Image-Based Localization Using Deep Neural Networks

Xiaotian Li

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 2.10.2017

Thesis supervisor:

Prof. Juho Kannala

Thesis advisor:

D.Sc. (Tech.) Juha Ylioinas

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Xiaotian Li

Title: Image-Based Localization Using Deep Neural Networks

Date: 2.10.2017 Language: English Number of pages: 6+72

Degree programme: Automation and Electrical Engineering

Major: Control, Robotics and Autonomous Systems

Supervisor: Prof. Juho Kannala

Advisor: D.Sc. (Tech.) Juha Ylioinas

Image-based localization, or camera relocalization, is a fundamental problem in computer vision and robotics, and it refers to estimating camera pose from an image. It is a key component of many computer vision applications such as navigating autonomous vehicles and mobile robotics, simultaneous localization and mapping (SLAM), and augmented reality.

Currently, there are plenty of image-based localization methods proposed in the literature. Most state-of-the-art approaches are based on hand-crafted local features, such as SIFT, ORB, or SURF, and efficient 2D-to-3D matching using a 3D model. However, the limitations of the hand-crafted feature detector and descriptor become the bottleneck of these approaches. Recently, some promising deep neural network based localization approaches have been proposed. These approaches directly formulate 6 DoF pose estimation as a regression problem or use neural networks for generating 2D-3D correspondences, and thus no feature extraction or feature matching processes are required.

In this thesis, we first review two state-of-the-art approaches for image-based localization. The first approach is conventional hand-crafted local feature based (Active Search) and the second one is novel deep neural network based (DSAC). Building on the idea of DSAC, we then examine the use of conventional RANSAC and introduce a novel full-frame Coordinate CNN. We evaluate these methods on the 7-Scenes dataset of Microsoft Research, and extensive comparisons are made. The results show that our modifications to the original DSAC pipeline lead to better performance than the two state-of-the-art approaches.

Keywords: computer vision, machine learning, deep neural networks, image-based localization

Preface

This thesis was carried out under the supervision of Prof. Juho Kannala, and I would like to thank him for his guidance and support. I am grateful to my advisor Dr. Juha Ylioinas for all the help. Finally, I would like to thank my family for the love and constant encouragement.

Otaniemi, 02.10.2017

Xiaotian Li

Contents

Abbreviations and Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DLT	Direct Linear Transformation
DNN	Deep Neural Network
DoF	Degrees of Freedom
DSAC	Differentiable Sample Consensus
ELU	Exponential Linear Unit
FCN	Fully Convolutional Network
FLANN	Fast Library for Approximate Nearest Neighbors
GPS	Global Positioning System
IBL	Image-Based Localization
LSTM	Long-Short Term Memory
MLP	Multilayer Perceptron
ORB	Oriented FAST and Rotated BRIEF
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
RGB-D	Red-Green-Blue-Depth
SCoRF	Scene Coordinate Regression Forests
SfM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Features
VPS	Vocabulary-based Prioritized Search

1 Introduction

Image-based localization (IBL) is an important problem in computer vision and robotics. It addresses the problem of estimating the 6 DoF camera pose in an environment from an image. Recently, with the widespread use of smartphones equipped with modern cameras, IBL has received increasing attention since it is an important component of many interesting applications such as robot localization [10], simultaneous localization and mapping (SLAM) [13], place recognition [40], and augmented reality [6, 43].

There are also other forms of localization available to smartphones such as GPS and WiFi based localization. Unfortunately, GPS is only suitable for outdoor, rather than indoor environments, and is often inaccurate in urban environments with tall buildings. Although WiFi based localization is a good alternative for indoor environments, it, however, suffers several drawbacks. For example, its accuracy critically depends on the number of available access points, thus requiring the setup and maintenance of a significant number of devices, especially for large buildings. Besides, WiFi based localization system cannot provide the orientation of a user, thus being unsuitable for augmented reality applications.

Unlike GPS or WiFi based localization approaches, image-based localization does not have these limitations. It can be operated in both indoor and outdoor environments using the low-cost camera without the setup and maintenance of infrastructure. In addition, with the rapid development of computer vision, image-based localization is becoming more and more robust and accurate.

1.1 Problem Statement

Currently, there are plenty of image-based localization methods proposed in the literature. Most state-of-the-art approaches [56, 57, 58] are based on local features, such as SIFT, ORB, or SURF [42, 54, 2], and efficient 2D-to-3D matching. Given a 3D scene model, where each 3D point is associated with the image features from which it was triangulated, localizing a new query image against the model is solved by first finding a large set of matches between 2D image features and 3D points in the model via descriptor matching, and then using RANSAC [15] to reject outlier matches and estimate the camera pose on inliers. Although these local feature based methods have been proven to be very accurate and robust in many situations (mainly in outdoor environments), due to the limitations of the hand-crafted feature detector and descriptor, extremely large viewpoint changes, occlusions, repetitive structures and textureless scenes often produce a large number of false matchings which will lead to localization failure. Therefore, indoor image-based localization is still a very challenging problem since indoor scenes often have repetitive structures, less texture and insufficient local features for matching.

Recently, some promising neural network based localization approaches have been proposed. Approaches such as PoseNet [33] formulate 6 DoF pose estimation as a regression problem, and thus no traditional feature extraction, feature description, or feature matching processes are required. It has been shown that these approaches

overcome the limitations of the local feature based approaches, i.e., they are able to recover the camera pose in very challenging indoor environments where the traditional methods fail. However, their localization performance is still far below traditional approaches in other situations where local features perform well. Unlike PoseNet, approaches such as DSAC [3] obtain the 6 DoF pose by a two-stage pipeline: first, regressing a 3D point position for each pixel in an image and, then, determining the camera pose using RANSAC based on these correspondences as in the conventional localization pipeline. Although these methods require depth maps associated with input images at training time, their localization performance even surpasses the state-of-the-art local feature based approaches on indoor scenes.

In this thesis, the main task is to determine how well a state-of-the-art neural network based method can perform compared to a state-of-the-art conventional local feature based method and how can we improve upon the state-of-the-art neural network based method. We first review two state-of-the-art approaches for image-based localization. The first approach is Active Search [57, 58] which is conventional hand-crafted local feature based. The second approach is DSAC [3] based on patch-based Coordinate CNN and novel differentiable RANSAC. We then further explore the DSAC method and examine the use of conventional RANSAC in the DSAC pipeline. Finally, a novel full-frame Coordinate CNN is introduced. Experiments on a widely used dataset are conducted to evaluate the performance of these approaches.

1.2 Structure of the Thesis

The rest of the thesis is structured as follows. In Chapter 2, the background literature related to image-based localization and deep neural networks is presented. Here we give a brief introduction to several IBL methods and the building blocks of the neural network based methods. The detailed descriptions of the two state-of-the-art methods, Active Search and DSAC, are presented in Chapter 3 and 4 respectively. Chapter 5 discusses the modifications to the DSAC method. Chapter 6 describes the details of experiments and presents results of the experiments to determine the performance of the discussed methods and provide extensive comparisons between them. In Chapter 7, we discuss the results, and some directions of future study are given. Finally, Chapter 8 summarizes and concludes this thesis.

2 Background

Before diving deeper, we first review the theoretical background necessary for understanding the approaches discussed in the rest of the thesis. This includes a brief introduction to the history of image-based localization and some approaches to solving the localization task. Additionally, we present the theoretical background on deep neural networks required for the better understanding of the neural network based approaches.

2.1 Image-Based Localization

Solving the image-based localization problem consists of estimating the 6 DoF camera pose (3 DoF position and 3 DoF orientation) of a query image in an arbitrary scene. This is also known as camera relocalization. More specifically, the localization system takes only one image as input, and then outputs camera pose according to a given representation of the scene. The representation of the scene depends on the approach used. It can be a database of images, a reconstructed 3D model, or a trained deep neural network. Here we only consider one image as input, although the image-based localization task can be extended by using a sequence of images as input. The input images are usually RGB-only without depth information, since current low-cost devices are not commonly equipped with depth sensors.

In the earliest stages, image-based localization was solved by treating it as a location recognition problem [75]. In these approaches, image retrieval techniques are often applied to determine the location of the query image by matching it to a database of images and finding the most similar images to it. The locations of the database images are known. Thus, the location of the query image can be estimated according to the known locations. Initially, these methods worked on databases consisting of tens of thousands of images and then they were improved to deal with more than a million images. However, the localization performance provided by these approaches is limited by the accuracy of the known locations of the database images [56] and the density of the key-frames.

To obtain a better localization performance, more recent techniques [29] are based on more detailed and structured information, i.e., they use a reconstructed 3D point cloud model, usually obtained from Structure-from-Motion, to represent the scene. This is enabled by some powerful SfM approaches such as Bundler [63]. It is even possible to construct huge models with millions of points [65]. Thus, image-based localization systems are enabled to work on a city-scale level.

Instead of matching the query image to the database images, these keypoint based methods solve the localization task by finding correspondences between the query image and the 3D model. This is achieved by the use of the conventional hand-crafted features, e.g., SIFT [42]. The features of the 3D points are computed during the 3D reconstruction, and the features of the query image are detected and extracted at test time. Therefore, the correspondence search is formulated as a descriptor matching problem. After establishing the 2D-3D correspondences, the camera pose is determined by a Perspective-n-Point [35] solver inside a RANSAC

[15] loop.

It is obvious that the pose estimation step can only succeed if the quality of the established correspondences is good enough. When the 3D model is too big in size or too complex, an efficient and effective descriptor matching step is essential. There are several techniques in the literature to handle this problem, such as 2D-to-2D-to-3D matching [29], 3D-to-2D-matching [40], prioritized matching [56] and co-visibility filtering [57]. However, even if the matching process is enough effective and efficient, limitations of the hand-crated features will still cause these keypoint based localization systems to fail.

Recently, it has been shown that machine learning methods have great potential to tackle the problem of image-based localization. PoseNet [33] demonstrates the feasibility of formulating 6 DoF pose estimation as a regression problem. The pose of a query image is directly regressed by a deep CNN with GoogLeNet [66] architecture pre-trained on large-scale image classification data. However, its performance is still below the state-of-the-art keypoint based methods and far from ideal for practical camera relocalization. In order to improve the accuracy of PoseNet, several variants have been proposed in recent papers. For example, LSTM-Pose [69] makes use of LSTM units [26] on the CNN output to exploit the structured feature correlation. The LSTM units play the role of a structured dimensionality reduction on the feature vector and lead to drastic improvements in localization performance. Another variant, Hourglass-Pose [48], is based on hourglass architecture which consists of a chain of convolution and upconvolution layers followed by a regression part. The upconvolution layers are introduced to preserve the fine-grained information of the input image and this mechanism has been proven to be able to further improve the accuracy of image-based localization using CNN based architectures. Besides, it has been shown that the use of a novel loss function based on scene reprojection error can also significantly improve PoseNet's performance [32].

Unlike PoseNet, the scene coordinate regression forests (SCoRF) approach [61] adopts a regression forest [9] to generate 2D-3D matches from an RGB-D input image instead of directly regressing the camera pose. The final camera pose is then determined via a RANSAC based solver. This whole pipeline is similar to the keypoint based localization approaches, but no traditional feature extraction, feature description, or feature matching processes are required. The original SCoRF pipeline is further improved by exploiting uncertainty in the model [68]. Training the random forest to predict multimodal distributions of scene coordinates results in increased pose accuracy. Although these methods are extremely accurate, they require depth information during test time.

In order to localize RGB-only images as well, the original SCoRF is extended by utilizing the increased predictive power of an auto-context random forest [4]. In the most recent DSAC paper [3], a neural network based SCoRF pipeline for RGB-only images is proposed. In contrast to previous SCoRF pipelines, two CNNs are adopted for predicting scene coordinates and for scoring hypotheses. Moreover, the conventional RANSAC is replaced by a new differentiable RANSAC, which enables the whole pipeline to be trained end-to-end.

2.2 Approaches to Localization

In this section, we present three approaches to solve the localization problem related to this thesis.

2.2.1 Keypoint Based Localization

In order to determine the full camera pose, i.e., both position and orientation, we need to know the detailed 3D structure of the scene. Fortunately, such a 3D model of discriminative feature points can be obtained efficiently from a set of images using modern Structure-from-Motion techniques. During the reconstruction process, as each 3D point is obtained by triangulation made on corresponding image features, every 3D point in sparse point cloud model is already associated with feature descriptors. Thus, registering a 2D image to the 3D model can be solved by matching 2D image points and 3D model points. Then, the pose of the query image can be solved by feeding theses 2D-3D correspondences into a PnP [35] algorithm. Since the correspondences are usually not perfect but rather noisy, the PnP algorithm is used in combination with random sample consensus (RANSAC) [15] to be robust to outliers.

Keypoint based localization problem is essentially a descriptor matching problem. However, if the 3D point cloud model is too large or dense, the descriptor matching is typically inaccurate and computationally expensive. Therefore, several matching techniques have been proposed to make the descriptor matching efficient and effective. The matching techniques can be divided into two main classes: indirect and direct matching. Indirect matching approaches usually use an intermediate construct to represent the 3D point descriptors. For example, since 3D point descriptors are generated from database images, we can establish 2D-3D correspondences by first finding database images similar to the query image and then matching the query image against these database images (2D-to-2D-to-3D matching). Another example is to do 3D-to-2D matching, i.e., matching the model against the query image with co-visibility information. Contrary to indirect matching, direct matching techniques directly match the 2D image features against 3D points, i.e., they try to find the nearest neighbors of a 2D image feature in the space containing the 3D point descriptors.

It has been shown that classical direct matching techniques such as approximative tree-based search [49] achieve a better performance than indirect matching techniques, but they are much more computationally expensive, especially when the 3D point cloud model is very large and dense. In contrast, indirect matching methods are memory efficient and extremely faster than direct matching. However, they are not effective as they are more prone to fail to register an image.

In order to perform the matching both efficiently and effectively, Active Search [57, 58], which combines direct matching and indirect matching via a visual vocabulary based prioritization scheme, has been proposed. More details of Active Search are presented in Chapter 3.

2.2.2 PoseNet

PoseNet [33] first demonstrates the feasibility of casting pose estimation of a single RGB image as a regression problem and solves it using a deep neural network. Contrary to formulating localization as a classification problem which can estimate only the position of the image, this method recovers the full camera pose $\mathbf{p} = [\mathbf{x}, \mathbf{q}]$, where $\mathbf{x}$ is the 3D camera position and $\mathbf{q}$ is a quaternion representing the orientation.

The PoseNet system is simple. Only a convolutional neural network is used to estimate the camera pose. Therefore, it only takes 5ms to run, meeting the real-time requirement of many applications [33]. However, its localization performance is far below the state-of-the-art methods.

To regress pose, the convolutional neural network is trained in an end-to-end manner with the following objective loss function:

$$loss(I) = \|\hat{\mathbf{x}} - \mathbf{x}\|_2 + \beta \|\hat{\mathbf{q}} - \frac{\mathbf{q}}{\|\mathbf{q}\|}\|_2 \tag{1}$$

where $[\mathbf{x}, \mathbf{q}]$ and $[\hat{\mathbf{x}}, \hat{\mathbf{q}}]$ are ground truth pose and estimated pose respectively. Here β is a coefficient used to balance the position and orientation errors, and thus it is in general bigger for outdoor scenes due to larger positional error [33]. It has been found that learning position and orientation jointly can achieve better results than learning to regress them separately. Since the orientation here is represented by quaternions and the network cannot be regularized to generate only valid unit quaternions, the estimated quaternion should be normalized.

PoseNet adopts GoogLeNet model [66] as a basis to do pose regression. The final PoseNet architecture is illustrated in Figure 1. Some modifications are made to the original GoogLeNet architecture:

- All three softmax classifiers are replaced by affine regressors.
- A fully connected layer of width 2048 is added before the final regressors.

In addition, it has been shown that leveraging the idea of transfer learning to start the pose training from a network pre-trained on giant ImageNet [12] and Places [76] datasets can result in better localization performance and allow training on only a few examples.

A number of variants of PoseNet have been proposed to improve the localization performance, such as LSTM-Pose [69] and Hourglass-Pose [48]. Their architectures are illustrated in Figure 2 and Figure 3 respectively.

2.2.3 Scene Coordinate Regression Forests

Scene coordinate regression forests (SCoRF) [61] system is similar to the traditional keypoint based approaches: first generates point correspondences and then determines the final pose inside a RANSAC loop. However, unlike the traditional methods which formulate the 2D-3D correspondence search as a descriptor matching problem, SCoRF uses a regression forest [9] to directly regress the 3D scene coordinates in

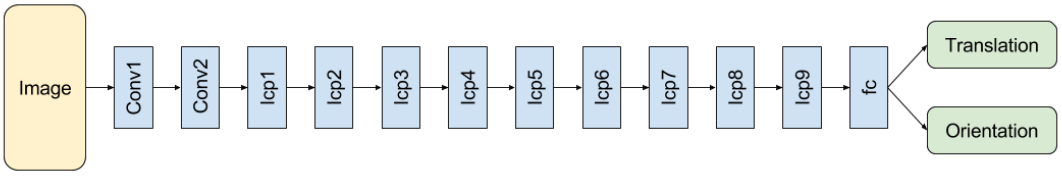


Figure 1: Architecture of PoseNet. Conv and fc are the convolutional layer and fully connected layer respectively (described in Section 2.3.3). Icp is the Inception module of GoogLeNet [66]. Figure adapted from [71].

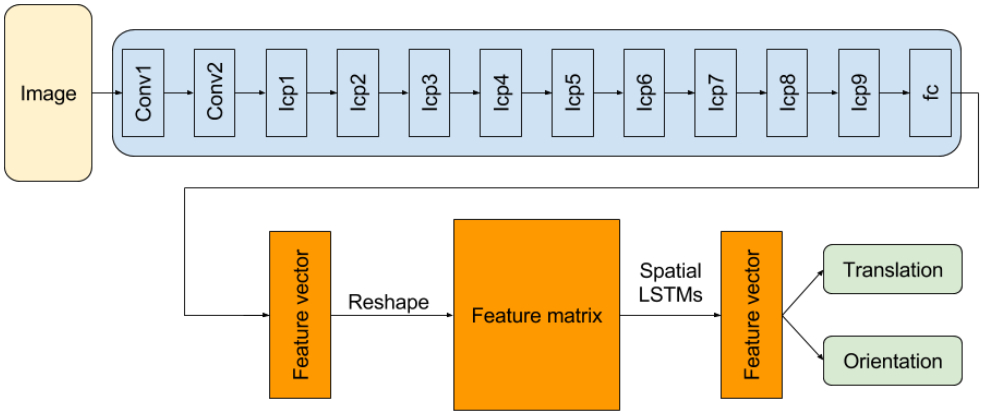


Figure 2: Architecture of LSTM-Pose. Figure adapted from [69].

the world space from the pixels of an RGB-D query image. Since at test time, the 3D coordinates of image pixels in camera space can be computed from the depth information and known camera intrinsics, 2D-3D correspondences become 3D-3D correspondences and the final pose is solved by the Kabsch algorithm [31] instead of the PnP algorithm. The whole SCoRF pipeline is illustrated in Figure 4.

In the SCoRF pipeline, the dense 2D pixel to 3D scene coordinate correspondences are learned using a dataset of RGB-D images. The ground truth camera poses of these images are obtained using KinectFusion [30, 51]. Using the known camera intrinsics, the depth information and the ground truth camera poses, 3D world coordinates of image pixels can be determined. This gives the ground truth 2D-3D correspondences. Then, a regression forest is trained using these correspondences. Both RGB and depth features at a pixel location are used by this regression forest to predict the corresponding 3D position of the pixel in the world space. Because the forest is trained using the densely sampled 2D-3D correspondences, it can be evaluated at any pixel location of a test image. Therefore, at test time, the forest can be efficiently applied to only a sparsely sampled set of image pixels to generate 2D-3D

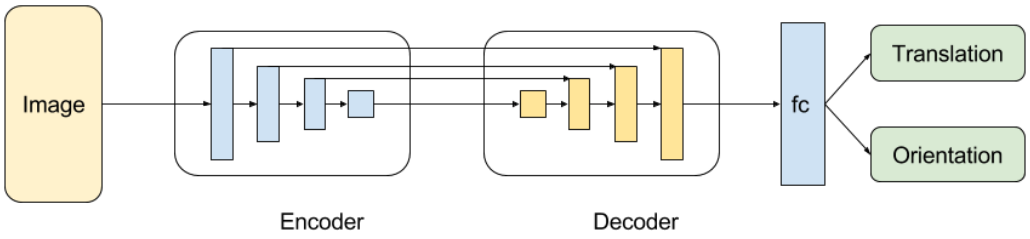


Figure 3: Architecture of Hourglass-Pose. Figure adapted from [48].

correspondences. In this way, the regression forest can be seen as an implicit 3D map of the scene. Finally, with this sparse set of correspondences and the computed 3D coordinates in camera space, an adapted version of preemptive RANSAC is used to obtain an accurate final camera pose.

