $\mathbf{I}$ is a 3×3 identity matrix. The transformation matrix, also called extrinsic matrix, first transforms the world coordinate of $\mathbf{p}$ to the camera coordinate. It is external to and do not depend on the camera, describing the camera position. On the other hand, The camera intrinsic matrix is unique and inherent to a given camera and relate to essential properties of the camera, such as its manufacturing.

Problem formulation. Given a query image $\mathcal{I}_q$ and a known camera intrinsic matrix $\mathbf{K}$, camera localization aims to estimate the camera pose $\mathbf{C} = [\mathbf{R}^{-1}, -\mathbf{R}^{-1}\mathbf{t}]$, represented by rotation $\mathbf{R}$ and translation $\mathbf{t}$, as shown in Eq. 2.2, under some world coordinate system $\mathcal{W}$ in a known environment.

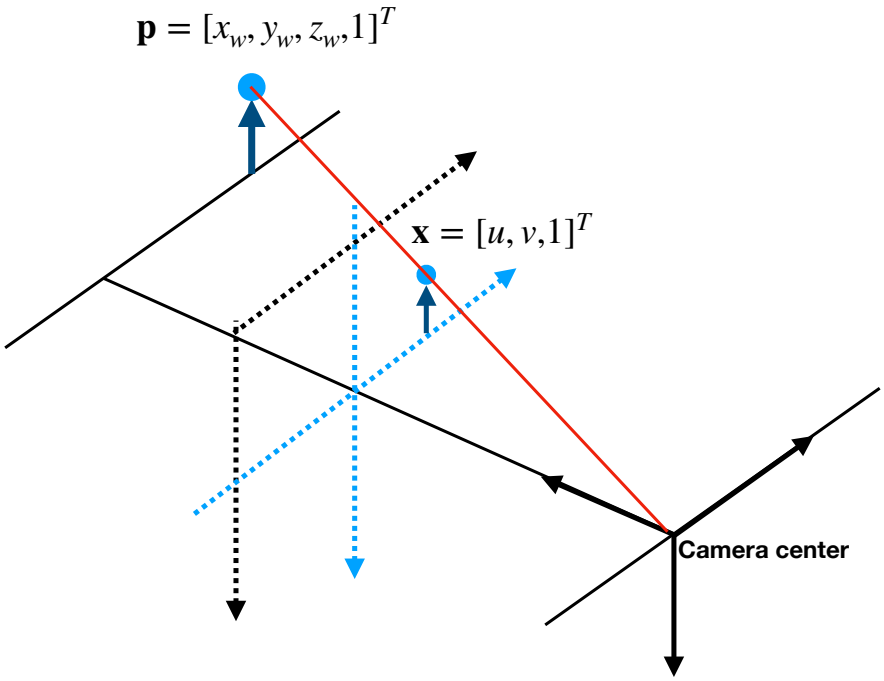


Figure 2.1: Pinhole camera model.

We can see that the camera pose $\mathbf{C}$ and the transformation matrix $\mathbf{T}$ can be computed if the other is known.

In this environment, prior knowledge is provided and exploited, through a 3D scene model. The 3D models are represented by a set of RGB or RGBD images with known 6 DoF camera poses in world coordinate system $\mathcal{W}$ obtained by 3D reconstruction algorithms, such as structure from motion [59] for RGB images or Kinectfusion [43] for RGBD images.

Previous method. In the following, we denote the query image as $\mathcal{I}_q$ and the database image as $\mathcal{I}_s$. The traditional way to address this problem is first building correspondences between a query image and each database image through key-point detection and matching [39], as shown in Fig.2.3. Suppose the correspondences are represented by a set of 2D positions $\{\mathbf{x}_q^i | i \geq 0\}$ in $\mathcal{I}_q$ and those $\{\mathbf{x}_s^i | i \geq 0\}$ in $\mathcal{I}_s$. Each image position $\mathbf{x}_q^i$ corresponds to a position $\mathbf{x}_s^i$. $\mathbf{x}_q^i$ and $\mathbf{x}_s^i$ are normalized image coordinates. To obtain the camera pose under $\mathcal{W}$, two methods are explored. If depth information of database images is not provided, we can compute the relative pose between the query image and the database image by using epipolar geometry. If $\mathbf{x}_q^i$ and $\mathbf{x}_s^i$ correspond to the same 3D point in the scene, they must satisfy the following equation,

$$(\mathbf{x}_q^i)^T \mathbf{E} \mathbf{x}_s^i = \mathbf{0}, \tag{2.4}$$

where $\mathbf{E}$ is the 3×3 essential matrix. Since each $\mathbf{x}_q^i$ and $\mathbf{x}_s^i$ is known, $\mathbf{E}$ can be estimated by the 5 point algorithm [44] or the 8 point [24] algorithm and thus the relative pose can be recovered from the essential matrix $\mathbf{E}$. The global camera pose of $\mathcal{I}_q$ is obtained by multiplying the relative

camera pose with the global pose of $\mathcal{I}_s$. If the depth maps of database images are provided, the 3D coordinates of $\mathbf{x}_s^i$ in $\mathcal{W}$ can be recovered by back projection, as following,

$$\mathbf{p}_s^i = \mathbf{C}_s d_s \mathbf{K}^{-1} \mathbf{x}_s^i, \tag{2.5}$$

where $\mathbf{p}_s^i$ is the 3D coordinate in $\mathcal{W}$, $\mathbf{C}_s$ is the camera pose of the image $\mathcal{I}_q$ and d_s represents the depth. Therefore, we can solve Eq. 2.1 by perspective-n-point algorithms [35] with known 2D-3D correspondences. Compared with relative pose estimation, this method can leverage 3D points across multiple images.

Recently, deep learning based approaches regress the 6 degree of freedom global camera poses $\mathbf{C}$ directly from RGB images with convolutional neural network. The input to the network is single images, while the output is the global positions and orientations of query images.

$$\mathbf{C} = \mathcal{F}(\mathcal{I}_q) \tag{2.6}$$

Where $\mathcal{F}$ is the function parametrized by CNN. These networks are trained in a supervised manner on database RGB images with known ground truths by a regression loss of pose errors.

Broadly, the camera localization describes the relationship between the sensor observations and map, by matching a query image or view against a pre-built model, and returning an estimate of the global pose.

2.2 Related Work

This section introduces related works: 1) Regression-based localization; 2) Structure-based localization; 3) Video localization and 4) Cost volume.

