Revisiting Absolute Pose Regression

Hunter Blanton
University of Kentucky, hblanton12@gmail.com
Digital Object Identifier: https://doi.org/10.13023/etd.2021.386

Revisiting Absolute Pose Regression

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy in the
College of Engineering at the
University of Kentucky

By
Hunter Blanton
Lexington, Kentucky

Director: Dr. Nathan Jacobs
Professor of Computer Science
Lexington, Kentucky 2021

ABSTRACT OF DISSERTATION

Revisiting Absolute Pose Regression

Images provide direct evidence for the position and orientation of the camera in space, known as camera pose. Traditionally, the problem of estimating the camera pose requires reference data for determining image correspondence and leveraging geometric relationships between features in the image. Recent advances in deep learning have led to a new class of methods that regress the pose directly from a single image.

This thesis proposes methods for absolute camera pose regression. Absolute pose regression estimates the pose of a camera from a single image as the output of a fixed computation pipeline. These methods have many practical benefits over traditional methods, such as constant inference speed and simplicity of use. However, they also have severe limitations, the most significant of which are high pose error and the fact that a network must be trained for each new scene. Despite the negatives, absolute pose regression is an exciting line of research with many potential use cases.

Our work focuses on three areas. First, we investigate the use of absolute pose regression across multiple scenes. We propose a method for using a mostly shared network to perform pose regression across multiple scenes without significant increase in pose error relative to per-scene networks. With this approach, we also show how the features learned during multi-scene training do not transfer to pose regression in new scenes. Next, we propose a new convolutional network to improve the accuracy of absolute pose regression. The new network takes inspiration from traditional methods to design a network explicitly for camera pose regression. As opposed to the black box approaches used by other methods, out method results in a significant decrease in pose error. Finally, we show an application of the new method to share network weights to estimate camera pose in multiple scenes. Due to the more explicit design of the network, it is naturally partitioned into scene-dependent and scene-agnostic layers, allowing us to transfer pretrained weights to novel scenes without needing to retrained the entire network.

The contribution of this thesis is a novel architecture for absolute pose regression which directly uses well known geometric relations that results in higher pose accuracy and allows for localization within novel scenes without needing to retrain the full network.

KEYWORDS: computer vision, machine learning, camera pose estimation

Author's signature: Hunter Blanton

Date: October 6, 2021

Revisiting Absolute Pose Regression

By
Hunter Blanton

Director of Dissertation:	Dr. Nathan Jacobs
Director of Graduate Studies:	Dr. Simone Silvestri
Date:	October 6, 2021

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

For decades, the field of computer science has strived to find ways to extract meaningful information from digital images. Building off of early works on signal processing for more basic tasks, convolutional neural networks (CNNs) have proved invaluable in computer vision. Notable successes include image classification [27], object detection from single images [47], and full image semantic segmentation [48]. In many cases, modern approaches can achieve accuracy very close to or even surpassing human performance, which itself is not perfect as there exists ambiguities due to visual similarity between objects as well as loss of information due to image resolution and noise.

Much of the recent success in computer vision has been due to the abundance and availability of hardware that allows for training deep learning networks with ever increasing parameter counts and complexity. In particular, increasingly deep and wide CNNs and, more recently, visual transformers [20] have been the driving technology behind the computer vision boom. Deep learning networks have allowed for accurate methods even for tasks that are difficult for humans, such as single-image depth estimation [24] and single image camera pose estimation [34]. These tasks rely on the ability to leverage prior knowledge about object size and positioning, as well as the image formation process, to reason about geometric properties of the image.

1.1 Camera Pose Estimation

Estimating the position and orientation of the camera that captured an image is a fundamental problem in computer vision. Sensor pose estimation is required for a wide variety of practical tasks such as 3D reconstruction, robot tool positioning, and AR/VR. To compute camera pose in these scenarios, it is common to have some external system for determining the sensor position such as base stations for VR headsets or keypoint based motion-capture systems [56]. These systems work by detecting and tracking known objects attached to the camera using a fixed external sensor. Due to the complexity of setup and calibration, these systems are impractical in many situations. The alternative is to use image content directly for pose estimation.