The SCoRF localization pipeline has been extended in several works. In the recent DSAC paper [3], an RGB-only CNN-based SCoRF pipeline is introduced. The DSAC method is explained in detail in Chapter 4.

2.3 Artificial Neural Networks

Deep learning is currently one of the most popular and important research topics in machine learning, computer vision and artificial intelligence.

Traditional artificial neural networks (ANNs) are usually composed of one input layer, one output layer, and a few hidden layers in between. Deep neural networks (DNNs) are distinguished from the traditional ones by their depth. That is, they are ANNs with multiple hidden layers.

In this section, we first discuss the basics of artificial neural networks and then give an introduction to the convolutional neural network (CNN) which is a class of successful deep learning models applied to solve visual tasks.

2.3.1 Multilayer Perceptrons

Artificial neural networks are inspired by our understanding of the biological neural networks which constitute animal brains. They are designed to mimic both the structure and the functionality of the biological neural networks. However, biological neural networks and their activity are far more complex than artificial neural networks and the mysteries of the brain are still unsolved. Therefore, it is misleading to overemphasize the connection between them.

An artificial neuron is the fundamental building block of a neural network. The standard model of a single artificial neuron is shown in Figure 5. It consists of four basic elements: a set of weights w_{ij}, a linear combiner which sums the weighted inputs, a bias term b added to the weighted sum, and an activation function $\phi(\cdot)$ applied to the linear response of a neuron, which introduces the nonlinearity. Mathematically,

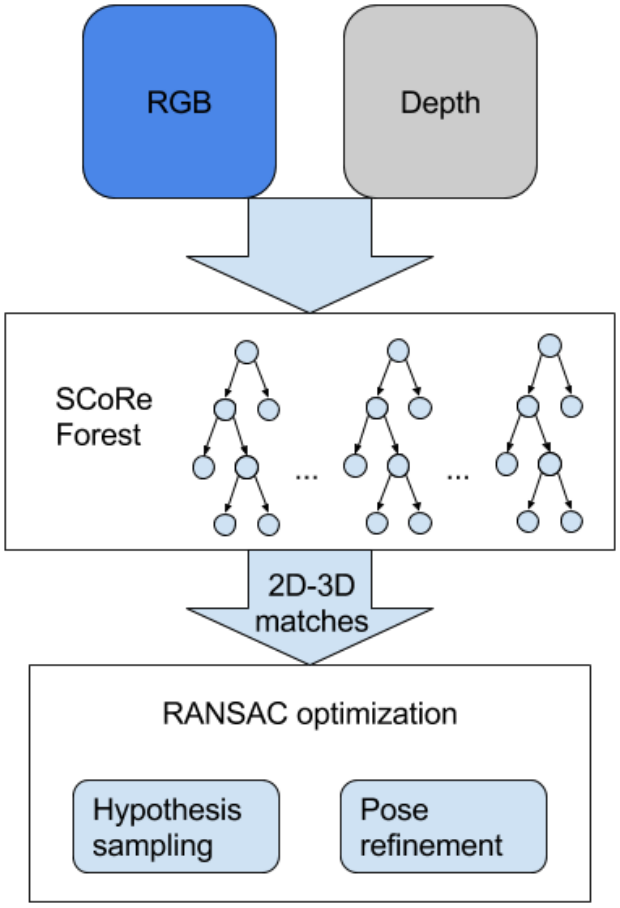


Figure 4: SCoRF pipeline. Figure adapted from [61].

given the weight vector $\mathbf{w}_j$ (the weights w_{ij}), the bias b, and the non-linear activation function $\phi(\cdot)$, the output of a single artificial neuron with n input values can be written as:

$$O_j = \phi(\sum_{i=1}^{n} w_{ij}x_i + b) = \phi(\mathbf{w}_j^T\mathbf{x} + b) \tag{2}$$

Several artificial neurons can further be interconnected to compose a neural network. Multilayer perceptron (MLP) is one of the most basic types of neural networks. A set of neurons with a common activation function is typically grouped into a layer. Multiple consecutive layers are then arranged into a neural network in a fully connected and feedforward manner. That is, all the neurons between subsequent layers are connected and output of a previous layer is fed as input to the next layer. A multilayer perceptron consists of three types of layers: input layer,

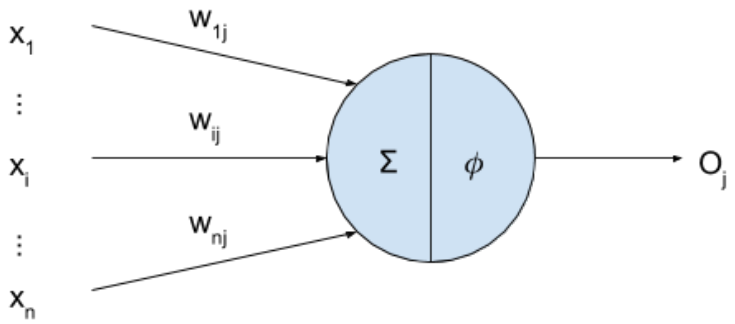


Figure 5: Basic model of an artificial neuron.

hidden layer, and output layer. Typically, there is only one input layer and one output layer, but the number of hidden layers can be larger than one. Figure 6 shows an illustration of a fully connected multilayer perceptron with one hidden layer. Formally, a one-hidden-layer MLP is a function $f(\cdot)$:

$$f(\mathbf{x}) = \phi_2(\mathbf{W_2}\phi_1(\mathbf{W_1}\mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}) \tag{3}$$

where $\mathbf{b_1}$ and $\mathbf{b_2}$ are bias vectors, $\mathbf{W_2}$ and $\mathbf{W_2}$ are weight matrices, $\phi_1(\cdot)$ and $\phi_2(\cdot)$ are activation functions. This can be easily extended to represent multilayer perceptron with more hidden layers.

The width of each layer is determined by the number of artificial neurons in each layer and the depth of the network is determined by the number of layers. A multilayer perceptron with at least one hidden layer has universal approximation property [27]. That is, it can approximate smooth nonlinear mappings with any desired degree of accuracy provided that the hidden layers are sufficiently wide. In fact, a deep but narrow neural network is usually more efficient to approximate the same function.

2.3.2 Backpropagation

Artificial neural networks are typically highly non-linear and thus have no closed-form analytical solutions. Therefore, they are trained numerically in a supervised manner using backpropagation algorithm. The standard backpropagation algorithm [55] is essentially an instantaneous stochastic gradient algorithm.

A neural network can be seen as a function $f(x \mid \theta)$ of input x parameterized by its weight parameters θ. The goal to train a neural network is to find the θ such that

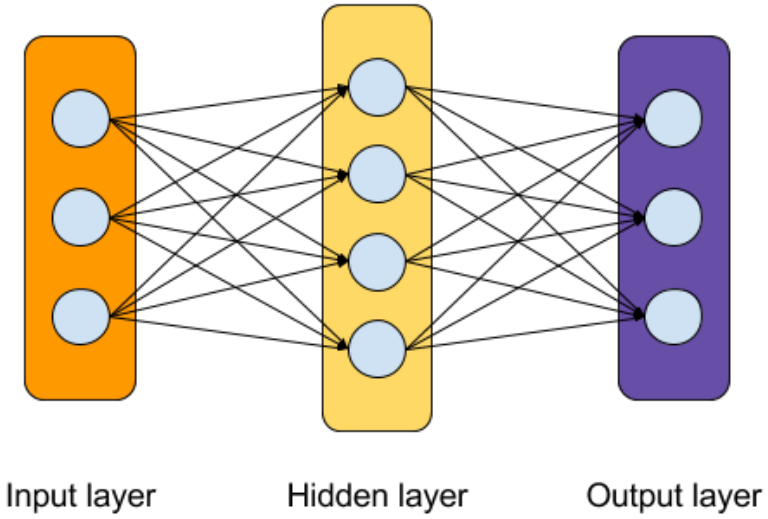


Figure 6: A fully connected multilayer perceptron.

the neural network can approximate a target function $f^*(x)$. That is, we want to find the θ that minimize the error between the desired output of the target function and the corresponding output of the neural network. Typically, the entire target function is unknown and only a set of training data is given, which can represent the function to some extent. Therefore, the error minimized during training is computed using the training data. In principle, the backpropagation algorithm tries to minimize the error and solve the weighs iteratively.

The mathematical description and derivation of the backpropagation algorithm are presented in many books (e.g. [22]) and thus we skip it. Here we briefly explain the steps of the standard backpropagation algorithm.

The first step of training is to initialize the weights of the network, for example to small random values. Then, a mini-batch of training data is sampled and the set of corresponding outputs of the network is calculated. The generated output predictions are compared with the corresponding ground truth training labels, and the training error is computed based on this comparison. Subsequently, the output error is propagated back through the network, and the gradient of each weighting parameter is calculated. Finally, the weights are updated using these gradients according to some specific rule. These steps except the initialization are repeated until the weight parameters converge.

The standard backpropagation algorithm converges slowly due to the use of the vanilla stochastic gradient descent, and bad choices for the learning rate can cause problems. Therefore, more efficient versions of backpropagation algorithm have

been proposed. One variant of it, Levenberg-Marquardt backpropagation (LMBP) algorithm [23, 45], uses a simplified version of Newton's method for training to speed up the convergence. Some other variants, such as RMSProp [67] and Adam [34], use adaptive learning rate when updating the weights.

2.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [38] are variants of MLPs which are well-suited to process data with grid-like structure such as images. CNNs allow processing of large digital images, which is impossible using conventional fully connected MLP networks. They are extremely successful in practical applications, such as [19, 25, 36, 44, 52, 62]. Therefore, the development of convolutional networks has tremendously increased the popularity of deep learning.

Convolutional neural networks are specially structured neural networks using convolution operation in at least one layer, but typically using a stack of convolutional layers. Due to the use of convolution operation and weight-sharing, CNNs have translation invariance characteristics.

Convolutional neural networks were inspired by biological visual processes [28]. Animal visual cortex has many simple neurons acting as local feature detectors. Each neuron is responsible for a particular region of the visual field (i.e. its receptive field) and is sensitive to a certain type of stimulus. For example, some neurons respond to edges with a specific orientation. The receptive fields of neurons are restricted in size, but they overlap each other and cover the whole visual field. On top of these simple neurons, there are more complex neurons which pool the responses of the simple ones, and thus there is a feature hierarchy.

In convolutional neural networks, the function of the neurons in the visual cortex is mimicked by applying convolution operation between the image and filters. That is, the filter is slid over the image spatially to compute dot products. Filters are trained to represent different features. A sequence of convolutional layers approximates the feature hierarchy in the visual cortex. It has been shown that the lower layers closer to the input learn to recognize simple features of the image, such as edges and bright spots, and the higher layers closer to the output learn to represent complex features of the image, such as shapes and patterns.

Similar to an MLP, a CNN also has one input and one output layer, and several hidden layers in between. Typically, three types of layers are used in convolutional neural networks: convolutional, pooling and fully connected.

Typically, a convolutional layer consists of N filters of size $F_x \times F_y \times d$ and operates on an input volume of size $I_x \times I_y \times d$. As mentioned before, each filter is slid over the input volume spatially to compute dot products at each 2D position to yield an output volume of size $O_x \times O_y \times 1$. The output volume is called activation map or feature map. This operation is also known as the convolution of two functions and thus we call the layer convolutional. Figure 7 is an example of convolution operation.

The size of the output feature map is determined by the strides of the convolution operation and the padding scheme. Strides determine the number of pixels with

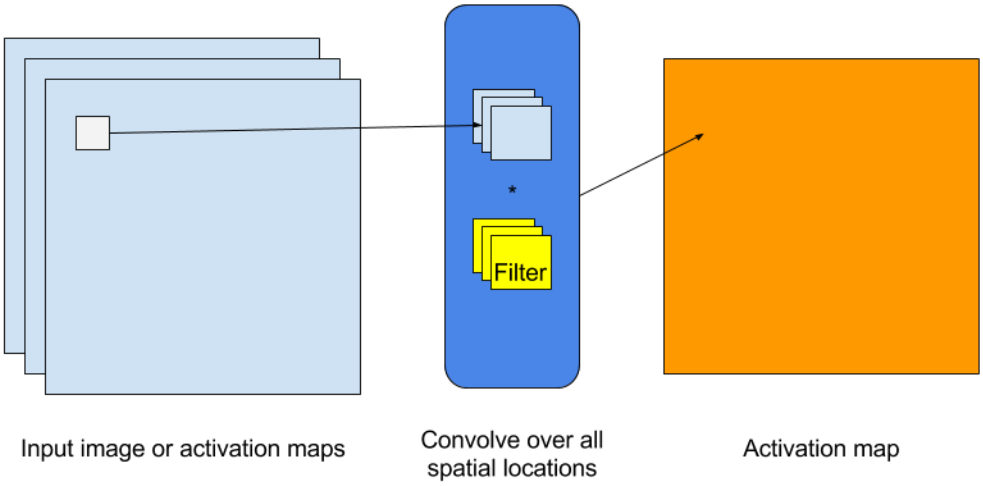


Figure 7: Convolution operation.

which we slide the filter at a time, horizontally or vertically. In other words, in case the strides are both one, we move our filter one pixel at each step, and the convolution output for every pixel is computed. Convolution operation typically does not preserve the spatial size of the input volume. Thus, it is usually useful to add zero padding to the border of the input volume. Without padding, the size of the output volume is always smaller than the input if the size of the filter is larger than 1×1.

Finally, the N separate activation maps generated by the N filters are stacked along the 3rd dimension. This yields the final output of a convolutional layer of size $O_x \times O_y \times N$. A convolutional layer with ten filters is illustrated in Figure 8.

Pooling layers are used to make the feature representations smaller in size and thus more manageable, while preserving the most important information in them. As illustrated in Figure 9, a pooling layer operates over each activation map independently to reduce the size of an input volume. This is done by sliding a small window across an input activation map and taking the maximum value or average value of the window at each step. These two different pooling schemes are known as max pooling and average pooling. As an example, max pooling is illustrated in Figure 10.

Since pooling reduces the size of the activation maps, it is a useful way to manage the computational complexity of the network. In addition, applying pooling makes the feature representations more invariant to small translations in the input. That is, the output remains almost the same when the input volume is slightly shifted.

However, it has been argued that applying pooling can result in losing valuable

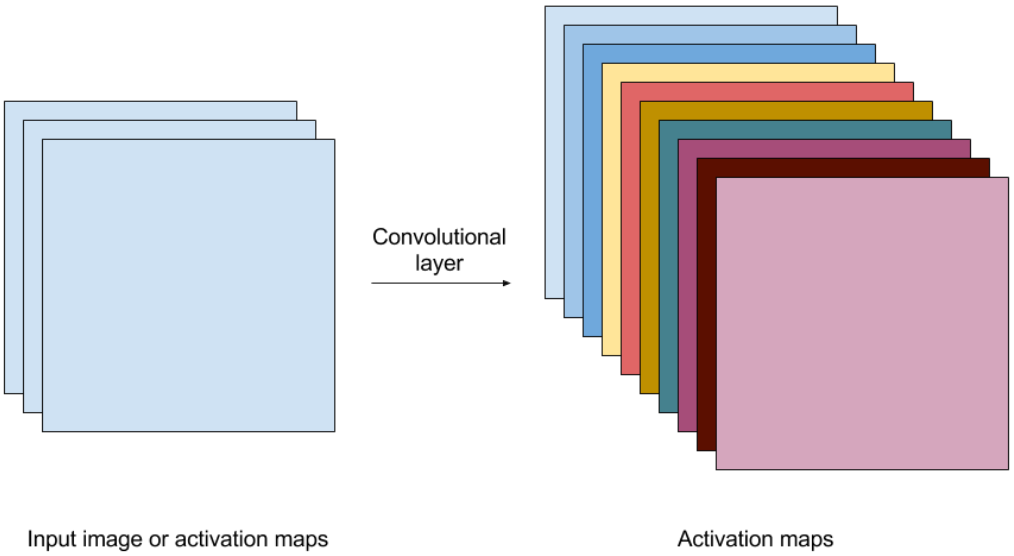


Figure 8: A convolutional layer.

spatial information. Therefore, in many convolutional neural networks, pooling layers are simply replaced by convolutional layers with increased stride and this results in no loss in accuracy [64]. In this thesis, all the neural networks we use do not contain pooling layers.

Typically, fully connected layers are used at the end of a convolutional neural network, i.e., they are the final hidden layers. In principle, they are the same as layers in the aforementioned traditional MLPs, which connect every neuron in the previous layer to every neuron in the next layer. There are also convolutional networks without fully connected layers which are called fully convolutional networks (FCNs) [41]. Unlike traditional convolutional neural networks, they are able to manage different input sizes.

Activation functions are used between layers of a CNN to introduce nonlinearity. The rectified linear unit (ReLU) [50], which was first proposed for restricted Boltzmann machines, is currently the most commonly used activation function for convolutional neural networks. The main advantage of ReLU is that it can alleviate the vanishing gradient problem. Formally, ReLU is defined as:

$$f(x) = \max(0, x) \tag{4}$$

In this thesis, another activation function exponential linear unit (ELU) [8] is also used. It speeds up the learning in deep neural networks and has improved learning

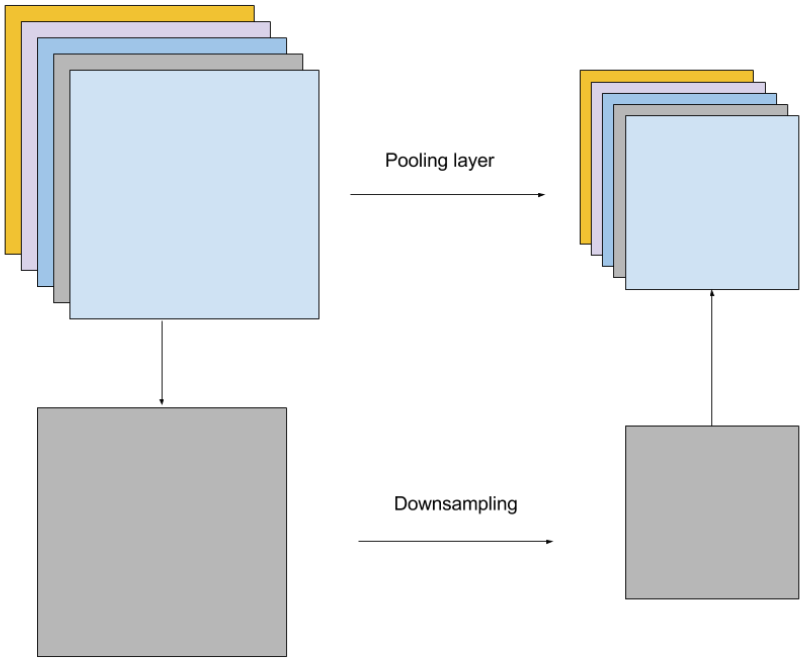


Figure 9: A pooling layer.

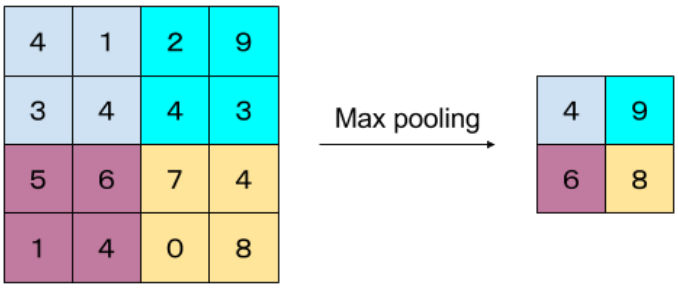


Figure 10: Max pooling with 2×2 filter and stride 2.

characteristics compared to other activation functions. ELU is defined as:

$$f(x) = \begin{cases} x & x > 0 \\ \alpha(\exp(x) - 1) & x \leq 0 \end{cases} \tag{5}$$

where $\alpha > 0$. ReLU and ELU are illustrated in Figure 11.

An example of a traditional convolutional neural network is illustrated in Figure

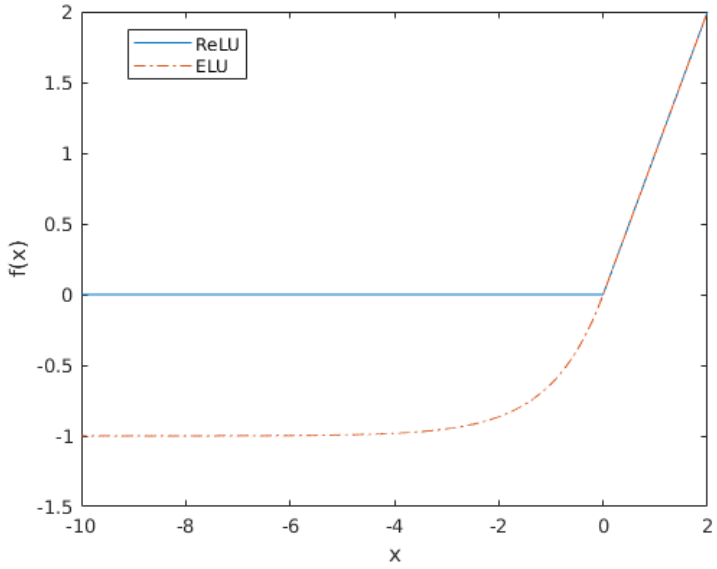


Figure 11: The rectified linear unit (ReLU) and the exponential linear unit (ELU, $\alpha = 1.0$).