2.2.1 Regression-based localization

Regression-based localization regresses the camera pose of an image against a 3D map. Such 3D map is explicitly built by a geo-referenced database or implicitly encoded in a neural network. This can be further classified into 2 categories: 1) direct pose regression and 2) relative pose regression.
Direct Pose Regression. The well-known PoseNet [33] and its variants [33, 32, 8, 31] regress the 6 DoF absolute poses directly from RGB images with convolutional neural network. The input to the network is single images, while the output is the global position and orientation of query images. These networks are trained in a supervised manner on database RGB images with known ground truths by a regression loss of pose errors. It is noted that one network can only be trained for one scene and fails to generalize to other scenes without retraining. Intuitively, these methods train a network to memorize the poses of all RGB images in a database. It has been demonstrated in the work [56] 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.

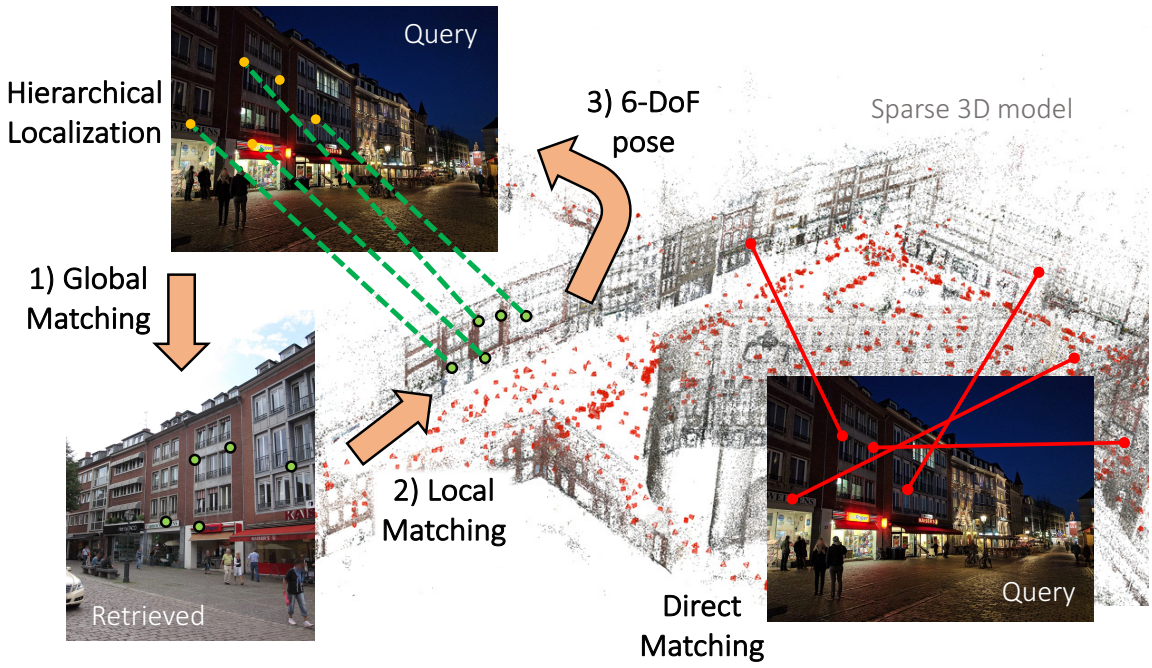


Figure 2.2: Pipeline of common structure-based localization

Relative Pose Regression. The relative pose regression methods use a coarse-to-fine strategy which first finds similar images in the database through image retrieval [25, 1] and then computes the relative poses w.r.t. the retrieved images [2, 34]. One problem here is how to find suitable image descriptors for image retrieval. Deep learning based approaches [1, 25] are based on a pre-trained convolutional neural network to extract visual features, and then use these features to evaluate the similarities against other images. The images with the top K highest similar values are considered as the retrieved ones. In order to obtain more accurate poses of the query image, additional relative 6 DoF pose estimation with respect to the retrieved image is desired. Traditionally, this is achieved by epipolar geometry, by using 2D-2D correspondences determined by local descriptors [39]. In contrast, deep learning approaches regress the relative poses straightforwardly from image pairs. For example, NN-Net [34] leverage convolutional neural networks to estimate the pairwise relative poses between the query images and the scene images, so that the absolute query pose can be naturally calculated by multiplying the relative poses with the scene image poses. They do not memorize the scene geometry and are thus scene-agnostic. However, similar to direct pose regression, they are not accurate.

2.2.2 Structure-based localization

These methods first build 2D-3D correspondences (2D pixels and 3D points), as shown in Fig. 2.2, and compute the global poses with respect to the 3D models. The poses are then solved by PnP algorithms [19] combined with RANSAC [18]. In the following section, two types, sparse feature matching and dense coordinate regression of Structure-based localization are introduced.

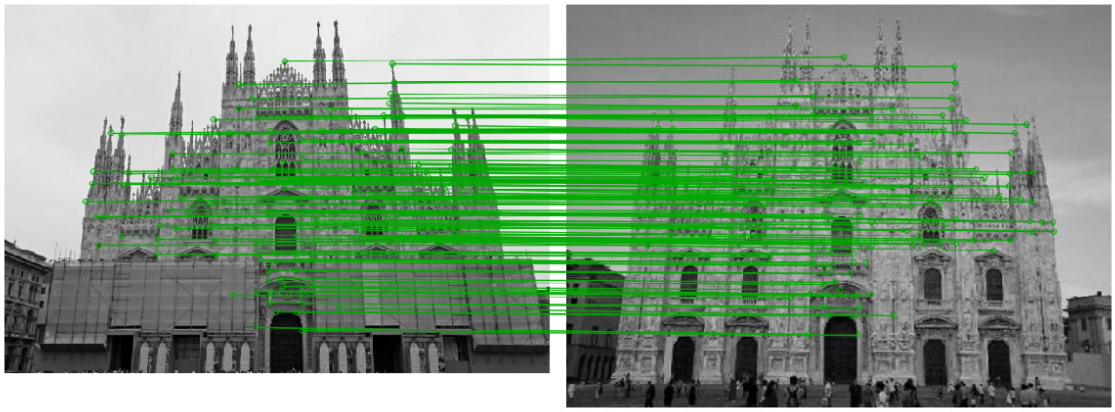


Figure 2.3: Correspondence visualization.