The most common way to compute camera pose from images themselves is structure-from-motion (SfM). The steps for SfM generally involve feature extraction (e.g. SIFT [38]), feature matching, and image registration. These steps require several overlapping images in order to find good image matches for registration. In general, this involves exhaustively searching for correspondences to find good initial pairs. A subset of SfM involves consecutive video frames to reduce the complexity of feature matching. These frame-by-frame methods are typically optimized for real-time use and fall under the umbrella of simultaneous localization and mapping (SLAM). Both SfM and SLAM build 3D maps of the environment as well as estimate camera pose for each image. However, these are only applicable when you have a collection of overlapping images in order to triangulate points in space through multi-view geometry. A sub problem inside these systems is the question of how to localize a single image inside the current map. This scenario arises often in practice. Common examples include adding newly collected data to improve the reconstruction quality of the SfM map, or to reinitialize tracking in a previously mapped location for AR/VR headsets.

There are a variety of techniques that are used for single image pose estimation. The foundation of many algorithms is the ability to generate a set of nearby images. In the case of SfM or SLAM, a course camera location can sometimes be assumed to reduce the search space for nearby images. But generally, the search is performed by computing distance metrics on global image descriptors, such as DenseVLAD [58], with all images of the surrounding area. This finds the most similar images in feature space to the query image. Once a set of nearby images has been found, there are many ways to compute the final image pose. Early work focused on finding explicit correspondence between 2D image points and 3D points in the map [28, 51]. Once the 2D-3D correspondences are found, the camera pose can be computed using the perspective-n-point algorithm [21]. Other methods use CNNs for regressing relative pose directly from images [19]. An alternative to retrieval based methods is scene coordinate regression. For scene coordinate regression, either a random forest [55] or CNN [7] is used to directly estimate the 3D position of a pixel from the pixel content alone. This removes the need for image retrieval since information about image correspondence can be implicitly embedded into the weights of the random forest or CNN.

A recent line of work, beginning with PoseNet [34], forgoes feature correspondence all together. These methods instead rely on a CNN to directly estimate the camera pose from a single image. In literature, these methods have come to be known as absolute pose regression (APR). This approach has been demonstrated to work on a variety of indoor and outdoor scenes [11, 34]. They tend to work equally well across scene complexity and size, and are limited largely to the availability of training data. However, while several improvements have been made since the introduction of PoseNet, the pose error is still typically much worse than more traditional methods [52]. And unlike methods that depend on retrieval, correspondence matching and/or relative pose estimation which can often be applied generically to new scenes, another disadvantage of APR is that they must be trained individually for each scene.

1.2 Contributions

Our thesis focuses on estimating the pose of a single image using absolute pose regression. We focus on some of the major issues and attempt to solve them.

- **Weaknesses of APR:** We investigate the weaknesses of recent absolute pose regression approaches, specifically the inability to work beyond a single scene and low accuracy compared to traditional methods. We propose a method for absolute pose regression in multiple scenes using a single network that allows for joint training of several scenes as well as inference across multiple scenes without significant increase in pose error. We are the first to explore the problem of multi-scene absolute pose regression.

- **Improving the Architecture for APR:** We explore how a network architecture more suited for absolute pose regression based on geometric concepts. We make use of scene coordinate regression and depth estimation from a single image. Joint inference of the 3D geometry of the scene in these two coordinate frames allows us to make use of well understood 3D principals to compute camera pose in efficiently and accurately. Our proposed method exhibits significant improvement over other absolute pose regression methods.

- **Removing Scene Dependence from APR:** Finally, we show how the improved architecture allows for explicit partitioning of components to allow for APR across several scenes with a largely shared network. In this work, we show how the architecture can be partitioned into scene-dependent and scene-agnostic components such that only a portion of the network must be trained for each new scene. This allows for both state-of-the-art accuracy and pose error for single scene absolute pose regression, but also works well when applied to novel scenes.

The contribution of this thesis is an improved framework for absolute pose regression that is more accurate than previous approaches while also allowing part of the network to be reused in novel scenes without needing to completely retrain. Compared to other absolute pose regression methods, our work is a significant departure from prior work and explores a novel approach to pose regression with CNNs.