12. It is composed of a sequence of convolutional layers combined with pooling layers and one fully connected layer at the end.

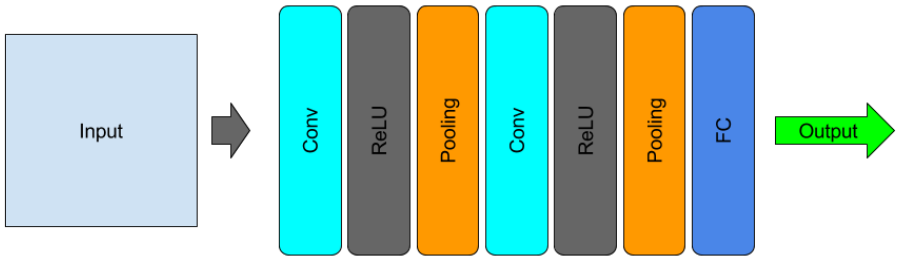


Figure 12: An example of a convolutional neural network.

2.3.4 Fully Convolutional Networks

A fully convolutional network (FCN) [41] is a variant of the traditional convolutional neural network where all the learnable layers are convolutional, and thus it does not include any fully connected layers. By building the CNN fully convolutional with

upsampling layers inside the network, FCN could be applied to input of arbitrary size and output pixel-wise predictions efficiently. Note that the upsampling layers are not always needed, e.g., due to the use of dilated convolutions [72].

FCN was initially proposed for semantic segmentation task. Semantic segmentation is the task of labeling each pixel in the image with one of the predetermined category labels. Its goal is to understand the image in pixel level. Before the existence of FCN, solving semantic segmentation with CNN was performed in a patch-based manner [7, 14]. As illustrated in Figure 13, in order to classify an image pixel, the traditional patch-based approaches use an image patch around a pixel together with the label of that pixel as a sample to train a CNN. This is known as patch-wise training. During test time, again an image patch is extracted around a pixel and fed into the CNN to produce a category label for that pixel. This is done for every pixel in the image to produce a dense final prediction. However, these approaches have several deficiencies. First, it requires much memory at test time. For example, if the size of the extracted image patch for each pixel is 40×40, the required amount of memory is 1600 times larger that of the original image. Besides, it is time-consuming and inefficient. Although the neighboring image patches are always overlapping, these approaches fail to reuse the shared features between patches, and this results in unnecessarily repeated computations. In addition, the size of the image patch limits the size of the receptive field. Typically, the size of the image patch is much smaller than the size of the entire image, and thus only local information without global context can be extracted, which affects the performance of the algorithm.

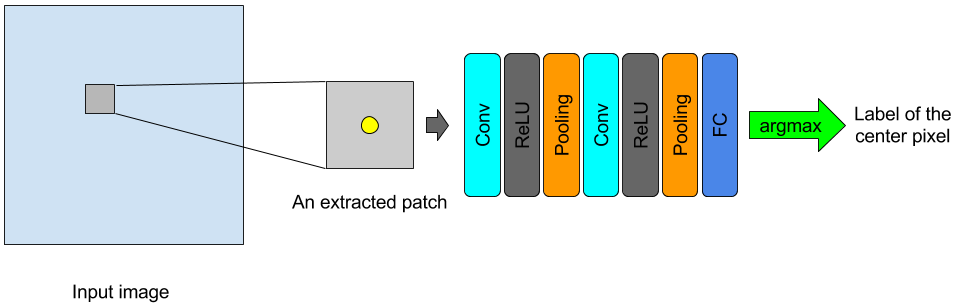


Figure 13: Patch-based sematic segmentation using CNN.

Unlike the patch-based approaches, FCN can be trained end-to-end simply using whole images and ground truth labels of the same size to make dense predictions for semantic segmentation efficiently at test time [41]. FCN and its variants have demonstrated a significant improvement in segmentation accuracy over traditional methods on standard datasets. Therefore, FCN has driven great breakthrough on semantic segmentation task and the idea of FCN has been successfully applied to other computer vision tasks.

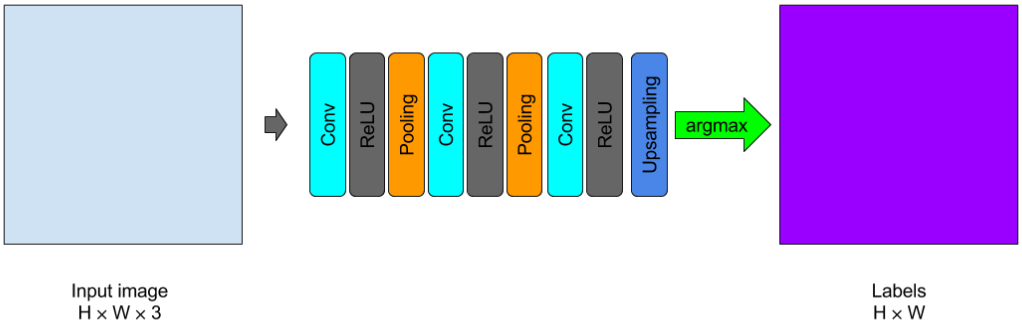


Figure 14: Fully convolutional network.

An example of the FCN structure is shown in Figure 14. Typically, a FCN-based model used for producing dense predictions is composed by successive convolutional layers with downsampling and upsampling inside the network. As discussed before, downsampling can be done by pooling layers or convolutional layers with larger stride value. In order to upsample the activation maps, upconvolution (sometimes called deconvolution) [52] and unpooling [73, 74] are introduced. Unpooling is the reverse operation of pooling which can enlarge the size of activation maps without learnable parameters. Similar to unpooling, upconvolution is the reverse operation of convolution. Here the reverse operation means the reverse of the forward and backward passes of convolution operation rather than the reverse of the convolutional effect (the mathematical deconvolution). In contrast to unpooling, the parameters of upconvolution can be learned. upconvolution and unpooling operations are illustrated in Figure 15.

2.3.5 Transfer Learning and Data Augmentation

Transfer learning is a machine learning technique for applying the knowledge gained during solving one problem to a related problem. Transfer learning with deep neural networks is usually performed by initializing the network with weights of a pre-trained network instead of random initialized ones and then finetuning the network [1, 53]. In practice, training an entire deep neural network from scratch with random initialization is usually not feasible, because of the insufficient size of a dataset. Even when the dataset is large enough, transfer learning can often help to boost generalization performance. Besides, training from scratch usually takes much more time. Therefore, transfer learning has become a common trend for training deep neural networks.

Typically, the weights are initialized from a classification network which is pre-trained on a very large dataset such as ImageNet [12]. Since the lower layers closer to the input learn to represent more generic features that are useful for almost all tasks,

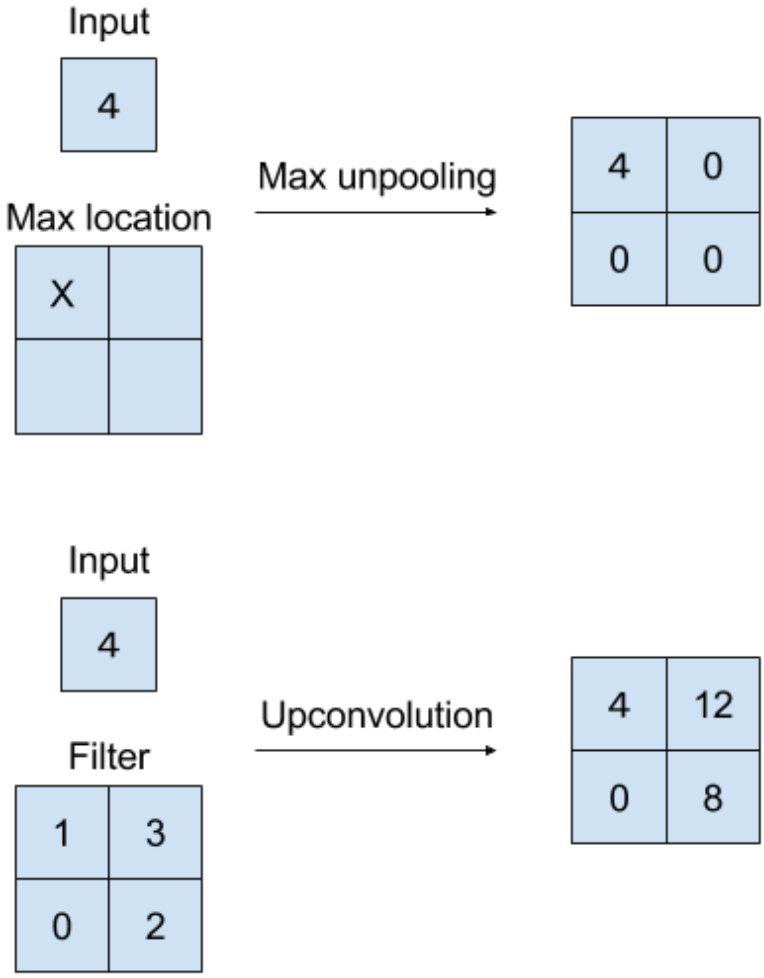


Figure 15: Upconvolution (deconvolution) and unpooling operations.

we usually freeze the weights of the lower layers and only finetune or retrain the higher layers of the network. It has been shown that transferring the learned representations of classification networks to solve other tasks such as semantic segmentation is often useful.

Data augmentation is a common machine learning technique that can improve the generalization capabilities of machine learning models by increasing the amount of training data, and thus avoid them from overfitting and increase their accuracy at test time [70]. It has been proven to be useful for training deep neural networks, especially on smaller datasets. Typically, data augmentation generates new samples from the existing data by applying transformations. For example, we can translate, rotate, warp, flip, scale, and crop the images. Even more realistic and representative samples can be generated synthetically.

2.3.6 VGGNet

VGGNet [62] is a deep convolutional neural network architecture proposed by the Visual Geometry Group (VGG) from the University of Oxford. It explores the relationship between the depth of the convolutional neural network and its performance. By increasing the depth to 16-19 layers, VGGNet shows significant improvement in accuracy. To reduce the number of parameters, only 3×3 filters are used in all convolutional layers. Compared to networks with fewer layers but larger filters, it has more non-linearities and fewer parameters. Two VGG based models VGG16 and VGG19 are shown in Figure 16. In the DSAC localization pipeline, the two CNNs are based on the VGGNet architecture.

2.3.7 DispNet

DispNet has been proposed in [47]. It is a fully convolutional neural network model with multi-scale predictions, which can be trained end-to-end to predict the dense disparity map from a pair of images. It adopts an encoder-decoder architecture where the encoder is used to compute abstract features and encode the global context, and the decoder recovers the original resolution by an expanding upconvolutional architecture [47]. Shortcut connections are added between the encoder and decoder to better preserve finer details of the input image. An example of the DispNet architecture is shown in Figure 17. In this thesis, our full-frame Coordinate CNN is based on the DispNet architecture.

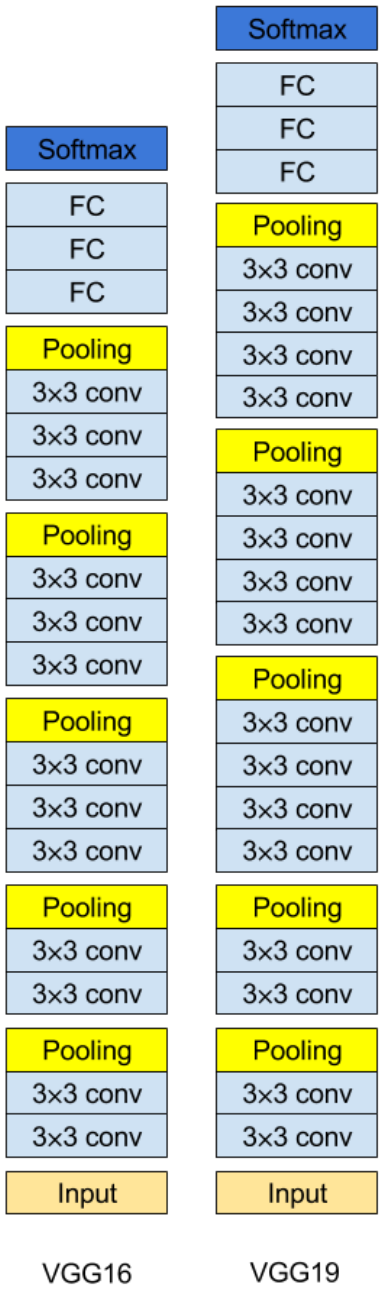


Figure 16: VGG16 and VGG19 architectures.

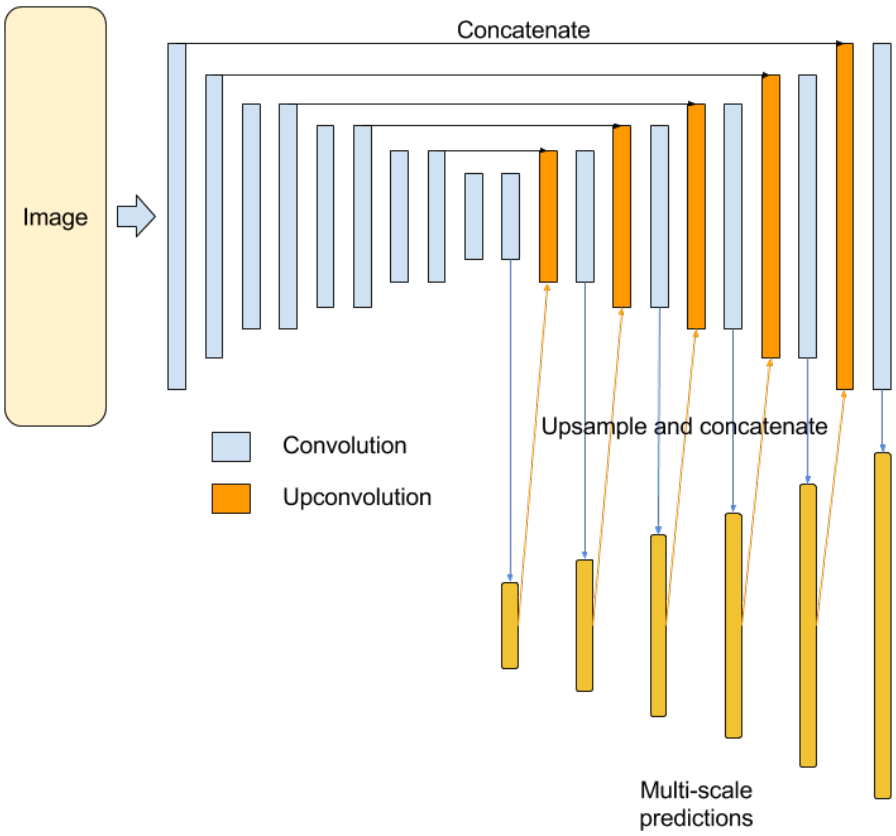


Figure 17: DispNet architecture.

3 Active Search Pipeline

In this chapter, we explain the state-of-the-art Active Search method [57, 58]. As described in Section 2.2.1, it is a traditional keypoint based localization approach which solves the localization task by correspondence search. In this thesis, we primarily focus on improving neural network based methods. However, as we compare the neural network based methods to Active Search, we describe its pipeline in detail.

Active Search consists of a powerful prioritized keypoint search pipeline which is able to efficiently and effectively generates 2D-3D matches for pose estimation. It is called Active Search because the key component of it is an algorithm which can actively search for additional matches to improve the registration performance.

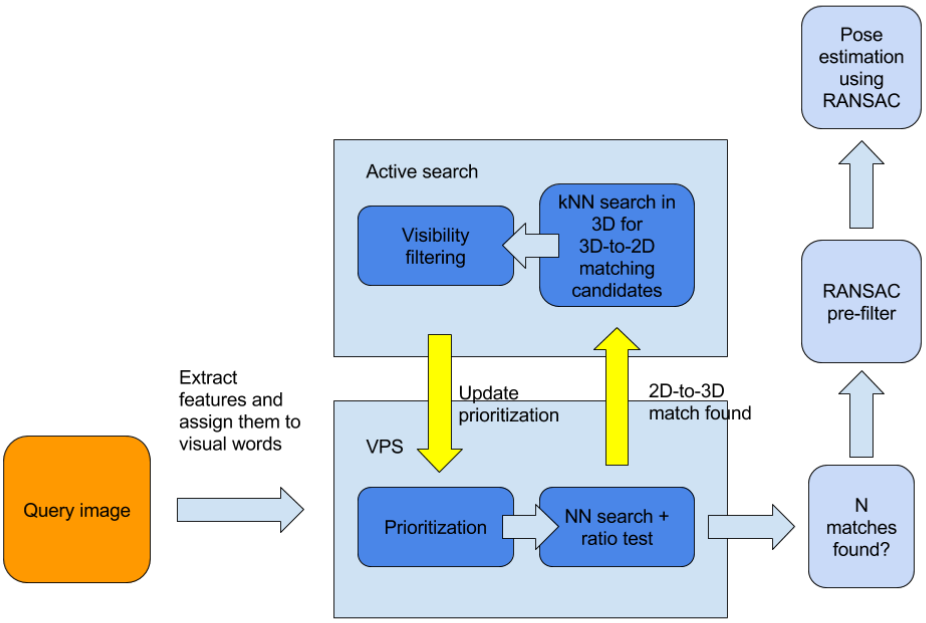


Figure 18: Active Search pipeline. Figure adapted from [58].

Active Search uses a visual vocabulary based prioritization scheme to accelerate the 2D-to-3D descriptor matching, where the SIFT [42] descriptors of the 3D models are clustered for faster indexing and estimating matching cost. Due to the quantization of the descriptor space, potential correct matches could be lost. Thus, a 3D-to-2D search scheme is used to reestablish these matches. Co-visibility of 3D points in the 3D model is further considered to speed up the matching. An overview of the Active Search pipeline is illustrated in Figure 18.

3.1 Vocabulary-Based Prioritized Search (VPS)

As discussed in Section 2.2.1, there are two types of matching schemes: indirect and direct matching. Many previous methods adopt indirect matching to obtain efficiency. Although an indirect way of feature matching is faster than direct 2D-to-3D matching, the quality of the established correspondences in this way is much lower. Contrary to previous methods, the Active Search pipeline demonstrates the power of direct 2D-to-3D matching. The FLANN library [49] can be used to perform the 2D-3D matching between the query image descriptors and 3D points in the scene model represented by a kd-tree. If the ratio test of two nearest neighbors with the threshold set to 0.7 is passed, a potential 2D-to-3D match is accepted [56]. It has been shown that the quality of the matches established by the direct 2D-to-3D matching is higher [56].

However, the direct search based on kd-tree is time-consuming. In order to perform fast localization while preserving the quality of the matches, a better search scheme is needed. In fact, tree-based search spends most of the time on features which do not result in correspondences [56]. Therefore, we could first consider the most promising features which could lead to a match during the search by prioritizing the search using a meaningful strategy. And once enough matches have been established, the search can be terminated. To address this issue, the vocabulary-based prioritized search (VPS) scheme is proposed. VPS uses a visual vocabulary to estimate the matching cost of features. First, the 3D points associated with SIFT descriptors are clustered and assigned to a visual vocabulary consisting of a predefined number of visual words. This step is done offline. During test time, the extracted image features are also assigned to this set of visual words. Then, for each image feature, only the 3D points with which the same visual word is associated are the candidates for the nearest neighbor search. In this way, the number of feature descriptors assigned to the visual word of an image feature is proportional to the cost of finding the nearest neighbors of this image feature. Thus, the number of descriptors can be a good estimate of the matching cost to prioritize the search. VPS first processes the image features with lower matching costs. That is, features assigned to words with fewer points are evaluated first. The search stops when enough correspondences have been found. An illustration of VPS is shown in Figure 19.

3.2 Active Correspondence Search

Although the use of prioritization scheme results in far more efficient direct 2D-to-3D matching, VPS is not as robust as the search based on kd-tree. This is because of the quantization effects introduced by the use of a visual vocabulary. That is, if corresponding features have different visual words, the correspondence could not be established during the search. Therefore, the number of high-quality correspondences is limited. Using soft assignments could be a solution to recover the matches, but at the cost of computational efficiency. In contrast, active correspondence search is able to achieve this efficiently.