Sparse Feature Matching. These methods adopt a coarse-to-fine methodology [51, 62, 52, 42]. They first retrieve similar images with depth information from databases using image retrieval methods [25, 1]. Then, as shown in Fig. 2.3, between query images and retrieved images, 2D-3D correspondences are built [11, 55, 51, 62, 41, 41] by interest point detection [39, 15, 17, 4, 23, 42] and local descriptor matching [51, 15, 53, 17, 39, 9]. The other methods generally focus on improving the capability of local feature detectors [15, 17, 57, 70], descriptors[3, 71, 15, 17, 70] and correspondence matching[53]. A recent work along this direction is SuperGlue [53] and it achieves strong pose accuracy, especially in large-scale outdoor scenes. However, as limited by local feature descriptors, those methods tend to fail on scenes with textureless regions or repeated patterns. Instead, by leveraging global contexts, our method shows 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 match-based methods that establish 2D-3D correspondences before calculating pose, these methods directly regress the dense 3D scene coordinates in world coordinate system of the query image and obtain the final camera pose by dense 2D-3D correspondences [5, 6, 72, 7, 60, 36]. Fig. 2.4 is the visualization of coordinate maps. It can be viewed as learning a transformation from the query image to the global coordinates of the scene. Shotton et al. [60] propose to regress the scene coordinates using a Random Forest. Along this direction, DSAC [5] and DSAC++ [6] employ convolutional neural networks to predict a dense coordinate map from a single RGB image. 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 [66] 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 [46, 47] 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

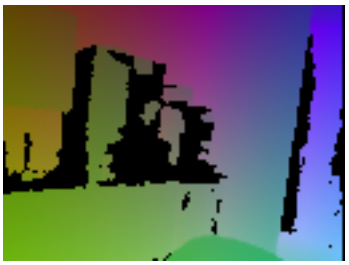


Figure 2.4: Visualization of coordinate maps.

compute 3D scene coordinates, which outperforms recent scene-specific and scene-agnostic methods including DSAC [5], DSAC++ [7] and SANet [66].

2.2.3 Video Localization

Many works have extended single frame localization to video localization. [12, 48, 63, 65, 72]. VidLoc [12] performs batch relocalization offline for fixed-length video-clips by long short term memory [27]. Coskun et al. refine camera poses by integrating LSTM units in the Kalman filters [13]. VLocNet [63] and VLocNet++ [48] propose to learn the visual odometry and pose regression jointly and achieve significant improvement over single frame localization. LSG [65] combines LSTM with visual odometry to further exploit the spatial-temporal consistency. Since all the methods are extensions of PoseNet, their accuracies are fundamentally limited by the retrieval nature of PoseNet. In contrast, KFNet is the first structure-based method for video localization. Specifically, a recurrent network named KFNet is proposed in the context of Bayesian learning by embedding coordinate regression into the Kalman filter within a deep learning framework. It achieves the top performance on both single frame and video localization tasks. However, KFNet is still scene specific since it regresses dense coordinate maps in the same way as DSAC++ [6].

2.2.4 Cost Volume

The proposed method in this thesis is inspired by the ideology of cost volume which has been widely adopted in computer vision tasks, e.g. optical flow[16, 61, 29, 49], stereo matching [10, 40, 45] and multi-view stereo [67, 22, 69]. 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 [61, 10]. For example, flownet [29] constructs a cost volume that stores the matching costs for associating a pixel with its corresponding pixels at the next frame. The cost is computed using the following equations.

$$cost(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{N} \mathrm{f}_1(\mathbf{x}_1)^T \mathrm{f}_2(\mathbf{x}_2) \tag{2.7}$$

Where x_1 and x_2 are two pixel localization of two different images, c_1 and c_2 refer to the feature maps and N is their channel dimensions. The matching cost can be considered as a similarity measurement between two image pixels. After building costs for each pixel pairs, they are arranged into

a $H \times W \times (H \times W)$ regular cost volume for convolutional neural networks to process. Similar to optical flow estimation, stereo matching adopts the same strategy. 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) builds 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 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.

Chapter 3

Methods

3.1 Overview

The overall framework of our system is illustrated in Fig.1.1. 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 [21]. Next, we extract a L-level feature pyramid for each query and scene image via the Feature Pyramid Network [37]. 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. In DSM, the correlations between query pixels and scene points are calculated and arranged into a regular cost volume for a convolutional neural network to process. 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 [37]. We denote the query feature maps as $\mathbf{F}_q^l$ and the scene feature maps as $\mathbf{F}_s^l$. The feature vectors in $\mathbf{F}_q^l$ are referred as $\mathbf{f}_q^l$, and those in $\mathbf{F}_s^l$ are $\mathbf{f}_s^l$. 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 $\mathbf{F}_s^l$. 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 $\mathbf{D}^{l+1}$.

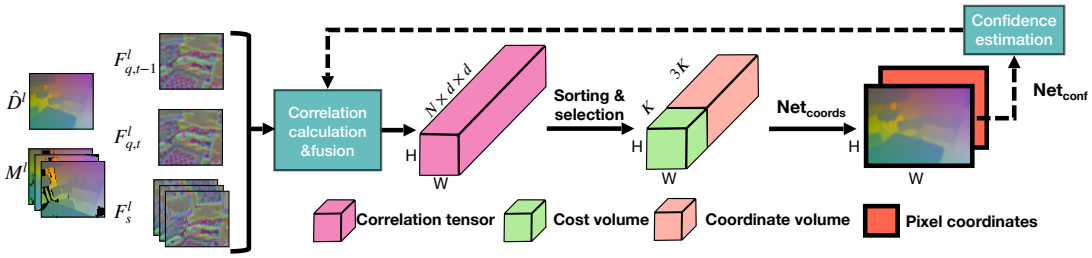


Figure 3.1: Illustration of Dense Scene Matching (DSM) module. For a specific pyramid level l, DSM takes 1) query image feature maps $\mathbf{F}^l_{q,t}$ at time t and $\mathbf{F}^l_{q,t-1}$ at time $t-1$; 2) scene image feature maps $\mathbf{F}^l_s$ with corresponding scene coordinates; 3) initial coordinate 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.

3.3 Dense Scene Matching

The overview of the DSM module is shown in Fig. 3.1. At a specific level l, the input of DSM module includes: 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 $\mathbf{F}^l_s$ 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.3.2, for an pixel $\mathbf{q}$ in the query feature map $\mathbf{F}^l_{q,t-1}$, 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, as shown in Eqn. 3.3.1. 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.

$$Corr_s = \frac{\mathrm{F}^l_{q,t}(q) \cdot \mathrm{F}^l_s(p)}{||\mathrm{F}^l_q(q)|| \cdot ||\mathrm{F}^l_s(p)||} \tag{3.1}$$

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 $\mathbf{p}'$. 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 $\mathbf{p}'$ and $\mathbf{q}$, as shown in Eqn. 3.3.1

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.

$$Corr_t = \frac{\mathrm{F}^l_q(q) \cdot \mathrm{F}^l_{q,t-1}(p')}{||\mathrm{F}^l_q(q)|| \cdot ||\mathrm{F}^l_{q,t-1}(p')||} \tag{3.2}$$

Correlation fusion. The final correlation score between a query pixel and a scene point is then computed from the scene correlation and the temporal correlation by the equation, $Corr = \alpha Corr_s + (1-\alpha)Corr_t$, where $Corr_s$ stands for scene correlation and $Corr_t$ is the 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 in descending order. After sorting, we apply non maximum suppression (NMS). We drop coordinates whose distance to a selected coordinate is smaller than a threshold. 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 a 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.