1.3 Synopsis

The remainder of this work is organized as follows:

- **Chapter 2 - Extending Absolute Pose Regression to Multiple Scenes**

  In this chapter, we propose an approach for performing absolute pose regression in multiple scene using a single network. We show that a large portion of the network can be shared to maintain similar error metrics to single scene approaches while being able to be used across multiple scenes. This modification to the standard PoseNet greatly reduces the number of parameters needed for camera pose estimation across a large number of scenes. We also show that, while it is possible to localize in multiple scenes, features do not transfer to novel scenes, so we must still retrain the network when going beyond the initial training set. This work was originally reported in [3].

- **Chapter 3 - A Structure Aware Method for Direct Pose Estimation**

  In this chapter, we describe a novel architecture for absolute pose regression. We incorporate depth estimation and scene coordinate regression which allows us to transform the camera pose estimation task into a corresponding point cloud alignment problem. We use a differentiable method for corresponding point cloud alignment to construct an end-to-end method for pose estimation. We show that our method is much more accurate than other absolute pose regression baselines. This work was originally reported in [4].

- **Chapter 4 - Factoring Out Scene Dependence in Absolute Pose Regression**

In this chapter, we show how our proposed method from the Chapter 3 can be extended to allow for absolute pose regression in several scenes with a largely shared network. We show a natural and explicit partitioning between scene-dependent and scene-agnostic layers. We show that this network extends to novel scenes that where unseen during training of the scene-agnostic layers of the network. This method achieves state-of-the-art performance is single scene training and matches the performance of early APR methods even in the case of inference on held out scenes.

- **Chapter 5 - Discussion**

  In this chapter, we summarize the contributions of this thesis. We highlight the key findings as well as discuss the significant of each contribution. Finally, we discuss several possible future research directions.

Chapter 2

Extending Absolute Pose Regression to Multiple Scenes

Absolute pose regression using deep convolutional neural networks has become a highly active research area. However, even with significant improvements in performance in recent years, the best performance comes from training distinct, scene-specific networks. We propose a novel architecture, Multi-Scene PoseNet (MSPN), that allows for a single network to be used on an arbitrary number of scenes with only a small scene-specific component. Using our approach, we achieve competitive performance for two benchmark 6DOF datasets, Microsoft 7Scenes and Cambridge Landmarks, while reducing the total number of network parameters significantly. Additionally, we demonstrate that our trained model serves as a better initialization for fine-tuning on new scenes compared to the standard ImageNet initialization, converging to lower error solutions within only a few epochs.

2.1 Background

Humans are extremely adept at localizing a camera from image content alone. If the general location is known, this process typically involves identifying recognizable geographic or architectural features to estimate how and where one would have had to position themselves to produce the image. In computer vision and robotics this task is often referred to as image-based localization, or camera localization, and represents the problem of estimating the position and orientation of the camera from which an image was taken, i.e., its corresponding camera pose, with respect to some underlying scene representation.

The predominant approach to solving this problem involves constructing a 3D model of the scene and estimating the camera pose using feature-based localization, or identifying 2D-3D correspondences between the image and the known scene model. Though this approach tends to be extremely accurate, it is slow and resource intensive. Recently, learning-based approaches [11, 32, 34] have used convolutional neural networks (CNNs) to regress camera pose in an end-to-end fashion. In such approaches, the weights of the network implicitly model the underlying scene layout, as opposed to feature-based methods which explicitly take advantage of a 3D model. In these absolute pose regression techniques, each model is specialized to a single area or scene. In other words, for each scene, a network is trained using a scene-specific set of training images and corresponding ground-truth camera poses.

The major benefit of an end-to-end approach is that estimating the pose of an image requires only a single forward pass through the network, which is much faster than structure-based methods. For example, PoseNet [33] takes less than $10ms$ per image, while DSAC++ [8] takes over $100ms$ per image. However, there are several drawbacks. First, adding new data requires completely retraining the model. Second, existing datasets are limited and provide very few training examples for each scene. In the case of Cambridge Landmarks [34], for example, there are as few as 250 training samples for a given scene. Finally, recent work has shown that learning-based approaches are still significantly less accurate than structure-based methods [52].