Figure 20 illustrated the active correspondence search. The lost matches can

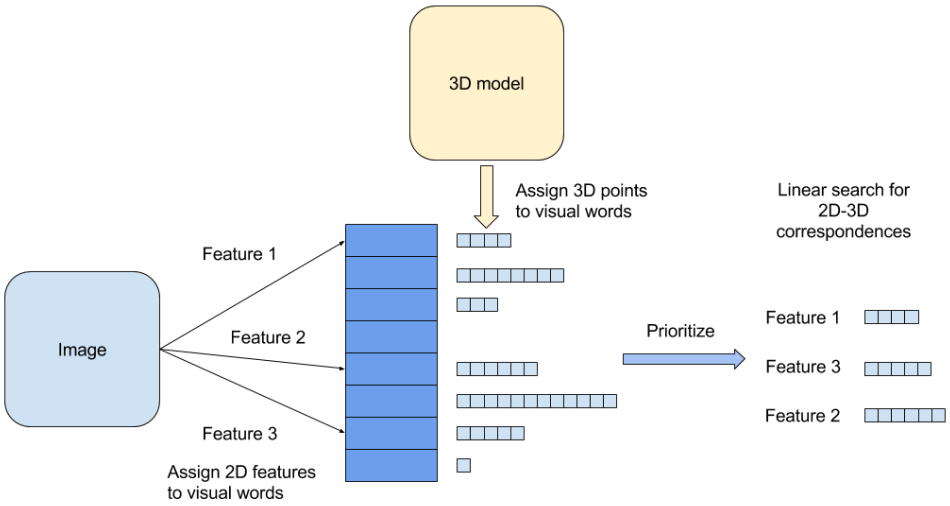


Figure 19: Vocabulary-based prioritized search(VPS). Figure adapted from [56].

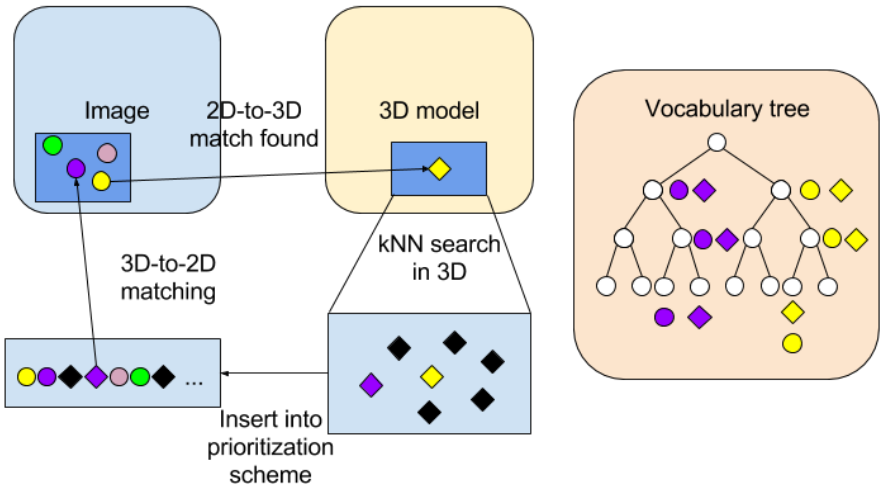


Figure 20: Active correspondence search. Figure adapted from [57].

be recovered by active correspondence search through 3D-to-2D matching. When a 2D-to-3D match has been found by VPS, we know that the nearby region of the 3D point could also be seen in the query image. Thus, we can consider finding matches for the neighboring 3D points via 3D-to-2D search. Similar to VPS, searching for the nearest neighbors of a 3D point can be again accelerated by using a visual vocabulary. Unlike the one used for 2D-to-3D matching, the vocabulary for 3D-to-2D matching is supposed to be coarser to guarantee that each word is associated with enough image features, since the number of features in the query image is tremendously fewer than

that of a 3D model. This coarse visual vocabulary can be obtained efficiently by extending the fine one used for 2D-to-3D matching to a hierarchical vocabulary tree and extracting the high-level words. Using such a coarser vocabulary also leads to fewer quantization effects such that some of the lost matches during the 2D-to-3D search can be recovered.

The 3D-to-2D matching is actively triggered, and it shares the same prioritization scheme with the 2D-to-3D matching. Once a 2D-to-3D match has been found, K nearest neighbors of the 3D point in 3D space are inserted to the prioritization scheme as candidates for 3D-to-2D matching. These 3D points are also prioritized by their matching costs, i.e., the numbers of features assigned to their visual words, together with other 2D features and 3D points that are already in the scheme. Once a 3D point is processed by the prioritization scheme, 3D-to-2D matching is performed rather than 2D-to-3D matching. Here, the same ratio test is used, and again the matching only considers the image features with the same visual word. Note that the search of K nearest neighbors of a 3D point in the 3D space is only performed when an 2D-to-3D match is accepted. There are also two other strategies to prioritize the search. One is to process all candidates for 3D-to-2D matching intermediately after a 2D-to-3D match is found. However, this can result in a large number of matches concentrated in a few parts of the image which should be avoided for pose estimation. The other one is to process all the 3D points only after all the 2D candidates are evaluated. In this way, active correspondence search could be useless, since enough correspondences can already be found before performing the 3D-to-2D matching.

3.3 Co-Visibility Information

Active correspondence search utilizes the assumption that the nearest neighbors of a found 3D point in the model can also be seen in the query image. However, this is not necessarily true. Therefore, the co-visibility information could be used to filter out unreliable neighboring points, and thus speed up the localization.

The co-visibility information can be approximated using the information obtained from the SfM pipeline. Since 3D points are generated from database images, two 3D points are unlikely to be seen at the same time in the query image if they are never seen together in a database image. Therefore, once the K nearest neighbors of a 3D point are found, only those have been co-visible in at least one database image are inserted into the prioritization scheme as candidates for 3D-to-2D matching. Other points are simply discarded. In this way, the correspondence search is accelerated.

The co-visibility information can also be used as a RANSAC pre-filter that can remove false matches before applying RANSAC. Once we have a set of 2D-3D matches ready to be fed into the RANSAC loop, we can connect two 3D points if they are co-visible in at least one database image. This results in a graph with multiple connected components, if we consider these 3D points as vertices of a graph. Points in different components should not be observed together in the query image. Therefore, only matches contained in the largest component should be used for pose estimation. By removing the false matches, the RANSAC-based pose estimation can be accelerated.

However, using only the database images results in merely an approximation to the true co-visibility information. Thus, the two filtering steps can also remove correct points and matches. In order to achieve better performance, cameras can be merged to obtain better and more continuous approximation of the true co-visibility relationship. For each image in the database, k images with the closest camera centers are found. Then the set of similar images are defined by the subset of these k images with relative orientation difference within 60°. This results in a set of image clusters. When determining the co-visibility of two points, the images clusters are used instead. That is, if two points can be found together in at least one image cluster, they are considered co-visible. As a result, however, the filtering steps become far more computationally expensive. In order to reduce the running time, we can select only a minimal set of image clusters that covers all the database images using a greedy set cover algorithm.

4 DSAC Pipeline

In this chapter, we discuss the neural network based DSAC localization pipeline [3] which is based on the SCoRF pipeline [61] explained in Section 2.2.3. The DSAC pipeline is the main focus of this thesis, as we present two modifications to the DSAC pipeline in the next chapter.

As mentioned in Section 2.2.3, the original SCoRF pipeline requires the depth information during test time, which makes it limited. Instead of using a random forest to predict the 3D coordinates from a combination of RGB and depth features, the DSAC pipeline adopts a powerful deep neural network which can be discriminative enough without using the depth information. This enables the camera localization from RGB-only images.

Figure 21 gives an overview of the DSAC pipeline. It consists of two stages, each of them containing a CNN. In the first stage, a Coordinate CNN is adopted to generate 2D-3D correspondences from a given RGB image. In the second stage, a differentiable RANSAC (DSAC) scheme is performed to determine the final pose estimate. Here several pose hypotheses are generated, and they are evaluated by a Score CNN. Due to the use of two CNNs and the differentiable RANSAC, the entire pipeline can be trained in an end-to-end manner.

Although the 6 DoF camera pose can also be directly regressed by a single CNN as demonstrated by PoseNet, the localization performance obtained in this way is inferior. Therefore, the intermediate step of generating 2D-to-3D correspondences contained in both the Active Search pipeline and the DSAC pipeline is critical for high-quality camera localization.

4.1 Differentiable RANSAC

Random sample consensus (RANSAC) algorithm [15] is one of the most famous tools in computer vision. It is a simple but powerful framework for model fitting when outliers are present. It is applicable to numerous problems in computer vision, such as pose estimation, camera calibration and 3D reconstruction, and often works well in practice.

RANSAC runs in a hypothesize-and-verify manner. Firstly, multiple hypotheses are generated by fitting models to randomly selected minimal subsets of data points. Then, these hypotheses are scored by how much they are consistent with the entire dataset. This is usually done by counting the inliers of each model, which fit the model within some error threshold. Finally, the hypothesis with the highest score is selected as the final output. Typically, an optional step can be performed to obtain a better estimation by refining the final output using all its inliers.

In the context of image-based localization, the RANSAC algorithm is usually used in combination with a PnP algorithm which can generate a camera pose estimation from a set of 2D-3D correspondences. A minimal subset of four correspondences is used to generate the pose in the case of known intrinsic parameters, and at least six correspondences are needed if intrinsic parameters are unknown. In this thesis, we consider the situation where the intrinsic parameters are known. Once a pose

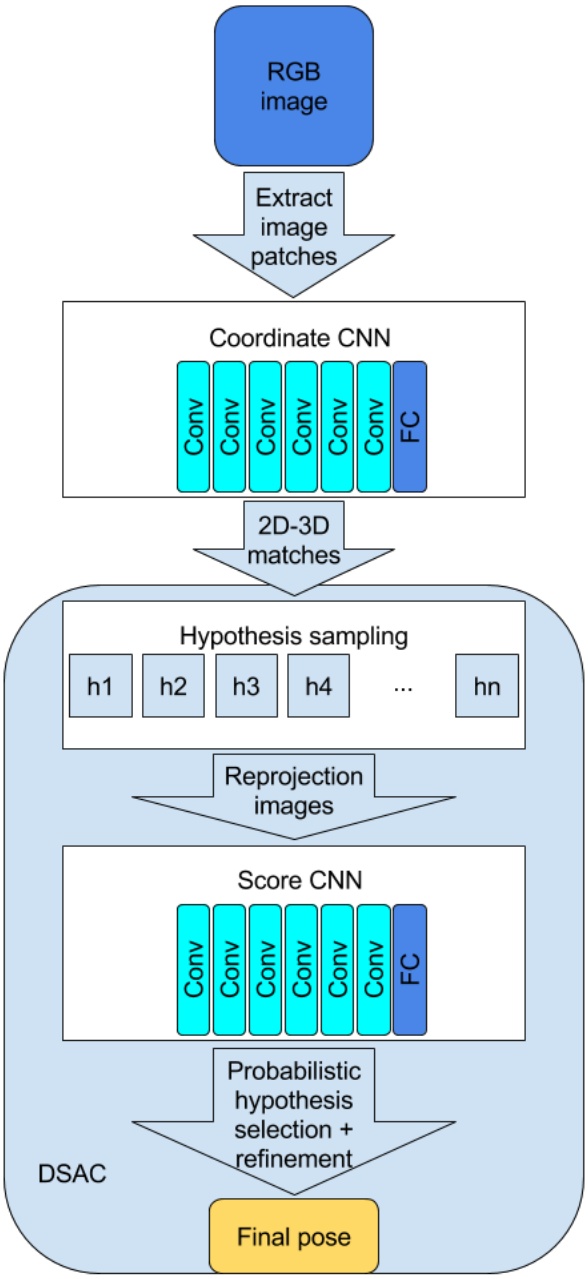


Figure 21: DSAC pipeline. Figure adapted from [3].

hypothesis is generated, a 3D point can be projected onto the image plane using the hypothesized camera pose and the known intrinsic parameters. The reprojection error is then defined by the distance between its corresponding 2D pixel and the reprojected one. A 2D-3D correspondence is considered as an inlier of a pose hypothesis if the reprojection error is less than some threshold. The selected hypothesis is the one with the highest number of inliers and it can be further refined.

However, the traditional RANSAC algorithm is not differentiable, thus cannot be integrated into an end-to-end deep learning pipeline. If we want to propose an image-based localization pipeline which contains the intermediate step of predicting 2D-to-3D correspondences and at the same time can be trained end-to-end by directly minimizing the localization error (e.g. training PoseNet), a differentiable RANSAC is needed.

In order to make the traditional RANSAC differentiable, a differentiable score function, e.g., a Score CNN (explained in the next section), can be used instead of counting inliers. More importantly, the non-differentiable argmax operator which select the best hypothesis with the highest score should be replaced by a differentiable one. There are two different ways to achieve this. The first way (SoftAM) is to use soft argmax instead of argmax which turns the hard selection into a weighted average of all hypotheses. However, this results in learning a good average of hypotheses instead of learning to select the best one. Inspired by the policy gradient approaches in reinforcement learning, the second way is to preserve the hard selection but make it probabilistic. It is called DSAC (Differentiable SAmple Consensus). It has been shown experimentally that DSAC is less sensitive to overfitting than the first option for image-based localization. An overview of the traditional RANSAC and the two differentiable variants are shown in Figure 22.

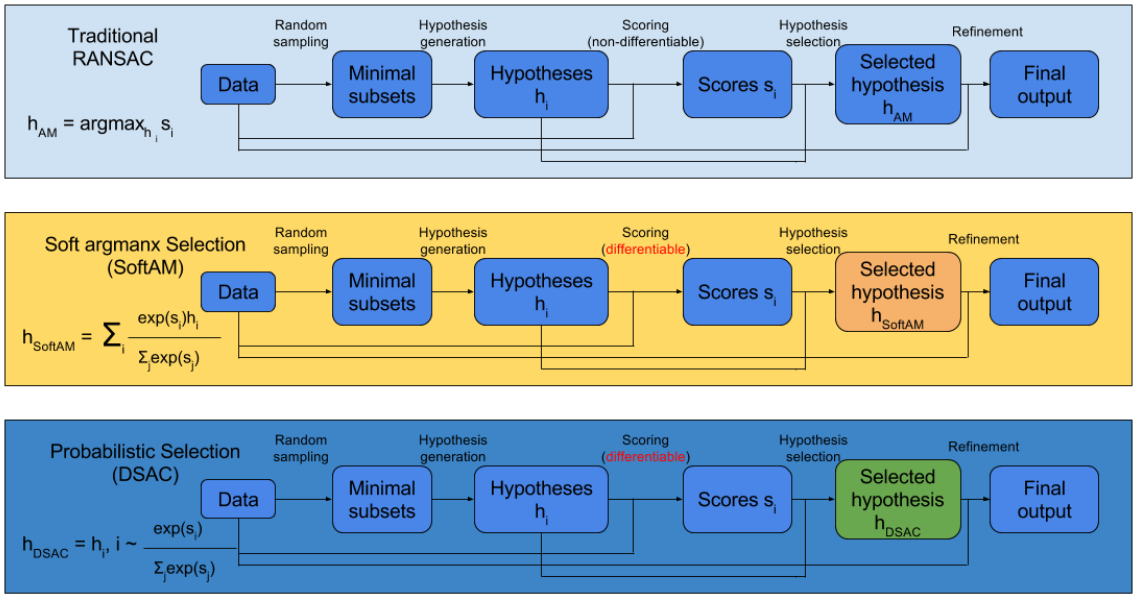


Figure 22: A graphical representation of the traditional RANSAC and two differentiable variants of it. Figure adapted from [3].

Using this proposed differentiable RANSAC together with a Coordinate CNN and a Score CNN (discussed in the next section), an end-to-end trainable image-based localization pipeline is enabled, although componentwise training for good initialization is needed.

4.2 Coordinate CNN and Score CNN

As already mentioned, the DSAC pipeline contains two separate CNNs, a Coordinate CNN and a Score CNN. The Coordinate CNN is used to predict 2D-3D correspondences, and the Score CNN is for scoring hypothesis. Both CNNs are based on the VGGNet architecture [62].

The Coordinate CNN generates the 2D-3D correspondences as follows. For a 2D pixel in the image, a 42×42 image patch centered at the pixel is cropped and fed into the Coordinate CNN, and a 3D scene coordinate estimate is generated by this CNN. The location of the 2D pixel in the image and the predicted 3D position in the world space form a 2D-3D correspondence. During test time, only 40×40 pixels per image are randomly sampled and processed to reduce the running time. The sampling is done by first dividing the image into 40×40 cells and choosing a random pixel location for each cell.

The Score CNN predicts a score for a hypothesis from a reprojection error image. Once a pose hypothesis is generated from a minimal subset of the 40×40 2D-3D correspondences using a PnP algorithm, we can calculate the reprojection error for each of the 40×40 correspondences using this hypothesis. This results in a 40×40 reprojection error image for the hypothesis. To assess the quality of the hypothesis, the reprojection error image is directly fed into the Score CNN.

The detailed configurations of the Coordinate CNN and the Score CNN are described in Table 1. All layers expect the output layers in both the Coordinate CNN and the Score CNN are followed by a ReLU non-linearity. Unless indicated, all the convolutional layers are zero-padded with 1-pixel border. Unlike the original VGGNet architecture, there are no pooling layers in the networks. Convolutional layers with stride equal to 2 are used instead for downsampling.

Training the two CNNs jointly in an end-to-end fashion from scratch quickly reaches a local minimum, thus making the procedure useless. Hence, training the two components separately to provide a good initialization for end-to-end training is necessary.

The Coordinate CNN can be trained directly by minimizing the following loss:

$$loss_{coord}(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\| \tag{6}$$

where $\mathbf{y}$ is the ground truth scene coordinate label of an image pixel and $\hat{\mathbf{y}}$ is the prediction of the Coordinate CNN. Here the Euclidean distance is used instead of the squared distance since it is more robust to outliers. The ground truth scene coordinates of a 2D pixel can be obtained using the 2D pixel coordinates, the depth value of the pixel and the camera intrinsic parameters. Hence, only image patches with valid depth information are sampled for training.

The Score CNN is trained using synthetically generated data. Given a training image, synthesized pose hypotheses can be generated by adding noise to the ground truth pose of this image. Then for each synthesized pose hypothesis, its reprojection error image can be computed using the predictions of the trained Coordinate CNN. Using the synthesized pose hypotheses and reprojection error images generated from training images and their ground truth poses, the Score CNN can be trained by

minimizing the following loss:

$$loss_{score}(\hat{s}, s) = |\hat{s} - s| \tag{7}$$

where $\hat{s}$ is the score predicted by the Score CNN and s is the ground truth score. To define the ground truth score, we need the loss between the ground truth pose and the pose hypothesis. It is given by:

$$loss_{pose}(\hat{\mathbf{h}}, \mathbf{h}) = \max(\angle(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}), \|\hat{\mathbf{t}} - \mathbf{t}\|) \tag{8}$$

where $\mathbf{h} = [\mathbf{t}, \boldsymbol{\theta}]$ and $\hat{\mathbf{h}} = [\hat{\mathbf{t}}, \hat{\boldsymbol{\theta}}]$ are ground truth pose and pose hypothesis respectively. The camera rotations $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$ are in axis-angle form and the angular distance between them are measured in degree. The camera translations $\hat{\mathbf{t}}$ and $\mathbf{t}$ are measured in cm. Then the ground truth score is defined as:

$$s = -\beta loss_{pose}(\hat{\mathbf{h}}, \mathbf{h}) \tag{9}$$

where β is a hyperparameter that controls the broadness of the distribution after applying softmax to the scores of hypotheses. As illustrated in Figure 22, this distribution is used for weights in SoftAM or as the sampling distribution in DSAC. It is set to 10 in the original DSAC paper. Training in this way, the Score CNN learns to predict small scores for hypotheses with large errors and large scores for hypotheses with small errors.

Coordinate CNN	Score CNN
$42 \times 42 \times 3$ RGB image patch	$40 \times 40 \times 1$ reprojection error image
3×3 conv, 3/64, $s = 1$, no padding	3×3 conv, 1/32, $s = 1$
3×3 conv, 64/64, $s = 2$	3×3 conv, 32/32, $s = 2$
3×3 conv, 64/128, $s = 1$	3×3 conv, 32/64, $s = 1$
3×3 conv, 128/128, $s = 2$	3×3 conv, 64/64, $s = 2$
3×3 conv, 128/256, $s = 1$	3×3 conv, 64/128, $s = 1$
3×3 conv, 256/256, $s = 1$	3×3 conv, 128/128, $s = 2$
3×3 conv, 256/256, $s = 2$	3×3 conv, 128/256, $s = 1$
3×3 conv, 256/512, $s = 1$	3×3 conv, 256/256, $s = 2$, no padding
3×3 conv, 512/512, $s = 1$	3×3 conv, 256/512, $s = 1$
3×3 conv, 512/512, $s = 2$, no padding	3×3 conv, 512/512, $s = 2$
FC 2048/4096	FC 512/1024
FC 4096/4096	FC 1024/1024
FC 4096/3	FC 1024/1

Table 1: Configurations of Coordinate CNN and Score CNN.