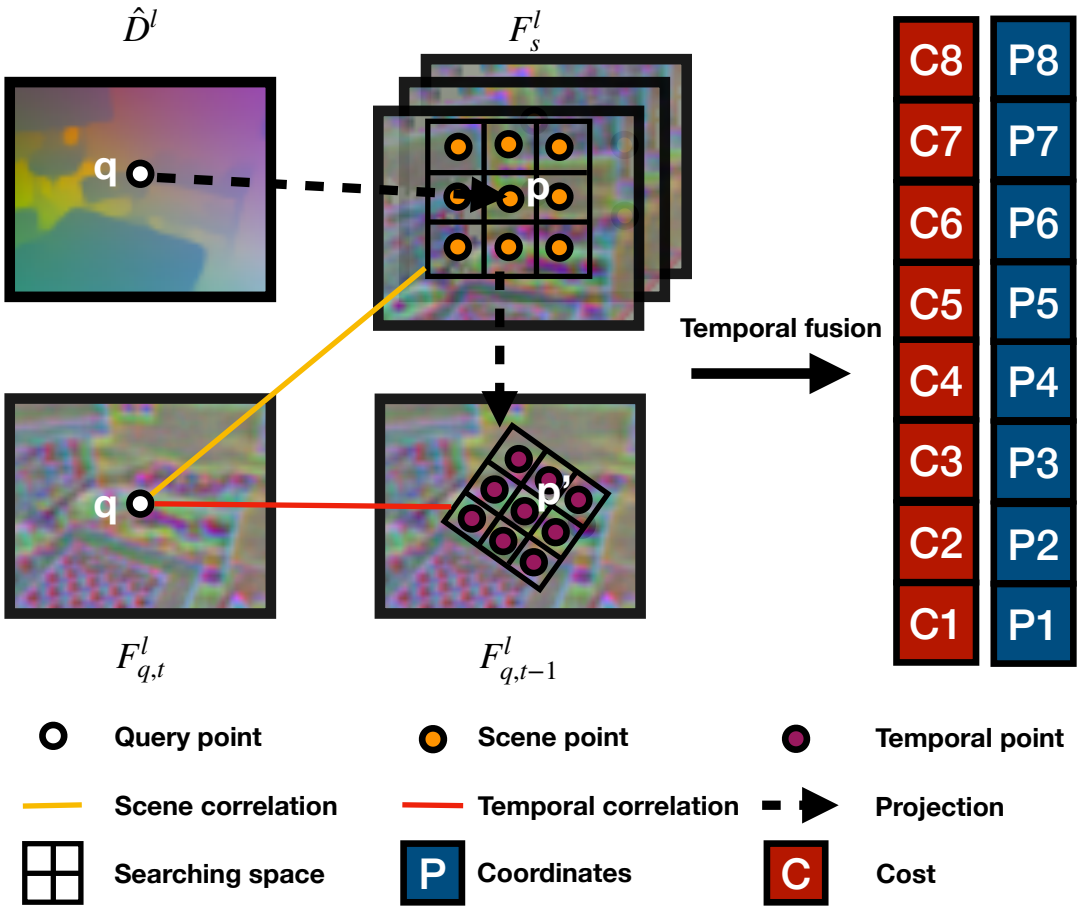


Figure 3.2: Demonstration of correlation fusion process. For a specific pixel $\mathbf{q}$ in $\mathbf{F}_{q,t}^l$, we obtain its scene correlation by projecting its corresponding 3D coordinate predicted in $\hat{\mathbf{D}}^l$ to a searching space of a $d \times d$ window in retrieved N scene feature 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.

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 is first fed into a network consisting of 1×1 convolutional layers and produce a coordinate feature map. This coordinate feature map is concatenated with the image feature map and fed into another network consisting of 3×3 convolutional layers to predict the final coordinate map.

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.3.1, after predicting the coordinate map from only scene correlation, we concatenate it with the corresponding 2D pixel coordinates (the pixel positions in images) and feed to Net_{conf} which outputs certainty scores. 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 [50] 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 Archtecture of Net_{coords} and Net_{conf}.

Fig. 3.3 shows the architecture of Net_{conf} and Net_{coords}. The input of Net_{coods} is a $H^l \times W^l \times 4K$ ($K = 16$ in implementation) cost-coordinate volume formed by concatenating the cost volume with 3D scene coordinates. As shown in Fig. 3.3, Net_{coords} consists of 3 residual blocks [25] and one denseblock [28]. The residual blocks consist of 1×1 convolutional layer. It takes the input of cost-coordinate volume and generates a $H^l \times W^l \times 64$ coordinate feature map. Then, the scene coordinate map is estimated by the denseblock, which takes the concatenation of image features, coordinate features and the initial coordinate map up-sampled from the last layer (if applicable). On the other hand, Net_{conf} consists of 5 residual blocks with context normalization [68]. It takes the concatenation of the estimated scene coordinate map with the corresponding 2D pixel coordinate map and estimates a confidence score for each pixel.

3.5 Training loss.

The total loss is the summation of the regression loss, $L_{regress}$, for coordinate regression and the average precision loss [50], L_{AP}, for training certainty scores. For coordinate regression, we use L_1