Despite these issues, we believe absolute pose regression has many practical benefits over more accurate methods. Pose regression using a CNN can perform spontaneous relocalization in a way that (1) is deterministic, (2) is fast, and (3) runs in constant time regardless of scene size. No other method has all three of these properties simultaneously. These properties make CNN-based pose regression ideal for many tasks ranging from autonomous navigation to mixed reality where real-time, reliable performance is crucial. However, the use case is limited to the single location in which the network was trained. In this work, we propose a method to regress camera pose across several different scenes without loss of accuracy or the need to store several large networks.

We propose a variant of PoseNet which we refer to as Multi-Scene PoseNet (MSPN). Our key insight is that we can make scene identification explicit in the network architecture. We implement this as a two stage network, shown in Figure 2.1. The first part of the network is shared across scenes and learns a general camera localization feature. This feature is then used for scene prediction, which indexes a database of scene-specific weights, as well as final pose regression using the indexed weights. Since each scene effectively has a unique head in the network, our approach can be thought of as multi-task learning where each scene is a unique task. This allows us to move beyond memorizing each scene individually to learning a common model for camera localization across scenes. Further, MSPN enables training of the feature extractor using an order of magnitude more images, which we hypothesize may lead to more robust localization performance and improved accuracy.

We evaluate our proposed approach extensively on common benchmark datasets. Our key contributions include:

2.2 Related Work

Estimating camera pose, i.e., the position and orientation of a camera relative to a scene, from a captured image is a fundamental problem in computer vision. Applications include 3D Reconstruction, mixed reality, and autonomous driving. Unsurprisingly, there are a wide variety of methods. For a thorough overview of work in this area, see Sattler et al. [52].

Image retrieval has been used extensively as part of localization systems. For example, a standard approach for image localization is to build a large database of images with known location, and then infer the location of a query image from the closest match (or some combination of the set of closest matches). Hays and Efros [26] demonstrated the success of image retrieval for image geolocalization using a large dataset of 6 million images. Other methods [1,63] learn a low-level image feature and perform nearest-neighbors search to predict the location and pose of a query image. Image retrieval has been successfully applied for problems in structure-based pose regression (matching 2D to 3D) [66] and relative pose regression (estimating pose relative to a set of training images) [19]. In this work we focus on direct absolute pose regression with no retrieval step.

A large body of work has explored methods for directly estimating camera pose from an image using convolutional neural networks [11, 32, 34]. Though not as accurate as structure-based methods [52], these methods are attractive as they have fast, constant time inference regardless of scene complexity. The standard approach for such methods is to learn a feature embedding using a CNN, which implicitly captures details of the scene in the weights of the network, and then use features from this embedding to regress the camera pose. This requires only a single forward pass of the network. However, each scene typically has its own specialized model.

Methods for direct pose regression largely differ in choice of architecture or loss function. The simplest approach, PoseNet [34], trains a feed forward neural network using absolute error in orientation and position as a loss function. MapNet [11] enforces a relative pose correctness constraint that makes the final predictions more spatially consistent. LSTM PoseNets [61] replace a fully connected layer with an LSTM for feature reduction before final pose regression. Hourglass PoseNets [41] use an encoder-decoder network with skip connections. Despite the differences in these approaches, they tend to have similar performance overall. In our work, we focus on the PoseNet framework for absolute pose regression and explore how to take advantage of multiple scenes during model training. To our knowledge, our work is the first to explore methods for multi-scene absolute pose regression.

AnchorNet [49] explicitly partitions space into anchor points and regresses relative camera positions to every point. This work is similar to ours in that they take the first step towards a unified network that operates on distinct regions. However, AnchorNet still operates on a single scene at a time. The fundamental difference with our approach is that the final pose prediction is computed using a weighted average of relative anchor point poses. We instead directly regress a single absolute pose without averaging across many predictions. Our approach can be implemented in tandem with AnchorNet and we view them as complementary.

A separate branch of CNN-based pose regression is scene coordinate regression. DSAC [7] and the follow up DSAC++ [8] train a CNN to predict the 3D locations of pixels to establish 2D-3D correspondences. Next, the Perspective-3-Point algorithm and RANSAC [21] are used to compute the final camera pose. These methods are extremely accurate and compete with traditional structure-based approaches. However, they are much harder to implement and training takes much longer to converge. Also, due to the inherent randomness of RANSAC, these methods are not deterministic and have longer run-time. While some work has been done on performing pose regression across multiple scenes using these approaches [9,55], they still require a separately trained expert network for each scene. Instead, we use a largely shared network with only a small scene-specific component.