4.3 End-to-End Training

After initializing the Coordinate CNN and the Score CNN with componentwise training as described in the previous section, the entire DSAC pipeline can be trained

end-to-end. The differentiable loss of the entire pipeline is the loss between the ground truth camera pose and the estimated pose which can be computed according to Equation 8. However, some parts of the pipeline are still not directly differentiable. They are the pose hypothesis generation step using the PnP algorithm [35] and the final iterative pose refinement step.

For the pose estimation using the PnP algorithm, the derivatives can be calculated using central differences [17, 39]. The final refinement step is performed by iterating inlier sampling and pose estimation multiple times. Since it contains hard inlier selection operation, it is non-differentiable. However, because of the large number of inliers that are chosen, the refined poses typically change smoothly with regard to the scene coordinate prediction. Therefore, the derivatives can be again calculated using central differences by considering the refinement step as a whole. In order to make the derivatives more stable, refinement step is stopped if the number of inliers found is less than 50. In addition, to calculate the central differences efficiently, only 1% of the scene coordinates are sampled and the gradient is corrected by multiplying it by 100.

5 DSAC Variants

In this chapter, we present two modifications to the DSAC pipeline which is described in the previous chapter. First, we propose to discard the use of the Score CNN and the differentiable RANSAC, meaning that instead of first training the two CNNs separately and fine-tuning the entire pipeline end-to-end, we only train the Coordinate CNN to predict the 2D-3D correspondences and use the traditional RANSAC which is non-differentiable and contains no learnable parameters to generate the final poses. The second modification is that instead of performing scene coordinate regression in a patch-based manner, we propose to use a fully convolutional Coordinate CNN which takes the whole image as input and efficiently produce correspondingly-sized dense scene coordinate predictions.

5.1 Non-Differentiable RANSAC

As explained in Section 4.2, in the original DSAC pipeline, the Score CNN is used to regress the scores of hypotheses from their reprojection error images instead of directly scoring them by counting inliers. This makes the scoring function differentiable, and thus it enables the end-to-end training of the entire pipeline together with the differentiable hypothesis selection operation and the Coordinate CNN. However, while compared to the scoring scheme that uses only the number of inliers, the CNN-based score function has more discriminative power such that it can utilize more information from a reprojection error image to assess the quality of the hypothesis, it can easily overfit the training data. Furthermore, the componentwise training of the Score CNN is performed by using synthetically generated data. The pose hypotheses are generated by adding noise to the ground truth poses of training images, and the reprojection error images of these poses are generated using the scene coordinate predictions of the trained Coordinate CNN. Hence, the quality of the Score CNN is highly correlated with the quality of the Coordinate CNN and the density of training poses in the whole pose space. This can result in even more severe overfitting. For example, it is possible that during training the synthesized hypotheses with smaller errors have worse reprojection error images due to the quality of the Coordinate CNN, and thus the Score CNN learns to predict higher scores from worse reprojection error images. At test time, if the test image is similar to the training images, it will still work. However, if the test image is far from the training images, the generated hypotheses with better reprojection images which are actually the ones with smaller errors will not be recognized by the Score CNN and small scores will be given to them. Therefore, we argue that the use of the Score CNN can lead to worse performance for scenes with few training images or that are difficult for the Coordinate CNN (e.g. scenes contain repeated structures).

On the other hand, the conventional RANSAC has been already proven to be robust for many computer vision applications. Although the scoring using inlier count is simple and not as discriminative as a CNN-based score function, it can generalize well to different scenarios and is not sensitive to the size of training data and the quality of the Coordinate CNN. Therefore, we propose to use the conventional

RANSAC instead of the DSAC with the Scoring CNN, which means that we restore the use of inlier count and the argmax hypothesis selection. As a result, the entire localization pipeline is no longer end-to-end trainable. However, we argue that as long as the Coordinate CNN is accurate enough, training the entire pipeline in an end-to-end fashion is not necessary.

The detailed steps of the RANSAC algorithm used in this thesis are described in Algorithm 1. This is also applicable to the DSAC used in the original pipeline. For the conventional RANSAC, $score(H_i)$ is the number of inliers, i.e., the number of correspondences with reprojection errors less than a threshold τ, and the final pose H_f is the one with highest inlier count. For the DSAC, $score(H_i)$ is the prediction of the Score CNN, and the final pose H_f is selected randomly according to the softmax distribution of scores. In this thesis, 40×40 correspondences are generated using the Coordinate CNN ($N = 1600$) as mentioned before, 256 hypotheses are sampled per image ($K = 256$), the inlier threshold τ is set to 10 pixels, 8 refinement steps are performed ($R = 8$), in each refinement step at most 100 inliers are chosen ($P = 100$), and the refinement is stopped in case less than 50 inliers have been found ($Q = 50$).

Algorithm 1 Pseudocode for RANSAC optimization

```
evaluate the Coordinate CNN to obtain N correspondences
i ← 0
while i < K do
    sample 4 2D-3D correspondences
    generate a pose hypothesis h using PnP
    if all the 4 correspondences are inliers of h then
        H_i ← h
        S_i ← score(H_i)
        i ← i + 1
select the final pose H_f according to the scores
j ← 0
while j < R do
    sample at most P inliers
    if less than Q inliers are found then
        break
    recalculate the pose H_f from the inliers
return pose H_f
```

5.2 Full-Frame Coordinate CNN

In the original DSAC pipeline, besides the differentiable RANSAC and the Score CNN, the Coordinate CNN is one of the most important components that enable the entire pipeline to achieve great localization performance. As described in the previous section, we propose to discard the use of differentiable RANSAC and the only learnable part left is the Coordinate CNN. Therefore, the performance of the pipeline critically depends on the quality of the Coordinate CNN.

The Coordinate CNN used in the original DSAC pipeline is trained and evaluated in a patch-based manner. Similar to the patch-based approaches for semantic segmentation, patches need to be sampled during both training and test time. The patch-based Coordinate CNN is trained using image patches of fixed size (42×42) centered around pixels and the corresponding world coordinates of these pixels. At test time, patches of the same size are used as inputs of the Coordinate CNN to generate 2D-3D correspondences. However, obtaining a set of 2D-3D correspondences in such a way is very inefficient. In the DSAC pipeline, 1600 correspondences are needed for pose estimation. Thus 1600 patches should be extracted and fed into the deep neural network. This is obviously time-consuming and also requires much memory. Moreover, the fixed patch size limits the size of the receptive field, and thus limits the information that can be processed by the network. This makes it difficult for the Coordinate CNN to predict accurate 2D-3D correspondences when the scene exhibits ambiguities (e.g. repeated structures), since the global context cannot be used by the network. Besides, the size of the patch can be considered as a hyperparameter of the network and it is not trivial to select the best value for it. It is difficult to distinguish between different patches if the size is set too small. On the other hand, although large patches contain more information and are more representative, they capture too many spatial dependencies such that patches centered at the same 3D point observed from only slightly different viewpoints can look extremely different. Thus, using large patch size is prone to overfitting. In addition, the use of fixed patch size at both training and test is problematic. For example, if the training images are taken near an object but the test images are taken from afar, the difference in scale can cause the Coordinate CNN to fail. And it is not trivial to design an adaptive patch size selecting scheme.

To overcome such limitations, we propose to use a CNN which accepts a whole image as input and produces scene coordinate predictions for all pixels in the image. This is inspired by recently presented architectures for solving per-pixel tasks, such as semantic segmentation, depth estimation, and disparity estimation. Performing coordinate regression in this way, patches are not required to be sampled and the 2D-3D correspondences can be generated efficiently at test time. Since the whole image is used as input, more context is added to the regression process and more overall information is considered. Meanwhile, the patch size selection is no longer a problem. However, such a network structure is more sensitive to the spatial correlation of image pixels and thus is prone to overfitting. We propose to use data augmentation to mitigate the problem. We call our network full-frame Coordinate CNN.

5.2.1 Network Architecture

Our full-frame Coordinate CNN is based on the architecture of DispNet which is described in Section 2.3.7. It is fully convolutional such that dense per-pixel scene coordinate predictions can be generated from arbitrary-sized input images. We call the outputs scene coordinate images. The network consists of a contractive part and an expanding part. The input image is first spatially compressed via the contractive part and then refined via the expanding part. Moreover, shortcut connections are

added in between to overcome the data bottleneck. Unlike DispNet, there is only one final output layer at the end of the network and no multi-scale side predictions are used. Instead of ReLU, we use ELU for the nonlinearity between layers. The details of our network architecture are described in Table 2. Layers start with upconv are upconvolutional layers and all other layers are convolutional. Also for each layer, the size of the kernel (filter), the stride, the number of input channels, the number of output channels, and the input (+ is a concatenation) are given.

Name	Kernel	Str.	Ch I/O	Input
conv1a	7×7	2	3/32	image
conv1b	7×7	1	32/32	conv1a
conv2a	5×5	2	32/64	conv1b
conv2b	5×5	1	64/64	conv2a
conv3a	3×3	2	64/128	conv2b
conv3b	3×3	1	128/128	conv3a
conv4a	3×3	2	128/256	conv3b
conv4b	3×3	1	256/256	conv4a
conv5a	3×3	2	256/512	conv4b
conv5b	3×3	1	512/512	conv5a
conv6a	3×3	2	512/512	conv5b
conv6b	3×3	1	512/512	conv6a
conv7a	3×3	2	512/512	conv6b
conv7b	3×3	1	512/512	conv7a
upconv6	3×3	2	512/512	conv7b
iconv6	3×3	1	1024/512	upconv6+conv6b
upconv5	3×3	2	512/512	iconv6
iconv5	3×3	1	1024/512	upconv5+conv5b
upconv4	3×3	2	512/256	iconv5
iconv4	3×3	1	512/256	upconv4+conv4b
upconv3	3×3	2	256/128	iconv4
iconv3	3×3	1	256/128	upconv3+conv3b
upconv2	3×3	2	128/64	iconv3
iconv2	3×3	1	128/64	upconv2+conv2b
upconv1	3×3	2	64/32	iconv2
iconv1	3×3	1	64/32	upconv1+conv1b
upconv0	3×3	2	32/16	iconv1
iconv0	3×3	1	16/16	upconv0
coord_pred	3×3	1	16/3	iconv0

Table 2: Our network architecture.

5.2.2 Training Loss

Similar to the patch-based Coordinate CNN used in the original DSAC pipeline, we train our full-frame Coordinate CNN by minimizing the Euclidean distance between

the scene coordinate ground truth and the prediction as given in Equation 6. However, unlike training in a patch-based manner where patches and single predictions are used and only patches centered at pixels with valid depth values are sampled for training, our network uses a pair of color image and scene coordinate image (dense scene coordinate values) as a training sample where the ground truth scene coordinates can be missing for some pixels. Here we simply ignore the pixels without the ground truth scene coordinates and mask out their contributions to the final loss. More formally, loss for a training sample can be written as:

$$loss = \frac{1}{\sum_{i,j} \mathbf{M}_{ij}} \sum_{i,j} \mathbf{M}_{ij} \|\hat{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij}\| \tag{10}$$

where $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ are ground truth scene coordinate image and scene coordinate image prediction respectively, $\mathbf{M}$ is a mask and (i, j) is a 2D pixel coordinate. The mask $\mathbf{M}$ has the same resolution as the color image and the scene coordinate image. A pixel of $\mathbf{M}$ is set to 1 if the corresponding scene coordinate ground truth exists and 0 otherwise. Invalid pixels of the scene coordinate image $\mathbf{Y}$ are all set to 0.

5.2.3 Data Augmentation

While our network has larger receptive field and can encode more global context information for better understanding of the scene, it is more sensitive to the spatial relation between the neighbor pixels in the color images compared to a network trained and tested in patch-based manner where only the information from small patches is processed and the predictions of different patches are not necessarily correlated. Therefore, we need more training images taken from a lot of different viewpoints to regularize the network. Unfortunately, a common problem for image-based localization is that the training poses in the whole pose space is not dense enough to regularize such a powerful network. Also, image-based localization is typically performed scene-wise, which means that only the images belonging to a particular scene can be used during training. It does not make sense to use data belonging to other scenes, and unlike for other computer vision tasks such as semantic segmentation, it is not straightforward to apply transfer learning technique for image-based localization. Hence, overfitting can be a serious problem in training our full-frame Coordinate CNN for solving image-based localization.

To alleviate the overfitting problem, we propose to use data augmentation to generate more data for training. A simple and common way of doing data augmentation in the context of solving per-pixel tasks is to apply 2D affine geometric transformations to the training images. The transformations we perform include translation, rotation, and scaling.

However, these 2D affine transformations do not always preserve the real geometric relations between pixels in an image. Since the camera is in the 3D space and the 3D points are projected onto the image plane according to a camera model, if the camera pose is changed, the transformation between the old and new 2D coordinates of 3D points is much more complex than the combination of 2D transformations in the image plane. Therefore, we propose a second way to augment data. For a

training sample, we add a random transformation to its pose to synthesize a new camera pose, and then using this new camera pose and scene coordinates of the pixels, we project the pixels to the new camera plane to generate a set of new color and coordinate images.

6 Experiments and Results

In this chapter, we describe the experiments conducted to assess the performance of the methods explained in the previous chapters and present the results of these experiments. We first introduce the 7-Scenes dataset [20] used for experiments. Next, we reproduce the results of the Active Search method and the DSAC method. After that, we evaluate the proposed modifications to the original DASC pipeline and report the quantitative comparisons.

6.1 Dataset 7-Scenes

For the experiments, we use the 7-Scenes dataset [20] provided by Microsoft Research. The 7-Scenes dataset is a widely used RGB-D dataset which consists of seven different indoor environments. The RGB-D images are captured using a handheld Kinect camera at 640×480 resolution and associated with 6 DoF ground truth camera poses obtained via the KinectFusion system. A dense 3D model is also available for each scene. Each scene contains multiple sequences of tracked RGB-D camera frames and these sequences are split into training and testing data. The dataset exhibits several challenges, namely motion blur, illumination changes, textureless surfaces, repeating structures (e.g. in the Stairs dataset), reflections (e.g. in the Redkitchen dataset), and sensor noise. Table 3 shows an overview of the dataset and Figure 23 illustrates the scenes. Figure 24 gives a number of example RGB images.

Scene	Training images	Test images	Spatial extent
Chess	4000	2000	$3 \times 2 \times 1$m
Fire	2000	2000	$2.5 \times 1 \times 1$m
Heads	1000	1000	$2 \times 0.5 \times 1$m
Office	6000	4000	$2.5 \times 2 \times 1.5$m
Pumpkin	4000	2000	$2.5 \times 2 \times 1$m
Redkitchen	7000	5000	$4 \times 3 \times 1.5$m
Stairs	2000	1000	$2.5 \times 2 \times 1.5$m

Table 3: The 7-Scenes dataset.

6.2 Reproducing Active Search Results

To benchmark the performance of the novel neural network based DSAC pipeline, we need to know the performance of the state-of-the-art traditional keypoint based approaches. In this section, we present the experiments to reproduce the results of Active Search on the 7-Scenes dataset reported in [69].

6.2.1 Implementation Details

Our Active Search implementation is based on an existing open-source but relatively old implementation[1] [57]. We use the default recommended parameter settings for our experiments except the number of the visual vocabulary words. An overview of the settings is given in Table 4.

Number of visual words	1000
Type of 3D point representation	Integer mean per visual word (cf. [56])
Number of nearest neighbors in 3D used as candidates for 3D-to-2D search	200
Use the point filter?	Yes
Use the RANSAC pre-filter?	Yes
Group cameras?	Yes
Number of camera per cluster	10
Terminate the correspondence search after finding N matches	N=100

Table 4: Overview of the parameter settings.

To use Active search, a 3D point cloud model reconstructed from images, where each point is associated with SIFT descriptors is needed. However, the 7-Scenes dataset does not contain the required 3D models. Thus, we use Colmap [59] to reconstruct the models from the training images. Note that only the color images are used for reconstruction, the depth images and the ground truth poses are not needed during reconstruction. The sparse reconstruction pipeline of Colmap consists of three steps, namely the feature extraction step, the feature matching step, and the reconstruction step. For feature extraction, we use the vocabulary tree matching mode with the pre-trained vocabulary tree 256K. All the parameters used are set to default. After the reconstruction finished, we register the models against the ground truth poses of the training images using the geo-registration function of Colmap.

The visual vocabulary containing 1000 words used by Active Search is trained using the descriptors extracted from the training images of the 7-Scenes dataset. We train the vocabulary by running the mini batch K-means algorithm [60] on all descriptors extracted from a subset of all the training images (all 7 scenes), i.e., one out of every ten images in a sequence is used.

Following [57], for RANSAC pose estimation, we use the 6-point DLT algorithm [24]. The inlier threshold is set to $\sqrt{10}$ pixels. Following common practice [40], if the final pose found by RANSAC has less than 12 inliers, the localization of a query image is considered to be failed.

[1]Source code available at https://www.graphics.rwth-aachen.de/software/image-localization.

6.2.2 Results

The reconstructed 3D models and the camera tracks of the training sequences are shown in Figure 25. However, we observe large registration errors when registering the reconstructed models against the ground truth poses of the training images. The mean and median registration errors are given in Table 5.

Following [69], we report the median localization errors and the number of images that are not successfully localized. Our results together with the results from [69] are reported in Table 6. We notice that our reproduced results are worse than the original results. This is because the performance of Active Search highly depends on the quality of the reconstructed model. In [69], the median registration errors reported are at most 5cm, which means that their reconstructed models are more accurate. We are unable to reproduce the same models since the details for model reconstruction are not explained in [69]. In addition, a more advanced implementation of Active Search [58] is used in [69] which is not publicly available.

To provide quantitative comparisons between Active Search and DSAC, we also report the percentage of query images for which the camera pose error is below 5° and 5cm which is used as the test metric in [3]. Besides, for more detailed comparisons, we also provide the 0.75 and 0.95 quantiles of localization error, the percentage of query images for which the camera pose error is below 5° and 10cm, and the percentage of query images for which the camera pose error is below 5° and 20cm. The more detailed localization performance is described in Table 7. Figure 28 shows cumulative histograms (normalized) of localization errors.

Scene	Mean	Median
Chess	0.040m	0.036m
Fire	0.024m	0.022m
Heads	0.019m	0.018m
Office	0.095m	0.084m
Pumpkin	0.091m	0.080m
Redkitchen	0.068m	0.060m
Stairs	0.078m	0.066m

Table 5: Mean and median registration errors.

Scene	Ours	[69]
Chess	0.07m, 2.50° (0)	0.04m, 1.96° (0)
Fire	0.06m, 2.26° (1)	0.03m, 1.53° (1)
Heads	0.05m, 3.72° (2)	0.02m, 1.45° (1)
Office	0.14m, 3.26° (24)	0.09m, 3.61° (34)
Pumpkin	0.16m, 3.41° (9)	0.08m, 3.10° (71)
Redkitchen	0.13m, 3.98° (0)	0.07m, 3.37° (0)
Stairs	0.17m, 4.12° (0)	0.03m, 2.22° (3)

Table 6: The reproduced results and the results from [69].

Scene	0.75 quantile	0.95 quantile	5°, 5cm	5°, 10cm	5°, 20cm
Chess	0.12m, 4.25°	0.26m, 9.11°	30.5%	63.9%	79.2%
Fire	0.10m, 3.62°	0.25m, 7.44°	39.5%	71.1%	83.3%
Heads	0.13m, 6.14°	0.66m, 17.76°	42.4%	57.3%	63.6%
Office	0.21m, 5.49°	0.62m, 12.57°	6.5%	27.2%	56.4%
Pumpkin	0.26m, 5.52°	0.59m, 12.88°	4.9%	23.4%	51.5%
Redkitchen	0.21m, 6.33°	0.49m, 12.99°	8.7%	30.3%	52.7%
Stairs	0.30m, 7.09°	0.63m, 17.43°	6.1%	20.7%	41.1%

Table 7: Localization performance of Active Search.

6.3 Reproducing DSAC results

In this section, we describe the experiments to reproduce the results of DSAC on the 7-Scenes dataset reported in [3].

6.3.1 Implementation Details

Our implementation is based on the original DSAC implementation[2]. The network architectures are already presented in Section 4.2.

Following [3], for the componentwise training of the Coordinate CNN, we randomly sample 100 training images to form a training round and 512 patches are extracted from each image. These patches are used to train the Coordinate CNN with a batch size of 64. Once all the sampled patches are processed, we begin a new round. We use the Adam optimizer [34] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ for updating the weights. The initial learning rate is set to 0.0001 and is halved every 50k updates until the end. The CNN is trained from scratch for a total of 300k updates.