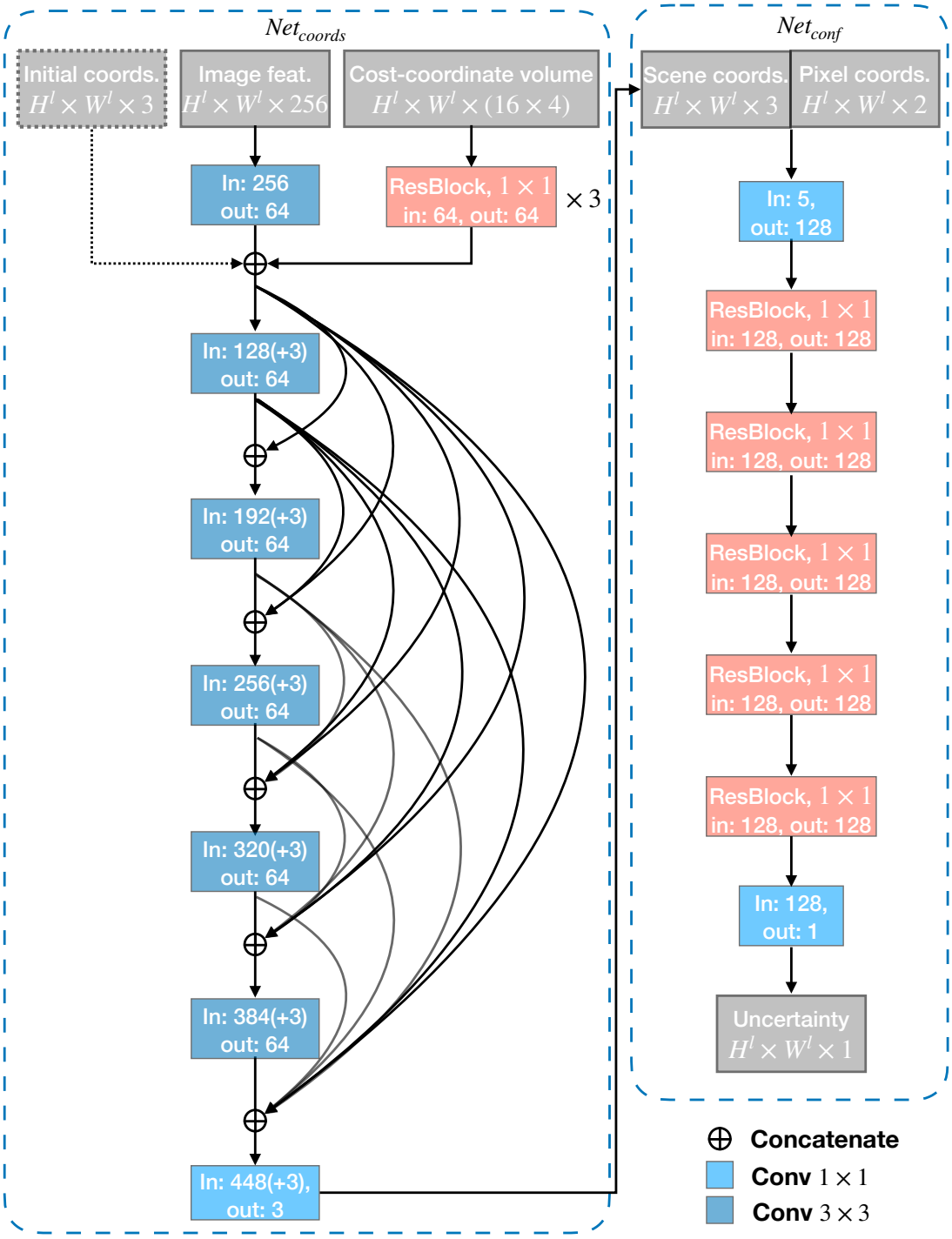


Figure 3.3: Archtecture of Net_{conf} and Net_{coords}. We use residual block for Net_{conf} and dense block for Net_{coords}

distance errors between predicted coordinates and ground truth coordinates as training loss.

$$L_{regress} = ||\mathbf{Y}_{coords} - \overline{\mathbf{Y}}_{coords}||$$
$$L = L_{AP} + \frac{1}{n}\sum_{i=0}^{n}(L_{regress})$$

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.

Chapter 4

Experiments

4.1 Experiment Settings

Dataset. We evaluate our method on both the indoor dataset 7scenes [20] and the outdoor dataset Cambridge Landmarks [33]. 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 [30]. 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 [14], 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 [20], Cambridge Landmarks [33] and ScanNet [14] 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 [21]. 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 [5] 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 [37] is regarded as our backbone network for all the following experiments. Our model is trained with an AdamW optimizer [38], 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 [54, 62, 51] and dense coordinate regression methods [5, 58, 72, 66]. We measure localization accuracy in terms of median errors in translation and rotation. As shown in Table. 4.1,

	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 4.1: 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 4.2: 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.

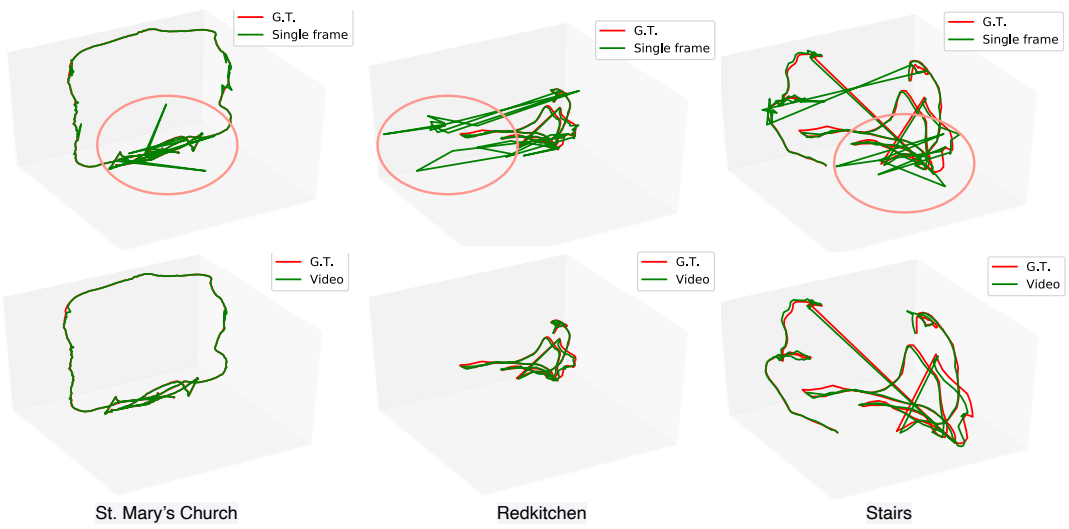


Figure 4.1: The comparison of camera trajectories between the single frame (first row) and video localization (second row) via the proposed dense scene matching network. The visualized results are respectively *Redkitchen* and *Stairs* in 7-scenes dataset, and *St. Mary's Church* sequence in Cambridge Landmarks dataset. In the first row, the outliers are shown in the red circles.

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 [54] and InLoc [62]. HLoc [51], upgraded with SuperPoint [15] for feature detection and Superglue [53] for feature correspondence matching, is considered and such upgrade brings higher relocalization accuracy compared with the original HLoc approach [51] as reported in the work [53]. 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 [5] by a large margin and obtains slightly superior performance than DSAC++ [6]. Even for KFNet [72] with the top performance on single frame and video localization tasks, our approach achieves comparable performance. In comparison with the scene-agnostic SANet [66], DSM shows obvious superior performance.

Table.4.2 shows the detailed comparison of metrics of median errors, mean errors and the pose accuracy falling within a certain accuracy threshold (*Acc. thresh*) between single frame localization and video localization methods. As shown, after applying the temporal fusion, the localization accuracy notably increases indeed. In addition, Fig. 4.1 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 contain some outliers while our video localization is able to remove most of them.