2.3 Background

We describe the current paradigm of learning-based methods for absolute pose regression in a specified scene. Given a single image, the model output is a vector representing camera pose p, composed of location t and orientation q:

$$p = [t, q]. \tag{2.1}$$

Camera location t is a triple, $[x, y, z]$, which defines the spatial location of the camera center relative to the origin of the scene in 3D Cartesian coordinates. Camera orientation q defines a relative rotation to a canonical pose origin. Typically, for absolute pose, this is represented as a quaternion [34] instead of more traditional angle parameterization due to the inherent non-uniqueness of Euler angles. Recent approaches have had success using the log-quaternions [11]. The log-quaternion is desirable because it has fewer parameters (3 vs. 4) and naturally defines a unit quaternion without the need for explicit normalization.

During training the mean absolute error of each pose component, weighted by a hyperparameter β, is minimized:

$$L(\hat{p}, p) = \left\|\hat{t} - t\right\|_1 + \beta \left\|\hat{q} - q\right\|_1, \tag{2.2}$$

where $\hat{p}$ is the ground truth pose and p is the output of the network. The value of β is specific to each scene. Some approaches [34] set β before training begins, typically between 300 and 500 for indoor scenes and between 500 and 1000 for outdoor scenes. More recent works learn this weighting as part of the training process with a Laplace likelihood [32]:

$$L(\hat{p}, p) = \left\|\hat{t} - t\right\|_1 e^{-s_t} + s_t + \left\|\hat{q} - q\right\|_1 e^{-s_q} + s_q, \tag{2.3}$$

where s_t and s_q are optimized during training. By allowing the loss to be weighted dynamically, training can be performed without the need for hyperparameter tuning.

2.3.1 Theory of Absolute Pose Regression

The standard deep CNN method for absolute pose regression can be segmented into three stages [52]:

1. A function that extracts a localization feature from the image.
2. A non-linear embedding of this feature into high dimensional pose components.
3. A linear mapping from pose components to final pose using the learned basis poses.

[noitemsep] In many learning-based approaches, the entire process is represented as a single CNN and a separate model is then trained end-to-end for each scene. For example in recent work on PoseNet [33], stage 1) is a standard ResNet-34 network, stage 2) is a fully connected layer with a ReLU activation, and stage 3) is another fully connected layer which outputs the final pose parameters. The fact that these methods use separate networks for each scene makes them very costly in terms of training time and number of stored parameters.

We propose an alternative, in which feature extraction is shared entirely by all scenes and only the final pose regression layer is trained to be scene-dependent. Additionally, We show that stage 1) and 2) above can be merged to reduce the network complexity without losing accuracy. In fact, our modification boosts performance in many cases. Our proposed alternative is more efficient in terms of learned parameters and can quickly learn to localize cameras in new scenes.

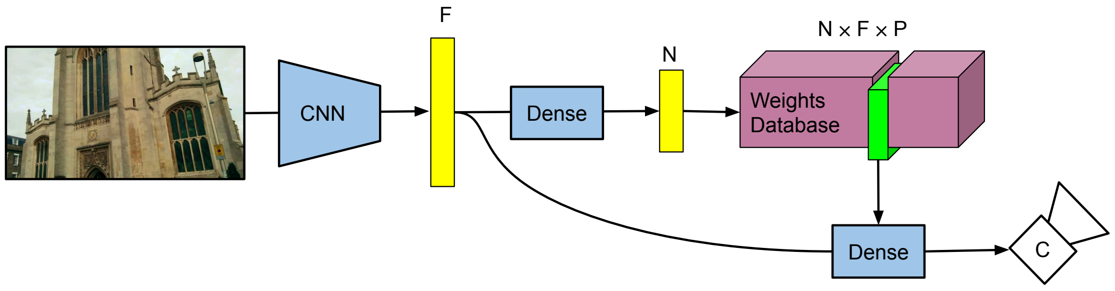


Figure 2.1: Our proposed Multi-Scene PoseNet architecture for multi-scene pose estimation. A convolutional neural network regresses an F dimensional image feature. This feature then goes through two branches. The first branch uses the image feature to predict a probability distribution over N possible scenes which is then used to index into a scene-specific weights database. The second branch uses the weights extracted from the weights database to construct a dense layer to transform the image features into the P dimensional pose parameters. The pose parameters are then used to construct the camera pose C.