For the componentwise training of the Score CNN, we again randomly sample 100 training images to form a training round. For each image, 16 synthesized pose hypotheses are sampled by adding noise to the ground truth pose and their reprojection images are used for training. The noise is added by applying a random transformation to the ground truth pose and the transformation is either large or small. For large transformation, the translational components in mm are sampled from a Gaussian with mean 0 and standard deviation 20, the rotational axis is uniformly sampled, and the rotational angle in radian is sampled from a Gaussian with mean 0 and standard deviation 20. For small transformation, the standard deviation is set to 1. The Score CNN is trained with a batch size of 64 and the same Adam optimizer is used. The learning rate is fixed to 0.0001 and the CNN is trained from scratch for 80 training rounds in total, i.e., 2k updates.

For end-to-end training, fixed learning rates of 10^{-5} and 10^{-7} are used for the Coordinate CNN and for the Score CNN respectively. Instead of the Adam optimizer, the stochastic gradient descent with momentum [55] of 0.9 is used, and all gradients are clipped to the range of -0.1 to 0.1. One randomly sampled image is processed in each training round and the CNNs are trained for 5k training rounds (5k updates).

[2]Source code available at https://github.com/cvlab-dresden/DSAC.

The details of the pose estimation (RANSAC optimization) is explained in Section 5.1 and Algorithm 1. For the PnP algorithm in the RANSAC optimization, we use the implementations available in OpenCV [5] (P3P for hypothesis sampling and ITERATIVE for pose refinement). A pinhole camera model with $f_x = 525$, $f_y = 525$, $c_x = 320$, $c_y = 240$ without distortion coefficients is used.

Note that the RGB and depth images in the 7-Scene dataset are not registered. Therefore, we need to manually calibrate the RGB and depth camera and register the images. The intrinsic parameters of depth camera we use for the registration are $f_x = 585$, $f_y = 585$, $c_x = 320$, $c_y = 240$. The registered depth images together with the ground truth poses are then used to compute the ground truth scene coordinate images for the componentwise training of the Coordinate CNN. An example of the coordinate image is shown in Figure 26, where the scene coordinates are mapped to the RGB values. During test time, only color images are used.

6.3.2 Results

The reproduced results are reported in Table 8 and compared with the original results, where the numbers indicate the percentages of test images for which the error is below 5° and 5cm. We successfully achieve almost the same results as reported in [3] for all the 7 scenes.

Scene	Ours	[3]
Chess	94.5%	94.6%
Fire	75.7%	74.3%
Heads	72.8%	71.7%
Office	71.5%	71.2%
Pumpkin	54.0%	53.6%
Redkitchen	50.5%	51.2%
Stairs	4.1%	4.5%

Table 8: The reproduced results and the original results for DSAC.

Again, for extensive quantitative comparisons, we report the numbers for other metrics. They are summarized in Table 9. We can then compare the localization performance of DSAC with Active Search.

Scene	Median	0.75 quantile	0.95 quantile	5°, 10cm	5°, 20cm
Chess	0.021m, 0.72°	0.031m, 1.06°	0.051m, 1.72°	99.0%	99.2%
Fire	0.028m, 1.00°	0.049m, 1.91°	0.229m, 10.04°	88.5%	91.5%
Heads	0.020m, 1.31°	0.061m, 4.00°	0.704m, 46.63°	76.5%	76.9%
Office	0.034m, 1.02°	0.053m, 1.56°	0.113m, 3.22°	93.3%	97.0%
Pumpkin	0.047m, 1.33°	0.080m, 2.38°	0.516m, 10.26°	81.0%	89.9%
Redkitchen	0.050m, 1.45°	0.074m, 2.13°	0.185m, 5.77°	86.8%	93.8%
Stairs	1.91m, 49.40°	2.86m, 99.21°	6.11m, 142.2°	8.7%	10.2%

Table 9: Localization performance of DSAC.

Scene	Hourglass-pose
Chess	0.15m, 6.17°
Fire	0.27m, 10.48°
Heads	0.19m, 11.63°
Office	0.21m, 8.48°
Pumpkin	0.25m, 7.01°
Redkitchen	0.27m, 10.15°
Stairs	0.29m, 12.46°

Table 10: The median localization errors for Hourglass-Pose [48].

By comparing the reported numbers, we can see that DSAC outperforms Active Search on almost all the 7 scenes expect Stairs. However, for Heads, DSAC has much worse 0.95 quantile of the rotational error, meaning that Active Search can better localize the most difficult testing frames. Since Heads contains the least amount of training images, it suggests that the DSAC pipeline is prone to overfit when training images are not enough while the non-learnable Active Search pipeline can generalize well. For Stairs, we see that DSAC fails to demonstrate a reasonable localization performance. Note that Stairs is difficult since it contains a large number of repeating structures (the stairs).

For comparison with another type of neural network based methods, namely PoseNet and its variants which directly regress the camera pose from a whole image, we present the results for Hourglass-Pose in Table 10. Hourglass-Pose is one of the PoseNet Variants that achieve the best results. We see that its overall performance is far below DSAC and Active Search. However, it provides reasonable localization performance for Stairs.

6.4 DSAC Variants

In this section, we examine our proposed modifications to the original DSAC pipeline via experiments on the 7-Scenes dataset. As explained in Chapter 5, the first variant is made by substituting the DSAC pose optimization part which contains the Score CNN with the traditional RANSAC which scores the hypotheses by inlier counting. Based on the first variant, the second one is made by using our full-frame Coordinate CNN instead of the patch-based one. We report the results and quantitative comparisons are made between the two variants and the original pipeline.

6.4.1 Implementation Details

The first variant is implemented simply by changing the Score CNN to an inlier counter and making the final pose selection operation deterministic, i.e., using argmax instead of probabilistic selection (see Section 5.1 and Algorithm 1 for detailed information). The same parameter settings for the componentwise training of the Coordinate CNN are adopted. Since the pipeline is no longer end-to-end trainable due to the use of non-differentiable traditional RANSAC, we do not perform end-to-end training.

The second variant contains the novel full-frame Coordinate CNN (see Section 5.2.1 and Table 2 for detailed network architecture). We train our network from scratch for 800 epochs with a batch size of 16 (i.e., 200k updates for Chess and so on) using the Adam optimizer where $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The loss is computed as described in Section 5.2.2. The initial learning rate is set to 0.0001 and is halved every 200 epochs until the end.

As mentioned in Section 5.2.3, we perform data augmentation online during network training. We perform the 2D affine transformation with a 40% chance, perform the '3D' transformation with a 50% chance, and use the original image with a 10% chance. Note that images in the same batch can be augmented in different ways. For the 2D transformation, we uniformly sample translation from the range $[-20\%, 20\%]$ of the image width and height for x and y respectively, sample rotation from $[-45°, 45°]$ and sample scaling from $[0.7, 1.5]$. For the '3D' transformation, we uniformly sample the rotational axis and the rotational angle is uniformly sampled from $[0°, 60°]$. The direction of the translation vector is again uniformly sampled, and its magnitude in mm is sampled from $[0, 200]$. Figure 27 shows an example of the data augmentation.

At test time, although our full-frame Coordinate CNN can directly generate 640×480 scene coordinate predictions, we only use 40×40 of the predictions for pose estimate to make it consistent with the patch-based Coordinate CNN.

6.4.2 Results

We refer to the two DSAC variants as DSAC-V1 and DSAC-V2 respectively. The detailed localization performance of DSAC-V1 is summarized in Table 11, Table 12, and Figure 28. According to the results, the overall performance of DSAC-V1 is better than the original DSAC. As we can see, while achieving almost the same results on the easy frames, DSAC-V1 has superior performance on the harder ones, i.e., DSAC-V1 provides better 0.95 quantiles. For example, for Heads, DSAC-V1 reduces the 0.95 quantile of the translational error by 36.4% and reduces the 0.95 quantile of the rotational error by 32.3%. Besides, for scenes that have fewer training images (Fire, Heads), DSAC-V1 provides better 0.75 quantiles, and more test frames are localized with the error less than 5° and 5cm. More importantly, DSAC-V1 is able to produce reasonable median localization error for Stairs, though the accuracy on the hardest frames of Stairs is still terrible. The results of DSAC-V1 verify that the use of traditional RANSAC makes the entire localization pipeline more robust and suggest that the Score CNN can easily overfit the training data.

We present the results for DSAC-V2 in Table 13, Table 14, and Figure 28. As we can see, its overall localization performance is extremely robust. Specifically, the use of our novel full-frame Coordinate CNN significantly improves the performance on the hardest frames (overall smaller 0.95 quantiles). In addition, while DSAC and DSAC-V1 often have extreme localization errors, i.e., rotational errors close to 180° and rotational errors larger than 2-5m, the maximum errors of DSAC-V2 are always reasonable (see Figure 28). However, we also observe a slightly degraded performance on the easiest frames (e.g. Chess, Fire and Heads) but the reason is not

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.021m, 0.69°	0.029m, 1.01°	0.050m, 1.68°
Fire	0.026m, 0.95°	0.043m, 1.76°	0.111m, 4.70°
Heads	0.017m, 1.15°	0.034m, 2.42°	0.448m, 31.58°
Office	0.036m, 1.01°	0.055m, 1.54°	0.110m, 2.89°
Pumpkin	0.050m, 1.34°	0.082m, 2.32°	0.417m, 6.51°
Redkitchen	0.052m, 1.53°	0.076m, 2.20°	0.152m, 5.04°
Stairs	0.112m, 2.87°	0.422m, 10.01°	1.29m, 30.80°

Table 11: Localization performance of DSAC-V1, part 1.

Scene	5°, 5cm	5°, 10cm	5°, 20cm
Chess	94.9%	99.0%	99.1%
Fire	79.5%	93.9%	95.5%
Heads	82.1%	86.7%	86.9%
Office	70.0%	93.9%	97.9%
Pumpkin	49.9%	81.2%	91.1%
Redkitchen	47.2%	86.2%	94.7%
Stairs	27.4%	47.3%	55.8%

Table 12: Localization performance of DSAC-V1, part 2.

clear. Remarkably, DSAC-V2 is able to provide the best localization performance for Stairs compared with Active Search, DSAC, and DSAC-V1. Even the hardest frames can be localized with the error less than 0.41m and 13°. It shows that the full-frame Coordinate CNN with the enlarged respective field can better cope with the repetitive structures while the patch-based Coordinate CNN and the local feature (SIFT) based Active Search are limited due to their local nature.

To show that our full-frame Coordinate CNN is more efficient at test time, we present the runtimes of the CNNs in Table 15. We see that our full-frame Coordinate CNN is one order of magnitude faster than the patch-based one. And this comparison is even not fair since our full-frame Coordinate CNN produces 640×480 predictions while only 40×40 are generated by the patch-based one.

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.024m, 0.82°	0.037m, 1.26°	0.064m, 2.27°
Fire	0.037m, 1.40°	0.068m, 2.65°	0.102m, 4.04°
Heads	0.024m, 1.73°	0.049m, 3.76°	0.123m, 8.53°
Office	0.035m, 1.01°	0.052m, 1.53°	0.099m, 2.72°
Pumpkin	0.049m, 1.29°	0.088m, 2.23°	0.325m, 5.09°
Redkitchen	0.042m, 1.21°	0.061m, 1.74°	0.099m, 2.85°
Stairs	0.079m, 2.13°	0.142m, 3.12°	0.371m, 5.00°

Table 13: Localization performance of DSAC-V2, part 1.

Following [46], in addition to the direct measure of localization performance, we

Scene	5°, 5cm	5°, 10cm	5°, 20cm
Chess	88.5%	98.1%	99.7%
Fire	62.3%	94.2%	98.7%
Heads	75.1%	85.4%	86.2%
Office	73.1%	95.0%	99.2%
Pumpkin	51.4%	77.7%	91.4%
Redkitchen	60.4%	95.2%	98.2%
Stairs	29.5%	61.8%	83.1%

Table 14: Localization performance of DSAC-V2, part 2.

GPU	Full-frame	Patch-based
NVIDIA GeForce GT 750M	∼0.3s	∼5s
NVIDIA GeForce GTX 1080	∼0.02s	∼0.3s

Table 15: The runtimes of the full-frame and patch-based Coordinate CNNs.

present the accuracy of the intermediate scene coordinate prediction on the test images. In table 16, we report the percentage of scene coordinate inliers and the mean Euclidean distance between the inliers and their ground truth scene coordinate labels. A prediction is considered as an inlier if its Euclidean distance to its ground truth label is less than 10mm. The normalized histograms of scene coordinate errors are illustrated in Figure 29.

As we can see, the end-to-end training does not have much effect on the overall accuracy of the patch-based Coordinate CNN, as the curves for DASC and DSAC-V1 are almost identical. Interestingly, our full-frame Coordinate CNN is able to produce significantly better scene coordinate predictions for all 7 scenes. This shows why it is more robust than the patch-based one. However, this does not directly lead to equally better localization accuracy. This is because the RANSAC-based optimizer is highly robust and non-deterministic [46].

Scene	DSAC	DSAC-V1	DSAC-V2
Chess	76.5%, 32.85mm	77.0%, 32.60mm	94.5%, 21,77mm
Fire	61.2%, 34.77mm	63.1%, 34.44mm	91.8%, 26.20mm
Heads	57.6%, 27.38mm	58.0%, 27.10mm	87.8%, 22.59mm
Office	59.0%, 44.75mm	61.5%, 44.07mm	93.5%, 27.34mm
Pumpkin	58.0%, 42.55mm	59.1%, 41.75mm	85.0%, 30.40mm
Redkitchen	60.8%, 45.64mm	61.3%, 44.68mm	92.8%, 31.54mm
Stairs	20.5%, 46.96mm	20.8%, 46.30mm	65.9%, 35.13mm

Table 16: The percentage of the scene coordinate prediction inliers and the mean errors of the inliers.

6.5 Effectiveness of Data Augmentation

In this section, we evaluate the effectiveness of data augmentation. We first report the performance of DSAC-V2 without the proposed data augmentation. We then add data augmentation to DSAC-V1 and see whether in this way the performance can be improved.

6.5.1 Full-Frame Coordinate CNN without Data Augmentation

The performance of DSAC-V2 without data augmentation is reported in Table 17, Table 18 and Figure 28. We refer to it as DSAC-V2-noaug. Note except the data augmentation, the same settings for training are used.

According to the results, training the full-frame Coordinate CNN without data augmentation dramatically degenerate the localization performance on all the scenes. Without data augmentation, the full-frame Coordinate CNN is too flexible to generalize well. This proves the importance of data augmentation which is used for regularizing our full-frame Coordinate CNN during training.

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.059m, 1.87°	0.100m, 3.23°	0.188m, 6.26°
Fire	0.132m, 4.49°	0.211m, 7.18°	0.622m, 36.87°
Heads	0.126m, 9.36°	0.239m, 15.23°	0.984m, 69.85°
Office	0.070m, 2.03°	0.119m, 3.51°	0.420m, 9.65°
Pumpkin	0.105m, 2.73°	0.193m, 4.69°	1.035m, 16.25°
Redkitchen	0.061m, 1.65°	0.095m, 2.59°	0.584m, 14.97°
Stairs	0.434m, 10.07°	0.705m, 16.00°	1.732m, 52.06°

Table 17: Localization performance of DSAC-V2 without data augmentation, part 1.

Scene	5°, 5cm	5°, 10cm	5°, 20cm
Chess	43.2%	75.0%	90.1%
Fire	10.7%	36.5%	53.9%
Heads	14.1%	19.9%	20.6%
Office	30.6%	67.8%	85.5%
Pumpkin	18.5%	47.6%	73.8%
Redkitchen	38.8%	76.7%	87.4%
Stairs	0.2%	1.8%	5.4%

Table 18: Localization performance of DSAC-V2 without data augmentation, part 2.

6.5.2 Patch-Based Coordinate CNN with Data Augmentation

Data augmentation can often lead to improved performance as already discussed. Since the patch-based Coordinate CNN is not trained using data augmentation, it is interesting to see how well it will perform if trained with data augmentation. We

adopt the same training settings and perform the 2D affine transformation to the training images before sampling the patches. We uniformly sample rotation from $[-45°, 45°]$ and scaling from $[0.7, 1.5]$.

We report the results in Table 19, Table 20 and Figure 28. We refer to it as DSAC-V1-aug. As we can see, training the patch-based Coordinate CNN with data augmentation does not lead to improved overall performance. Although for Fire and Stairs better performance on hard frames is achieved, for other scenes we even observe decreased accuracy. This suggests that the patch-based Coordinate CNN is not capable or discriminative enough to fit the augmented training data well.

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.025m, 0.85°	0.035m, 1.28°	0.058m, 2.06°
Fire	0.035m, 1.23°	0.053m, 2.00°	0.096m, 3.64°
Heads	0.018m, 1.31°	0.048m, 3.00°	0.663m, 34.70°
Office	0.046m, 1.24°	0.069m, 1.86°	0.160m, 4.17°
Pumpkin	0.055m, 1.43°	0.087m, 2.51°	0.770m, 17.88°
Redkitchen	0.061m, 1.68°	0.090m, 2.53°	0.160m, 5.14°
Stairs	0.111m, 2.87°	0.317m, 6.81°	0.852m, 17.17°

Table 19: Localization performance of DSAC-V1 with data augmentation, part 1.

Scene	5°, 5cm	5°, 10cm	5°, 20cm
Chess	92.3%	98.5%	99.1%
Fire	71.6%	95.5%	98.0%
Heads	75.4%	80.3%	80.8%
Office	55.8%	89.0%	95.6%
Pumpkin	44.5%	78.3%	86.9%
Redkitchen	36.0%	81.4%	94.3%
Stairs	25.3%	47.4%	63.0%

Table 20: Localization performance of DSAC-V1 with data augmentation, part 2.

6.6 Performance on Training Images

In practice, a good image-based localization system should localize well not only the images taken from novel viewpoints but also those already seen during training. Thus, it is also important to evaluate the localization performance on training images. In this section, we report the localization performance of Active Search, DSAC, DSAC-V1 and DSAC-V2 on the training images. The results are presented in Table 21, 22, 23, 24, 25, 26, 27, 28, and Figure 30. Note that Active Search is able to successfully register all the training images. Since DSAC and its variants are extremely accurate on training images, we also report the maximum translational and rotational errors for them.

Compared to its performance on the test images, Active Search does not show significantly improved performance on the training images. This is because Active Search is based on hand-crafted local feature detector and descriptor and formulates the correspondence search as a descriptor matching problem. On the one hand, the carefully designed feature detector and descriptor, and the matching scheme bring Active Search excellent generalization property. On the other hand, these non-trainable components prevent the Active Search to be optimized on a specific set of data. Moreover, since both the SfM pipeline for model reconstruction and the localization pipeline itself highly rely on the descriptor matching, limitations of both the feature detector and the feature descriptor can largely affect the final localization performance.

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.069m, 2.25°	0.122m, 3.86°	0.410m, 10.17°
Fire	0.041m, 1.71°	0.066m, 2.58°	0.128m, 4.46°
Heads	0.046m, 2.88°	0.077m, 4.51°	0.182m, 8.87°
Office	0.116m, 2.92°	0.177m, 4.91°	0.365m, 9.93°
Pumpkin	0.138m, 3.03°	0.207m, 4.58°	0.390m, 8.79°
Redkitchen	0.100m, 3.49°	0.161m, 5.65°	0.438m, 12.39°
Stairs	0.102m, 2.82°	0.150m, 4.55°	0.277m, 9.42°

Table 21: Localization performance of Active Search on training images, part 1.

Scene	5°, 5cm	5°, 10cm	5°, 20cm
Chess	31.0%	62.7%	77.5%
Fire	60.7%	87.7%	96.0%
Heads	51.8%	73.4%	79.7%
Office	10.3%	36.3%	66.1%
Pumpkin	6.3%	29.6%	63.8%
Redkitchen	14.1%	43.0%	62.3%
Stairs	18.1%	44.1%	73.1%

Table 22: Localization performance of Active Search on training images, part 2.

Unlike Active Search, we see that the learning based DSAC pipeline and its variants achieve dramatically improved accuracy on the training images. This is due to the flexibility of deep neural networks that allows them to fit the data well, although they are not trained to directly regress the camera pose. According to the results, we see that DSAC and DSAC-V1 show almost identical performance on the training images. Since the end-to-end training does not have much effect on the overall accuracy of the Coordinate CNN as already discussed, this suggests that the use of the powerful Score CNN does not result in a performance better than a simple inlier counting strategy can provide. Thus, we believe that in the DSAC pipeline, an accurate Coordinate CNN is more important.