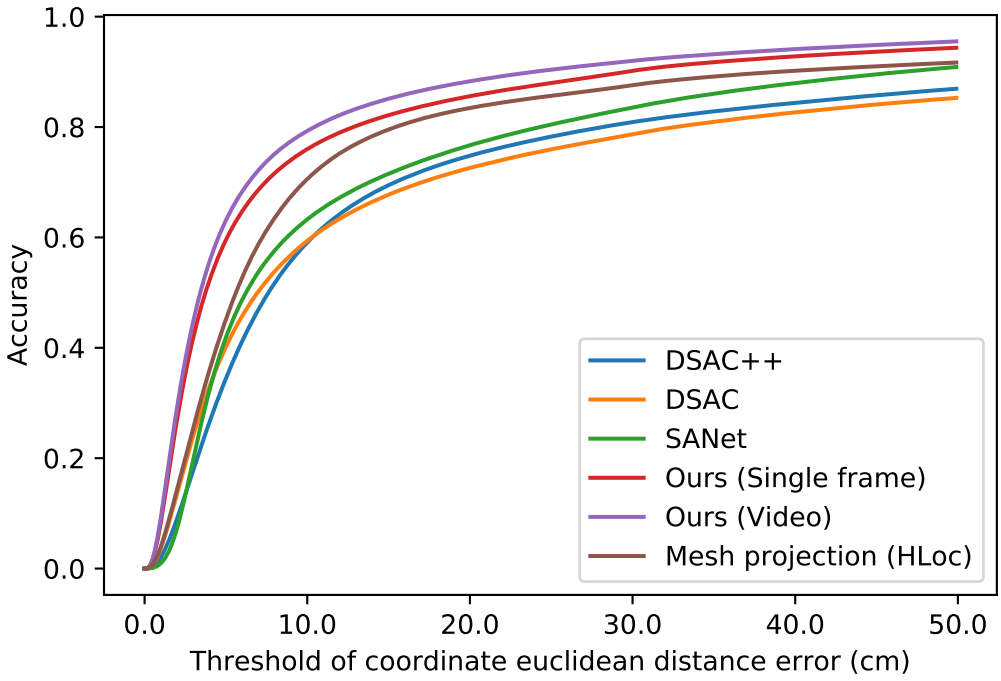
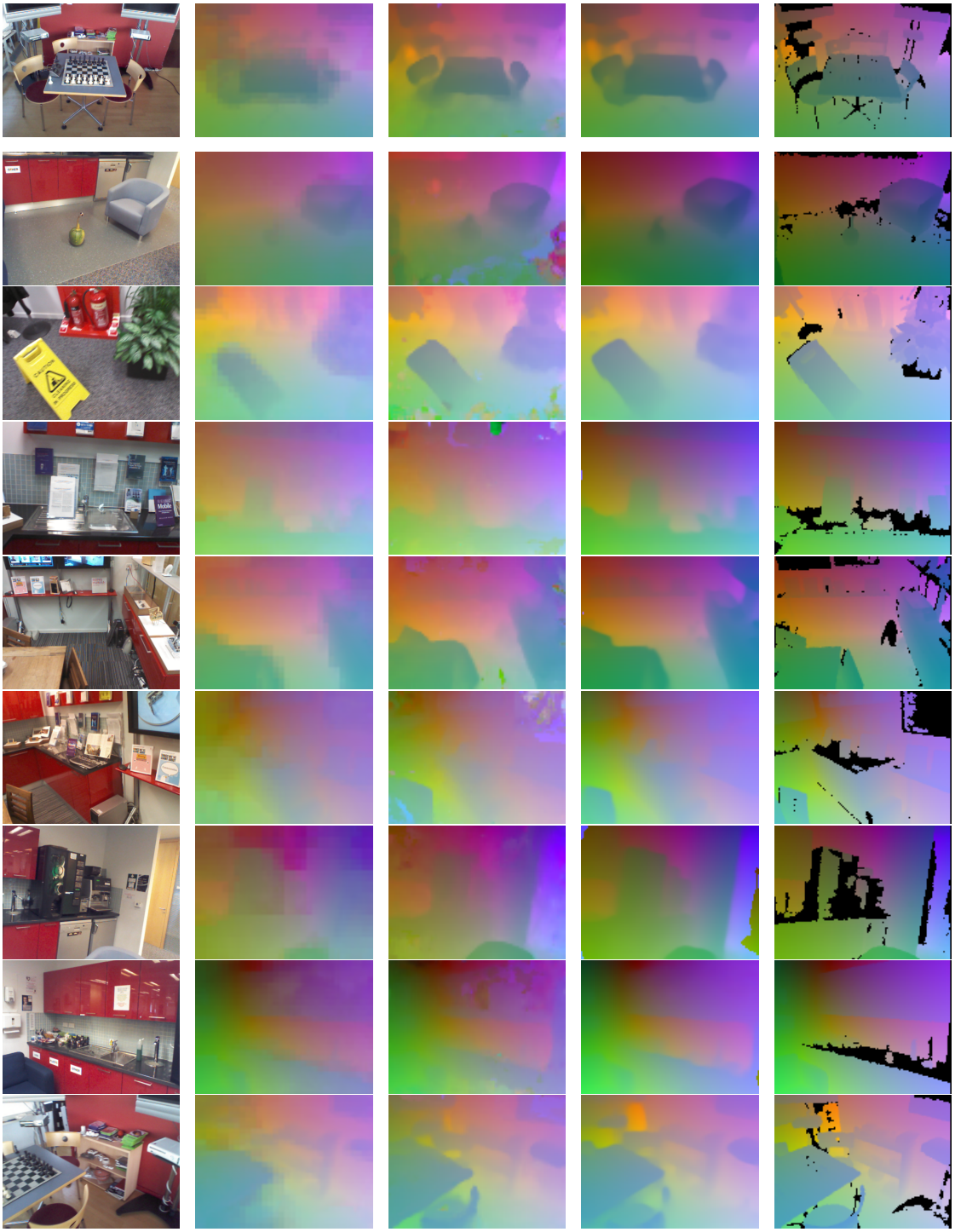


Figure 4.2: The comparison of cumulative distribution functions of scene coordinate errors between different localization approaches.



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

Figure 4.3: Coordinate map visualization for SANet, DSAC++ and DSM.

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

Table 4.3: Efficiency comparison of SANet and DSM.

4.3 Scene coordinate accuracy

In terms of scene coordinate accuracy, we compare our method with SANet [66], DSAC [5], DSAC++ [58] and HLoc [51] 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 the cumulative distribution function in Fig.4.2. 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.

We also visualize coordinate map in Fig.4.3 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 samples 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. DSAC++ is able to produce a coordinate map with more details, but artifacts exist in some regions as well.

4.4 Efficiency

Table.4.3 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 predicts dense coordinate maps for 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 adopting light-weight backbones.

4.5 Ablation Study

This section provides an analysis of DSM. All the experiments are conducted on 7scenes dataset. The data processing and training process are the same as described above. We use the following testing setting if not specified: at the inference time, 1 out of every 10 frames for each sequence is used to calculated pose accuracy, the percentage of predicted poses falling within the threshold (5°, 5cm).