As we can see there are still a number training images that are failed to be localized with the error below 5° and 5cm by the patch-based DSAC and DSAC-V1. Examples of the most difficult training images are shown in Figure 31. We see the difficulty is mainly caused by either motion blur or textureless surfaces. Although some images do have textured surfaces without motion blur, a large number of missing ground truth scene coordinate labels may also cause problems. This shows the limitations of the patch-based Coordinate CNN. Interestingly, both DSAC and DSAC-V1 perform well on the training frames of Stairs. However, the poor performance on the test images shows that the patch-based Coordinate CNN cannot generalize well in the existence of a large number of repetitive structures.

Again, our full-frame Coordinate CNN is able to localize well the most difficult training frames. The results show that DSAC-V2 significantly outperforms all the other three methods on the training images.

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.012m, 0.42°	0.019m, 0.60°	0.033m, 0.99°
Fire	0.010m, 0.35°	0.014m, 0.51°	0.024m, 0.88°
Heads	0.006m, 0.37°	0.008m, 0.53°	0.012m, 0.81°
Office	0.021m, 0.68°	0.030m, 0.92°	0.052m, 1.49°
Pumpkin	0.020m, 0.58°	0.033m, 0.88°	0.070m, 1.56°
Redkitchen	0.022m, 0.72°	0.032m, 1.03°	0.056m, 1.69°
Stairs	0.013m, 0.39°	0.018m, 0.54°	0.029m, 0.80°

Table 23: Localization performance of DSAC on training images, part 1.

Scene	Max	5°, 5cm	5°, 10cm	5°, 20cm
Chess	0.187m, 18.57°	98.3%	99.2%	99.3%
Fire	0.049m, 1.85°	100.0%	100.0%	100.0%
Heads	0.024m, 2.38°	100.0%	100.0%	100.0%
Office	0.209m, 3.61°	94.3%	99.4%	99.97%
Pumpkin	0.474m, 8.15°	88.8%	98.2%	99.3%
Redkitchen	0.253m, 11.19°	92.8%	99.2%	99.9%
Stairs	0.062m, 1.66°	99.9%	100.0%	100.0%

Table 24: Localization performance of DSAC on training images, part 2.

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.013m, 0.41°	0.018m, 0.58°	0.032m, 0.95°
Fire	0.009m, 0.33°	0.014m, 0.46°	0.021m, 0.78°
Heads	0.006m, 0.31°	0.008m, 0.44°	0.011m, 0.70°
Office	0.021m, 0.63°	0.031m, 0.88°	0.054m, 1.46°
Pumpkin	0.021m, 0.57°	0.035m, 0.88°	0.072m, 1.51°
Redkitchen	0.022m, 0.71°	0.034m, 1.06°	0.060m, 1.81°
Stairs	0.013m, 0.37°	0.019m, 0.51°	0.029m, 0.79°

Table 25: Localization performance of DSAC-V1 on training images, part 1.

Scene	Max	5°, 5cm	5°, 10cm	5°, 20cm
Chess	0.204m, 18.60°	98.6%	99.2%	99.3%
Fire	0.054m, 1.96°	99.95%	100.0%	100.0%
Heads	0.025m, 2.31°	100.0%	100.0%	100.0%
Office	0.227m, 3.85°	93.7%	99.2%	99.93%
Pumpkin	0.384m, 7.49°	87.4%	97.9%	99.2%
Redkitchen	0.186m, 7.82°	91.2%	99.1%	99.8%
Stairs	0.063m, 1.86°	99.8%	100.0%	100.0%

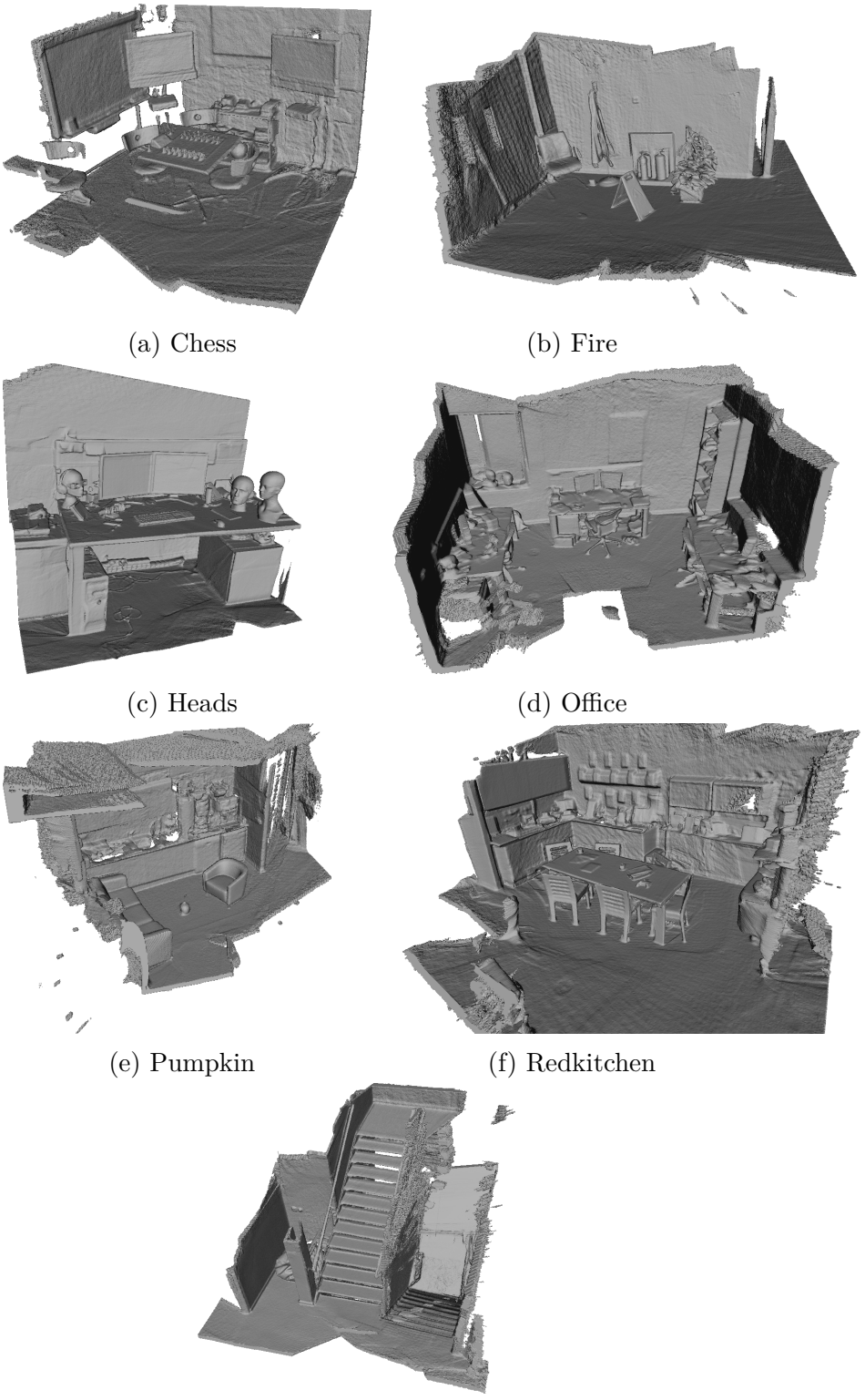
Table 26: Localization performance of DSAC-V1 on training images, part 2.

Scene	Median	0.75 quantile	0.95 quantile
Chess	0.009m, 0.24°	0.013m, 0.35°	0.020m, 0.64°
Fire	0.007m, 0.22°	0.009m, 0.30°	0.014m, 0.47°
Heads	0.006m, 0.34°	0.008m, 0.49°	0.012m, 0.78°
Office	0.009m, 0.28°	0.012m, 0.38°	0.020m, 0.58°
Pumpkin	0.009m, 0.20°	0.013m, 0.31°	0.023m, 0.51°
Redkitchen	0.008m, 0.23°	0.011m, 0.32°	0.017m, 0.54°
Stairs	0.010m, 0.31°	0.013m, 0.41°	0.020m, 0.61°

Table 27: Localization performance of DSAC-V2 on training images, part 1.

Scene	Max	5°, 5cm	5°, 10cm	5°, 20cm
Chess	0.044m, 4.25°	100.0%	100.0%	100.0%
Fire	0.026m, 1.69°	100.0%	100.0%	100.0%
Heads	0.022m, 1.90°	100.0%	100.0%	100.0%
Office	0.063m, 1.84°	99.97%	100.0%	100.0%
Pumpkin	0.052m, 1.64°	99.95%	100.0%	100.0%
Redkitchen	0.054m, 2.80°	99.97%	100.0%	100.0%
Stairs	0.032m, 1.38°	100.0%	100.0%	100.0%

Table 28: Localization performance of DSAC-V2 on training images, part 2.



(a) Chess

(b) Fire

(c) Heads

(d) Office

(e) Pumpkin

(f) Redkitchen

(g) Stairs

Figure 23: Dense reconstructions.

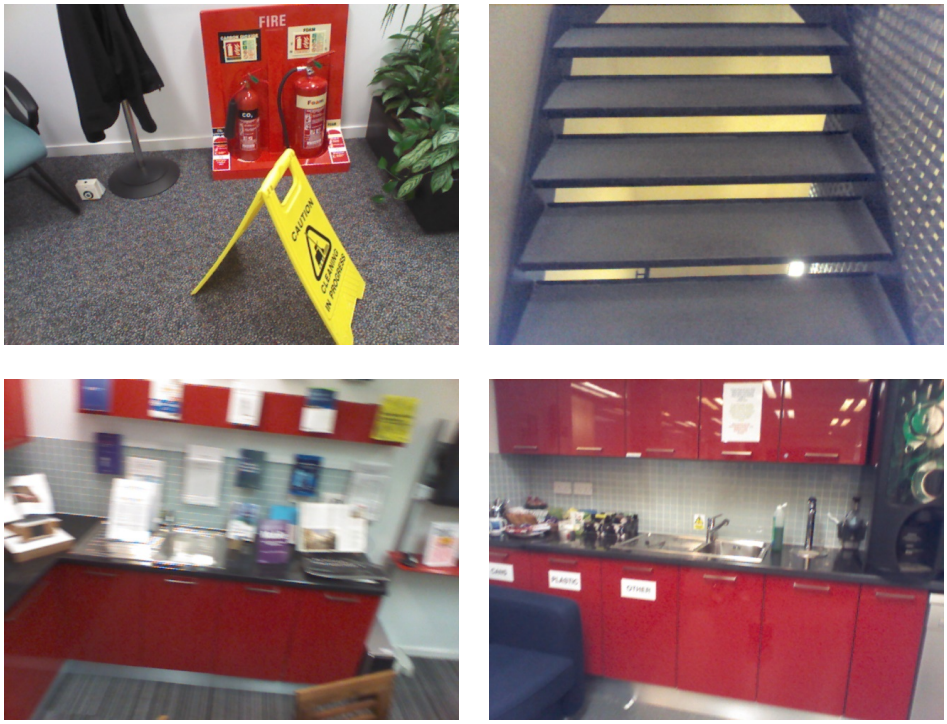
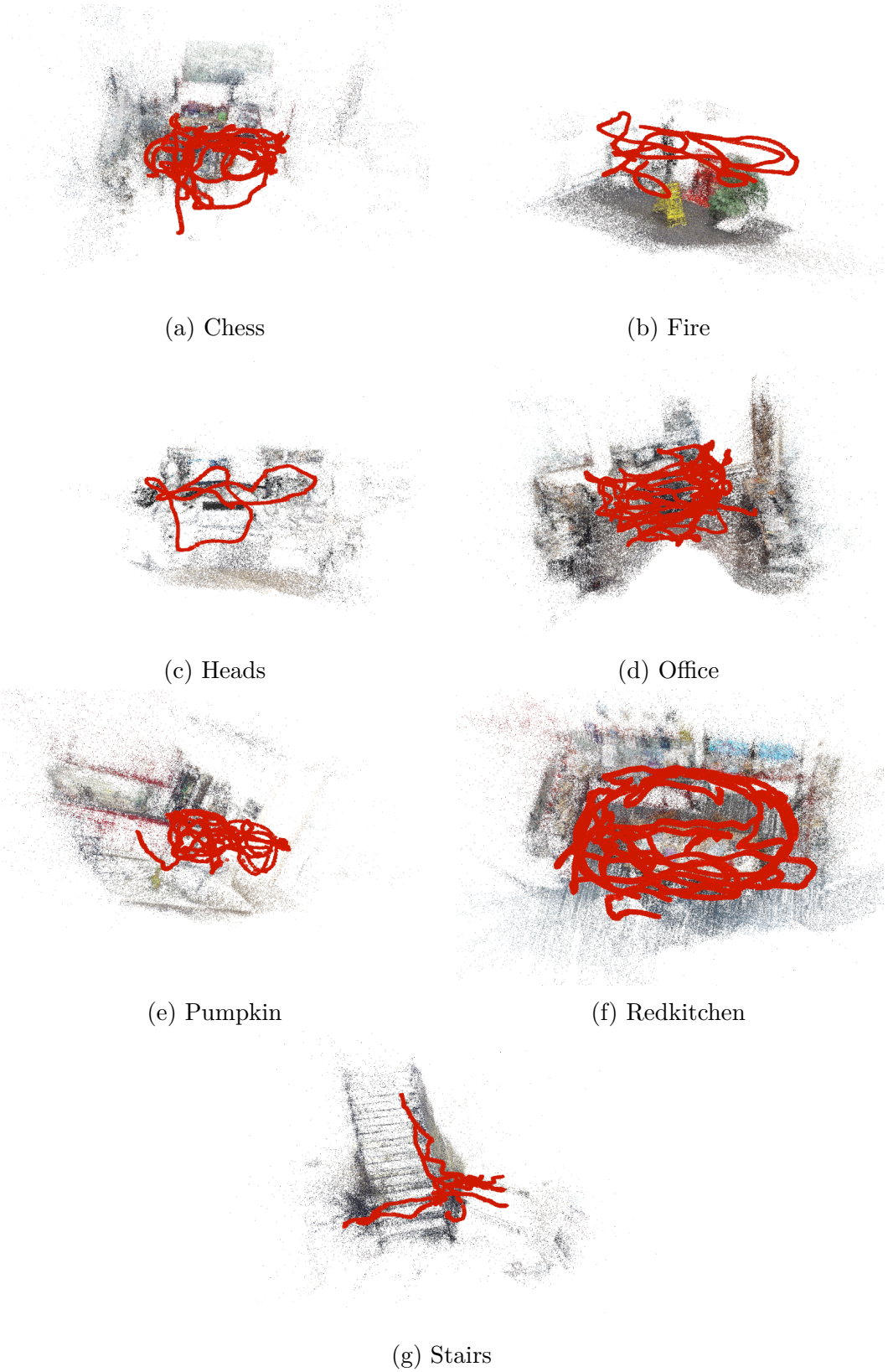


Figure 24: Example images from the 7-Scenes dataset.



(a) Chess

(b) Fire

(c) Heads

(d) Office

(e) Pumpkin

(f) Redkitchen

(g) Stairs

Figure 25: Reconstructed 3D point cloud models and camera tracks of the training sequences.

Figure 26: Example coordinate image.

Figure 27: Data augmentation.

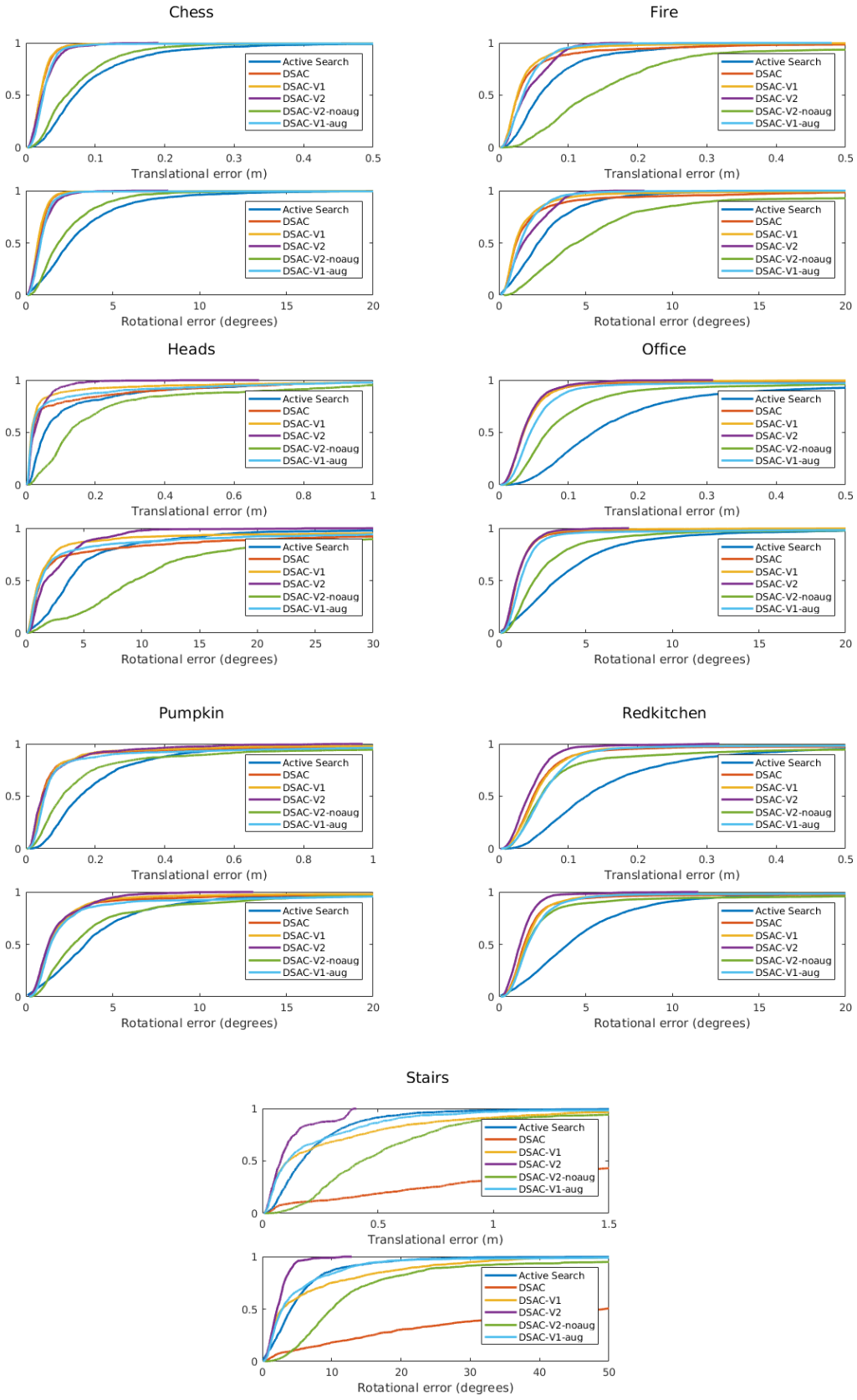


Figure 28: Localization performance presented by cumulative histograms (normalized) of errors.

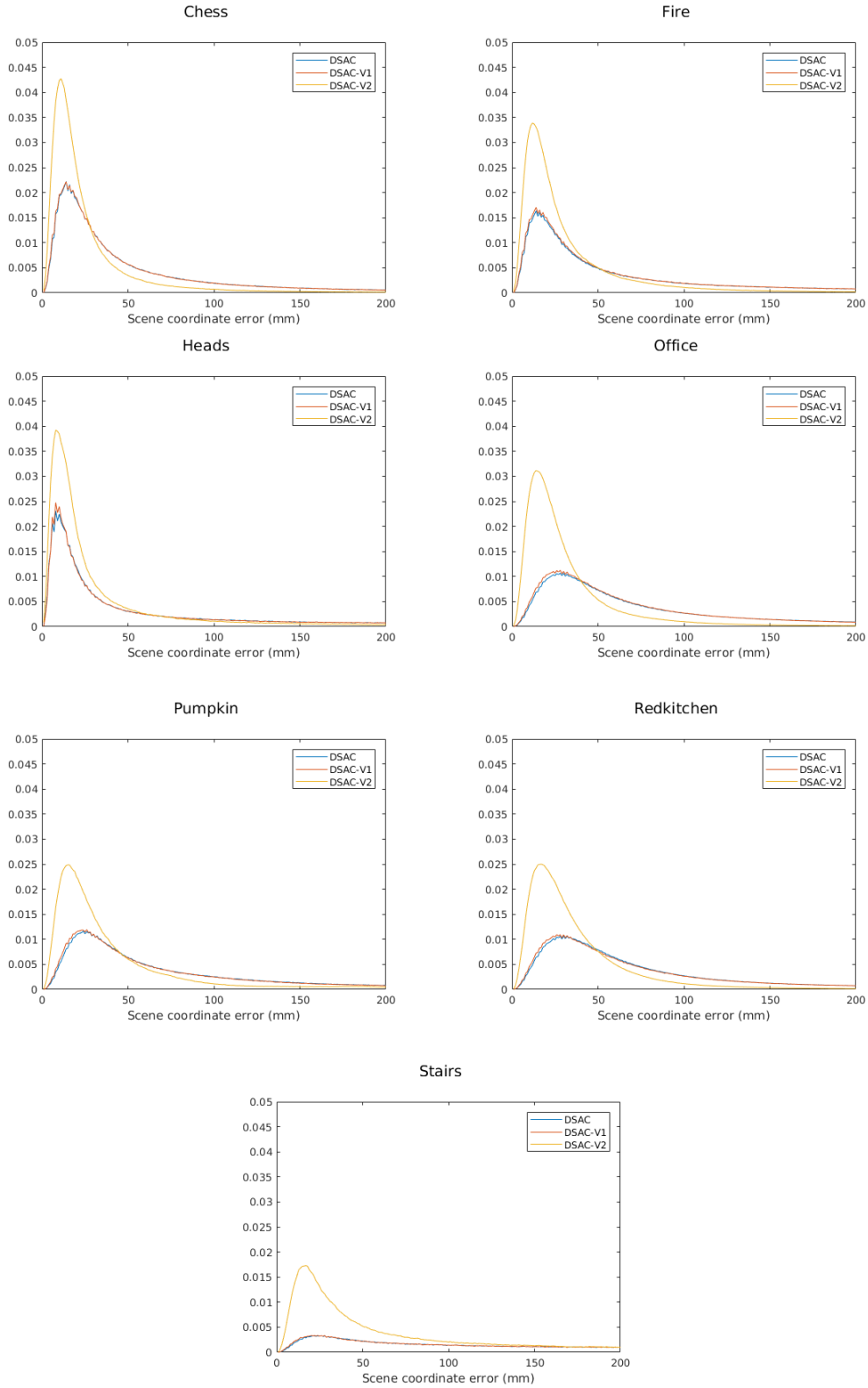


Figure 29: Histograms (normalized) of scene coordinate errors.