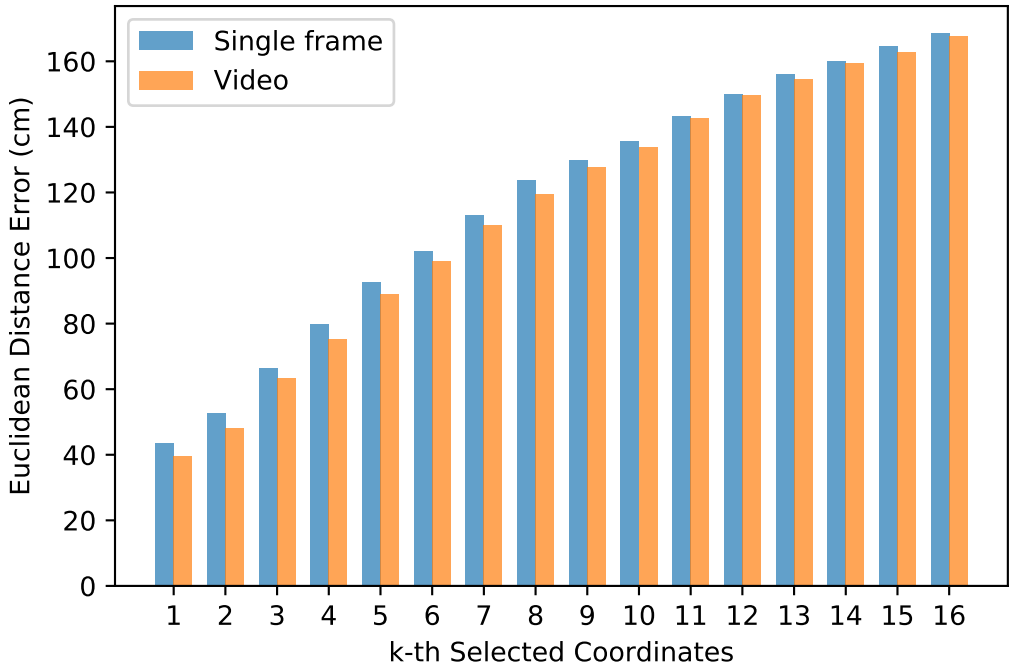


Figure 4.4: The average scene coordinate errors of the k-th selected coordinates with respect to ground truth.

	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
No sorting	0.82	0.74	0.85	0.72	0.43	0.58	0.05
Sorting	0.96	0.95	1.0	0.88	0.53	0.72	0.66

Table 4.4: Pose accuracy with/without top K correlation sorting. The estimated pose accuracy improves by correlation sorting consistently on all sequences.

Num.	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
1	0.87	0.85	0.87	0.71	0.45	0.63	0.17
3	0.90	0.94	0.91	0.79	0.46	0.67	0.20
5	0.94	0.94	0.94	0.80	0.54	0.68	0.24
10 ($\star$)	0.96	0.95	1.0	0.88	0.53	0.72	0.66

Table 4.5: Pose accuracy with respect to the number of scene images. The network is trained and tested with the corresponding number of scene images except the one with 10 scene images. The notation ($\star$) means we train the network with 5 scene images instead of 10 scene images.

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 the 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.4.4. 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 a temporal-based model, which obtains consistently lower euclidean distance errors than the single frame model, indicating that correlation fusion further improves the accuracy of selected scene coordinates.

Effects of correlation sorting. As described in Sec.3.3.1, one of the procedures in cost volume construction is sorting and selecting top K coordinates for each pixel from the correlation tensor. The motivation behind this operation is two-fold. Firstly, as the number of retrieved scene images varies, the top K selection results in a cost volume with a fixed size. Secondly, a sorted cost volume leads to a more accurate estimated coordinate map. To verify the effectiveness of correlation sorting, we fix the scene image number to 5 and directly use the correlation tensor as the cost volume for coordinate map regression. The results are shown in Table. 4.4. It can be seen that the estimated pose accuracy improves by correlation sorting consistently on all sequences. Moreover, since top K sorting and selection results in a fixed-size cost volume, we can use different scene image numbers for training and testing. During the training process, the scene image number can be fixed for better efficiency while for inference we can leverage more scene images for higher accuracy.

Importance of scene depth accuracy We also study the performance of DSM under noisy scene depths. We add some random noises to the ground truth scene depth map and calculate the localiza-

Reso.	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
192×256	0.92	0.84	0.89	0.78	0.49	0.64	0.23
384×512	0.96	0.95	1.0	0.88	0.53	0.72	0.66

Table 4.6: Pose accuracy with respect to different image resolutions. In our implementation, We resize all images to resolution of 384×512 for better efficiency and performance.

	S. Facade	S.M Church	K. College	O. Hospital	GreatCourt
w/o F.T.	0.53°, 0.09m	0.57°,0.16m	0.43°,0.26m	0.46°, 0.24m	0.40°,0.62m
w F.T.	0.31°, 0.06m	0.41°, 0.13m	0.36°, 0.22m	0.42°, 0.22m	0.35°, 0.44m

Table 4.7: The statistical comparison of median rotation and translation errors of Cambridge landmarks with or without fine-tuning.

tion accuracy. The noises are sampled from a standard Gaussian distribution multiply by a factor θ. Fig. 4.8 shows the overall localization accuracy under different noise values θ. We can see the localization accuracy drops when the noise value increases. The accuracy becomes 0 when θ is enlarged to 1.25.

Number of scene image. To show the effects of scene image number N, we change N from 1 to 10 to evaluate the pose accuracy. The model is re-trained with respect to the corresponding scene image number for $N = 1, 3, 5$. Since training with more than 5 scene images leads to unacceptable GPU memory consumption, we still use 5 scene images in training when testing with 10 scene images. As shown in Table 4.5, increasing N from 1 to 5 results in higher pose accuracy. In addition, we can see that 10 scene images obtain higher performance than 5 scene images. This indicates that even if the model is trained with fewer scene images, leveraging more scene images leads to better performance. Considering the trade-off between performance and efficiency, we set $N = 10$ in the main paper.

Image resolution. We test our model using 2 different image resolution size 192×256 and 384×512. As shown in Table. 4.6, we can see the resolution of 384×512 outperforms 192×256. A higher resolution than 384×512 could consume more GPU memory and lead to slower running time. Therefore, we resize all the images to 384×512 in our system for better efficiency.

Failure case analysis on single-frame localization. In Figure.4.5, we show some examples which are successful in video localization, but fail on single-frame localization. We can see that the query and scene images have small overlaps, featureless regions, repetitive texture patterns and large illu-

θ	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
0.00	0.96	0.95	1.0	0.88	0.53	0.72	0.66
0.25	0.75	0.77	0.66	0.68	0.44	0.53	0.34
0.5	0.46	0.31	0.18	0.38	0.23	0.33	0.17
0.75	0.27	0.15	0.12	0.18	0.13	0.16	0.11
1.00	0.17	0.08	0.06	0.10	0.05	0.08	0.07
1.25	0.04	0.02	0.01	0.03	0.00	0.00	0.00

Table 4.8: Localization accuracy under different

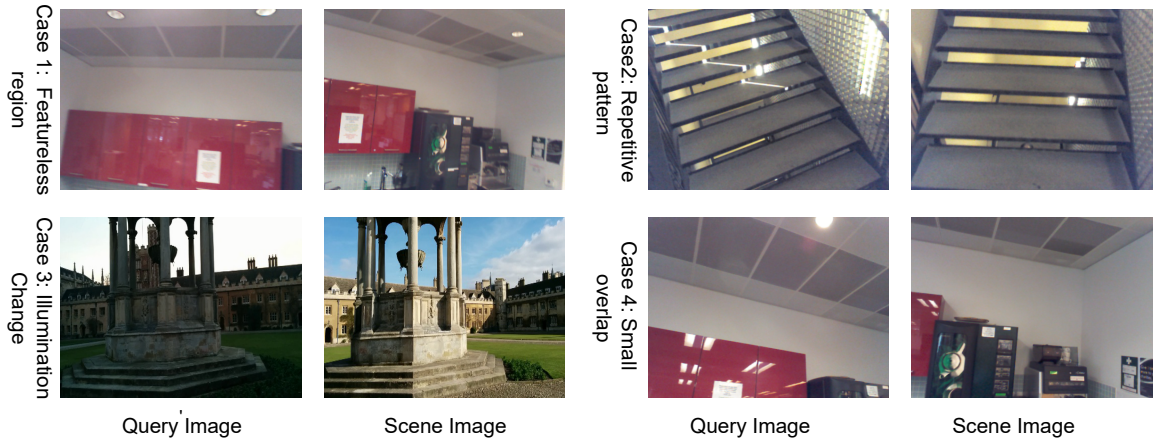


Figure 4.5: Examples fail in single-frame localization, but succeed in video localization. From the top left, top right, bottom left to the bottom right samples, the error of pose estimation reduces from single-frame localization (30.2°, 0.53m) to video localization (2.2°, 0.06m), (10.4°, 0.34m) to (2.2°, 0.04m), (19.4°, 0.05m) to (0.5°, 0.02m), (15.6°, 0.53m) to (1.4°, 0.07m) respectively.

mination differences. In such cases, scene correlations have lower weights while temporal correlations contribute more to the cost volume and lead to better performance.

Finetuing on Cambridge landmarks As shown in Table 4.7, we show that fine-tuning on Cambridge improves the performance.

Chapter 5

Conclusion

5.1 Discussions

Localization metrics. The common evaluation metrics are median errors of rotations and translations. These metrics measure errors in 3D space, so their absolute values depend on the scales of scenes. As a result, for small scale scenes, the improvement seems not obvious if looking at the absolute values. For example, in the scene Chess, which is a small room, DSM has $0.20^\circ, 0.01m$ lower errors than SANet while in the scene Street, which is a large street, DSM has $11.11^\circ, 8.05m$ lower errors. Median errors of rotations and translations are not consistent for scenes under different scales. In order to address this problem, a more reasonable metric would be the reprojection errors: the errors of projections of 3D points with ground truth poses and estimated poses in the image plane, as shown in the following equation,

$$E = \frac{1}{N}\sum_{i}^{N} \mathbf{K}\mathbf{T}_{gt}\mathbf{p}_i - \mathbf{K}\mathbf{T}_{pred}\mathbf{p}_i, \tag{5.1}$$

where $\mathbf{K}$ is the intrinsic matrix, $\mathbf{T_{gt}}$ and $\mathbf{T_{pred}}$ is the ground truth camera pose and estimated camera pose respectively, $\mathbf{p}_i$ is a 3D point of the image and N is the number of 3D points of the image.

Future work. Although DSM has achieved remarkable performance over 7scenes and Cambridge dataset, there are still deficiencies. Firstly, compared with feature matching methods, DSM requires dense depths of reference images, which are usually obtained by depth cameras or multi-view stereo algorithms. However, depth cameras only work in small-scale indoor scenes, while inaccurate depths from multi-view stereo algorithms may largely decrease the localization performance. This limits the usage of DSM to large-scale scenes, where accurate dense depths are hard to obtain. Secondly, DSM uses all 2D-3D pairs of an image to estimate 6 DoF camera poses. However, some of the scene coordinate predictions are not reliable since DSM can not find matches for those pixels in reference images. Including those pairs have a negative impact on localization performance. This problem is serious in sparse-sampled environments like InLoc [62]. The database images of InLoc are cropped from the original panorama images and the overlaps between two consecutive images

are small. Thirdly, our generalization to video localization leverages confidences estimated from a single frame and the errors of the previous frame is not considered. Therefore, if localization on the previous frame fails, temporal fusion could result in a worse performance.

To address the above problems, potential future work includes: 1) Leveraging sparse point clouds, which can be obtained by Lidar or triangulation conveniently, instead of dense depths to regress coordinates. 2) Using only accurate scene coordinate predictions to estimate the 6 DoF camera poses. 3) Designing a better temporal fusion module considering errors of both the current frame and the previous frame.

5.2 Conclusion

In this thesis, 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 is feed into a 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 the 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.

Bibliography

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[2] Vassileios Balntas, Shuda Li, and Victor Prisacariu. Relocnet: Continuous metric learning relocalisation using neural nets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 751–767, 2018.

[3] 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.

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

[5] 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.

[6] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662, 2018.

[7] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7525–7534, 2019.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6856–6864, 2017.

[13] Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5524–5532, 2017.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[20] 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.

[21] 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.

[22] 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.

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

[24] Richard I Hartley. In defense of the eight-point algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 19(6):580–593, 1997.

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

[26] 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.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia, and Juho Kannala. Camera relocalization by computing pairwise relative poses using convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 929–938, 2017.

[35] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[36] 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.

[37] 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.

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

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

[40] 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.

[41] 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.

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

[43] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011.

[44] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[45] 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.

[46] 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.

[47] 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.

[48] Noha Radwan, Abhinav Valada, and Wolfram Burgard. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, 3(4):4407–4414, 2018.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

[59] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[60] 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.

[61] 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.

[62] 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.

[63] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6939–6946. IEEE, 2018.

[64] 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.

[65] Fei Xue, Xin Wang, Zike Yan, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Local supports global: Deep camera relocalization with sequence enhancement. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2841–2850, 2019.

[66] 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.

[67] 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.

[68] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2666–2674, 2018.

[69] 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.

[70] 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.

[71] Hao Zhou, Torsten Sattler, and David W Jacobs. Evaluating local features for day-night matching. In *European Conference on Computer Vision*, pages 724–736. Springer, 2016.

[72] 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.