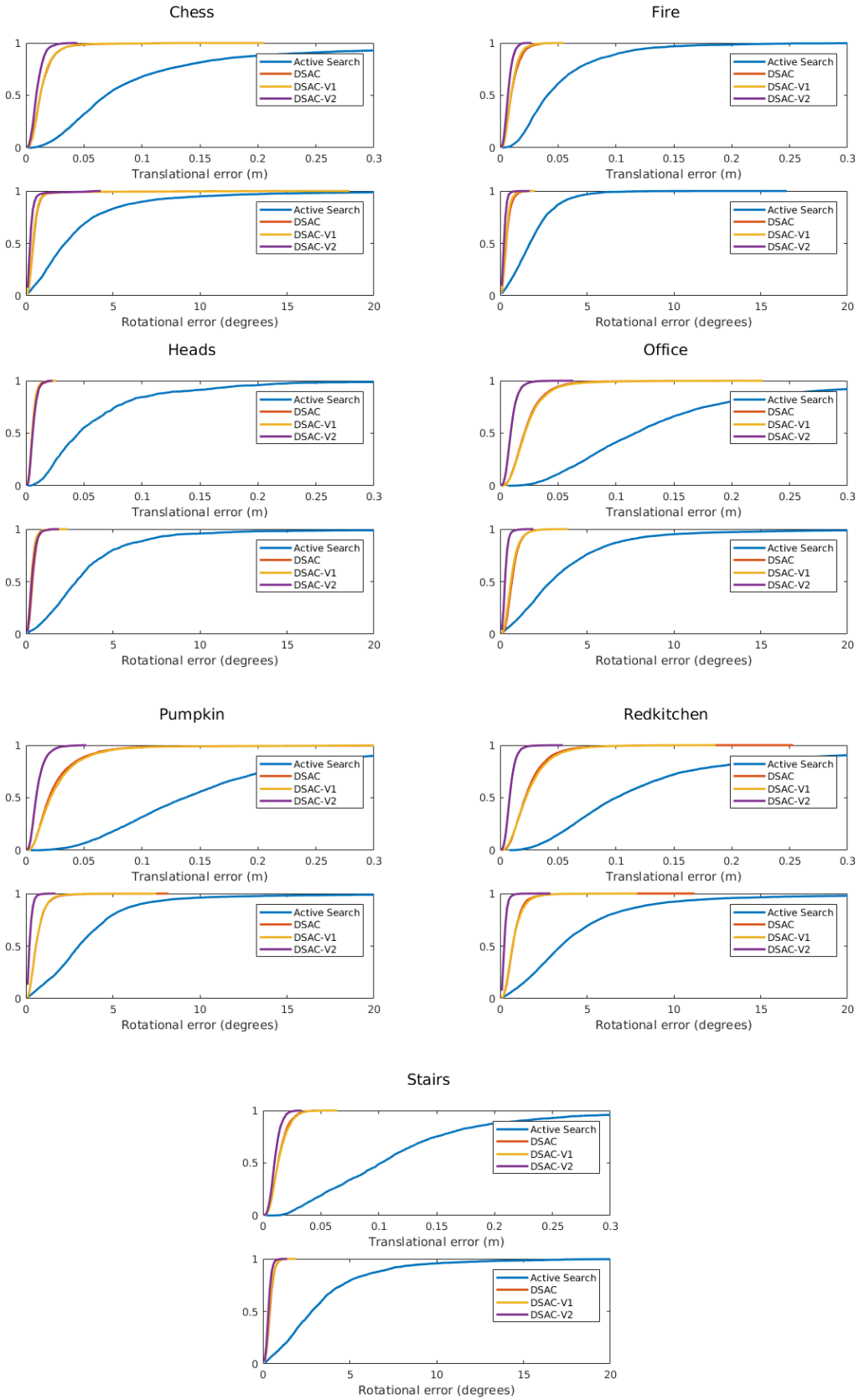


Figure 30: Localization performance on training images presented by cumulative histograms (normalized) of errors.

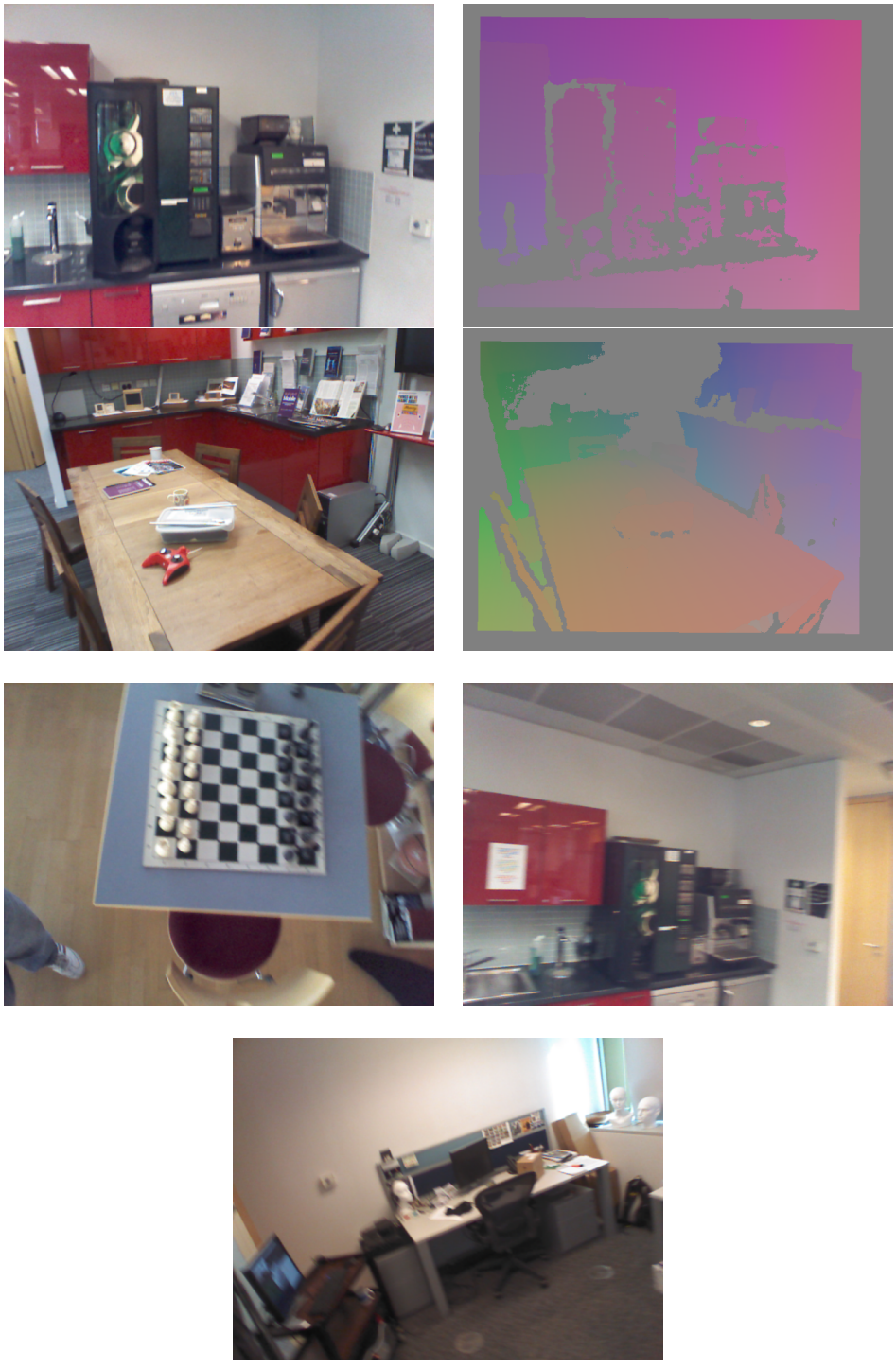


Figure 31: The most difficult training images.

7 Discussion and Future Directions

In the previous chapter, the presented results and extensive comparisons demonstrate the accuracy and robustness of Active Search, DSAC and DSAC variants on the 7-Scenes dataset. In this chapter, we give a more detailed discussion on the experimental results and identify some interesting directions for future research.

As already demonstrated, classic SIFT feature based Active Search does a reasonable job on the 7-Scenes dataset. However, obviously it can only succeed when enough matches are found by the correspondence search, and the accuracy of the final pose estimate highly depends on the quality of the matches. Since the matching performance of traditional hand-crafted local features such as SIFT is relatively poor in challenging scenarios due to repetitive structures, textureless surfaces, motion blur, and so on, the localization performance of Active Search and other keypoint based localization methods is limited.

DSAC demonstrates a promising way to overcome the limitation of keypoint based correspondence search. Instead of relying on the traditional hand-crafted local feature detector and descriptor and the correspondence search, DSAC utilizes the Coordinate CNN to generate 2D-3D matches. The Coordinate CNN can learn useful hierarchical features for the localization task from training images and directly use these features to map an image patch to a 3D position without matching. And at test time, the Coordinate CNN can be evaluated at any pixel in the image such that no feature detector is needed. The experiments show that DSAC can often provide better results than Active Search, but it is more likely to overfit the data and performs extremely bad when there are a large number of repetitive structures.

To improve the robustness of the DSAC localization pipeline, we propose to abandon the Score CNN and use the simple but robust inlier counting scheme. The results verify that this can lead to more robust localization performance. Therefore, we argue that it not necessary to use a Score CNN to make the whole pipeline end-to-end trainable. However, it still struggles when dealing with the repeating structures. In our opinion, this is caused by the local nature of the patch-based Coordinate CNN. Besides, its fixed patch size at both training and test time is problematic.

To cope with these problems, we decide to add more global information to the Coordinate CNN. We achieve this by using our full-frame Coordinate CNN which takes the whole image as input and outputs dense predictions of the same size. This also results in more efficient computation at test time. The results verify that by using our full-frame Coordinate CNN, the robustness of the localization system can be drastically improved even in the most challenging scenarios.

As mentioned before, there are also neural network based localization methods which directly formulate the localization as pose regression. However, compared to Active Search, DSAC and DSAC variants, their overall accuracy is lagged far behind. We believe that this is mainly because that the commonly used neural network architectures are not suitable for capturing the geometric structure of a scene and thus without adequate data they are unable to generalize well. In contrast, typically, the intermediate coordinate regression step in the DSAC pipeline can be

regularized well with enough data, and the follow-up RANSAC optimizer can further provide robustness to the final pose estimation. Thus, the two-step pipeline of DSAC leads to superior performance.

We believe that we should focus on the promising neural network based DSAC pipeline in our future research. There are still many problems that need to be addressed regarding the DSAC pipeline. For example, although it can do a great job on the 7-Scenes dataset, this does not necessarily imply that it can always work well. Besides, the use of depth information for obtaining the scene coordinate ground truth labels might be a limitation of this method. Moreover, although the introduction of full-frame Coordinate CNN leads to overall improved robustness and accuracy, the results also show a degenerated accuracy on some easy frames and thus it is interesting to explore the reason. More importantly, since the trainable Coordinate CNN overcome the limitations of hand-crafted methods by learning from data, it is crucial to improve its generalization property in case of less available data.

In the remaining part of this chapter, we enumerate some directions for future research and for each of them, we give a brief discussion.

7.1 Experiments on More Realistic Datasets

In this thesis, all the experiments are conducted using the 7-Scene dataset. Although it is a widely used dataset for benchmarking image-based localization methods, it is unable to present all scenarios that can be encountered in real life. For example, the scenes in the 7-Scenes dataset are registered to different coordinate systems without being linked to each other in a global coordinate system and the Coordinate CNN is both trained and tested scene-wise accordingly. However, in practice, we often have a large environment which contains several sub-scenes. It is largely unknown how the DSAC pipeline would perform if the Coordinate CNN is trained and evaluated in such a large global coordinate system.

Also, it is unclear how well the DSAC pipeline would work if the depth information used to obtain the scene coordinate ground truth labels is not as dense and as accurate as the one generated by Kinect depth sensor. This is crucially important for outdoor localization, since in these scenarios, typically, the depth images can only be obtained using the 3D reconstructed models. Such depth images are usually noisy and sparse.

Therefore, more experiments on challenging datasets (e.g. the University dataset [37]) need to be conducted to evaluate the performance of DSAC, to identify its limitations, and to propose improvements.

7.2 Unsupervised Training

As already mentioned, if we do not have available depth images for generating the scene coordinate ground truth labels, we can render the depth images from 3D models. However, it is also interesting to see whether we can train the Coordinate CNN without the scene coordinate ground truth labels, i.e., in an unsupervised manner. Recently, several unsupervised learning frameworks [18, 21, 77] have been proposed to train single view depth CNN without annotated ground truth depths. They achieve

the unsupervised training by exploiting geometry constraints between a pair of images and using view synthesis as training loss. Although it cannot be straightforwardly applied to train our Coordinate CNN, we believe it is a good starting point to design a suitable unsupervised training framework for the Coordinate CNN.

7.3 Better Network Architecture

The network architecture is always an important factor for achieving excellent results. For example, we have tried the original DispNet architecture for our full-frame Coordinate CNN, but the use of multi-scale side predictions can lead to substantially worse results. In our opinion, a better network architecture for our full-frame Coordinate CNN should have the ability to model geometric information well in a parameter efficient manner, i.e., it can generalize well with fewer data. However, currently the most commonly used CNN architectures all fail to model such information efficiently, i.e., they require a large amount of data to learn it. Recently, the new deformable CNN modules [11] have shown greatly enhanced capability of modeling geometric transformations. Thus, it is interesting to see how this kind of structures can improve the performance of the Coordinate CNN.

7.4 Better Data Augmentation

We have already seen that data augmentation can significantly improve the performance of our full-frame Coordinate CNN. It is an efficient and effective way to regularize the full-frame Coordinate CNN. However, the quality of the augmented data can also affect the performance of the network. We believe that if more realistic samples can be generated during training, the performance of the Coordinate CNN can be further improved. Our second way of data augmentation always results in a large amount of missing RGB pixels and the corresponding scene coordinate ground truth labels although it preserves the geometric relations. This makes the augmented color images noisy and the coordinate images sparse. An alternative way of generating better training samples could be using view synthesis methods (e.g. [16]).

7.5 Transfer Learning

As described in Section 2.3.5, transfer learning is also a useful way to improve the performance of a deep neural network especially when the size of the training data is not large enough. In this thesis, transfer learning is not applied since we have sufficient data for each scene and it is not straightforward to apply transfer learning technique for image-based localization. However, it is still interesting to see if transfer learning can further boost the performance of the Coordinate CNN. Moreover, in real life when we have much less training data, transfer learning could be extremely useful.

7.6 Better RANSAC Optimizer

Since the DSAC pipeline contains not only the Coordinate CNN but also the RANSAC optimization step, the performance of the RANSAC optimizer can also have a great influence on the accuracy of the final pose prediction. Although we have proved that the Score CNN used in the original DSAC pipeline is not as robust as counting inliers, we can design or train it in different ways to see how it would perform.

8 Conclusion

In this thesis, we review two state-of-the-art image-based localization methods, namely conventional SIFT feature based Active Search and neural network based DSAC. With a systematic evaluation on the 7-Scenes dataset, we show that the neural network based DSAC has promising performance compared to the conventional Active Search although it also has significant deficiencies. We propose two modifications to the DSAC methods. The first modification is to replace the differentiable RANSAC with traditional RANSAC. The results show that the simple inlier counting scheme is more robust than the Score CNN. Furthermore, we propose to use the full-frame Coordinate CNN which takes a whole image as input and produces scene coordinate predictions for all pixels in the image and can be efficiently evaluated at test time. The results show that our method outperforms the two state-of-the-art methods and has drastically improved robustness in challenging scenarios.

References

[1] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *ECCV*, 2008.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speededup robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[3] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. Dsac - differentiable ransac for camera localization. In *CVPR*, 2017.

[4] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *CVPR*, 2016.

[5] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] R. O. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *ISWC*, 2008.

[7] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, 2012.

[8] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2015.

[9] A. Criminisi and J. Shotton. Decision Forests for Computer Vision and Medical Image Analysis. *Springer*, 2013.

[10] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *IJRR*, 27(6):647–665, 2008.

[11] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *arXiv:1703.06211*, 2017.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[13] E. Eade and T. Drummond. Scalable monocular slam. In *CVPR*, 2006.

[14] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. In *TPAMI*, 2013.

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

[16] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. DeepStereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016

[17] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. In *TPAMI*, 2003.

[18] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[19] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[20] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-time RGB-D camera relocalization. In *ISMAR*, 2013.

[21] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[22] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[23] M.T. Hagan and M. Menhaj. Training feed-forward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, 1999, pp. 989–993, 1994.

[24] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2nd edition, 2004.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[26] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *NECO*, 1997.

[27] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4, 2 (1991), 251–257.

[28] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[29] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009.

[30] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *UIST*, 2011.

[31] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica*, 1976.

[32] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017.

[33] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015.

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[35] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR*, 2011.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[37] Z. Laskar, I. Melekhov, S. Kalia, and J. Kannala. Camera relocalization by computing pairwise relative poses using convolutional neural network. *arXiv:1707.09733*, 2017.

[38] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[39] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. In *IJCV*, 2009.

[40] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010.

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

[42] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[43] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart. Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization. In *RSS*, 2015.

[44] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool. Convolutional oriented boundaries. In *ECCV*, 2016.

[45] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, Vol. 11, No. 2, June 1963, pp. 431–441.

[46] D. Massiceti, A. Krull, E. Brachmann, C. Rother, and P. H. Torr. Random forests versus neural networks? what's best for camera localization? In *ICRA*, 2017.

[47] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[48] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. Image-based localization using hourglass networks. *arXiv:1703.07971*, 2017.

[49] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, 2009.

[50] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[51] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[52] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[53] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[54] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[56] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *ICCV*, 2011.

[57] T. Sattler, B. Leibe, and L. Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012.

[58] T. Sattler, B. Leibe, and L. Kobbelt. Efficient and effective prioritized matching for large-scale image-based localization. In *TPAMI*, 2016.

[59] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[60] D. Sculley. Web Scale K-Means clustering. In *Proceedings of the 19th international conference on World wide web*, 2010.

[61] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *ICCV*, 2013.

[62] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[63] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *IJCV*, 2007.

[64] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR*, 2015.

[65] C. Strecha, T. Pylvanainen, and P. Fua. Dynamic and scalable large scale image reconstruction. In *CVPR*, 2010.

[66] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[67] T. Tieleman and G. Hinton. Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report, 2012.

[68] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015.

[69] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using LSTMs for structured feature correlation. In *ICCV*, 2017.

[70] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell. Understanding data augmentation for classification: when to warp? In *DICTA*, 2016.

[71] J. Wu, L. Ma, and X. Hu. Delving deeper into convolutional neural networks for camera relocalization. In *ICRA*, 2017.

[72] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[73] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[74] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011.

[75] W. Zhang and J. Košecká. Image based localization in urban environments. In *3rd International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.

[76] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, 2014.

[77] T. Zhou, M. Brown, N. Snavely, and D. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.