2.4 Multi-Scene Absolute Pose Regression

A naïve approach to estimating pose across multiple scenes is to train a single PoseNet with a training set containing images from all scenes, with no further modification. While this approach is simple, it often suffers a loss in performance over a single network per scene. We show that this decrease in performance can be overcome in many cases without a significant increase in model complexity.

We propose Multi-Scene PoseNet to decouple scene and pose using a database of scene-specific weights, shown in Figure 2.1. First, we use a CNN to extract an F dimensional image feature. This image feature is then used to regress an N dimensional probability distribution over each of the N possible scenes. The most likely scene (maximum probability) is used to index into a database of scene-specific weights producing an $F \times P$ fully connected layer, where P is the dimension of the pose parameters. Finally, the image feature is passed through the selected fully connected layer for final pose prediction. This is similar to ESAC [9], which uses a set of scene-specific expert networks and a gating network for expert selection. However, in our method a majority of the network is shared between all scenes and only the final layer is specific to a given scene.

The network is trained to predict the image scene and pose parameters jointly in an end-to-end manner. Pose regression is optimized using (2.3) and scene prediction is optimized using cross entropy. This results in features that are explicitly discriminative by scene as well as being useful for pose regression. The final loss function for pose prediction p and scene prediction s, with corresponding labels $\hat{p}$ and $\hat{s}$ becomes

$$Loss = L(\hat{p}, p) - \frac{1}{n}\sum_{i=1}^{N} \hat{s}_i log(s_i), \tag{2.4}$$

where N is the number of scenes.

With the exception of the fully connected layers in the weights database, all other parts of the network are shared by all scenes. This results in an extreme reduction in required parameters per scene. For example, using the approach of [33] each new scene requires its own model with over 22 million parameters. In our approach, the base network contains 22 million parameters and each new scene adds approximately 14 thousand parameters. For 10 scenes, our method uses 10% of the parameters. For 100 scenes, our method uses only 1% as many parameters. Having such a significant portion of the network shared across scenes allows for the use of deep network based localization over diverse areas, even in memory starved environments such as embedded vision systems on small autonomous drones.

2.4.1 Improving Network Efficiency

As described in Section 2.3.1, the theoretical model of absolute pose regression has three stages: feature extraction, non-linear projection of the feature vector to contravariant components in the learned pose basis, and finally a linear mapping from the learned pose basis to the canonical basis of the dataset. These components are represented explicitly in PoseNet as ResNet-34 feature extraction and global average pooling, a fully connected layer with ReLU activation, and a final fully connected layer with no activation, respectively. The ReLU activation in the second stage is troublesome as it removes the possibility of negative components to be present in the final linear mapping. Since negative pose values are not only possible, but as likely as positive values, this means that the network must learn two effective copies of any useful basis pose to be able to produce both positive and negative poses. Methods have been explored to solve this redundancy with different activations [53], but we found removing this fully connected layer altogether to be effective. Using only a single fully connected layer also results in a more compact representation for our weight database.

2.4.2 Implementation Details

For our experiments we use a ResNet-34 [27] feature extractor initialized from pretrained ImageNet weights. We do this to be consistent with PoseNet and MapNet for fair comparison. We train for 100 epochs with a batch size of 64 using the Adam optimizer with a learning rate of 10^{-4}. During training, we scale all input images such that their smaller dimension is 256 and then train on random 224×224 crops. At test time, we use a 224×224 square center crop. As in [33], each scene has 2 learned loss weighting parameters that are specific to the scene. For example, this means there are 14 unique loss weighting parameters learned when training with 7 unique scenes. We opt to use the log-quaternion representation of orientation due to the fact that it requires no explicit normalization for conversion to a rotation [11]. During training, the true scene is used for determining which final layer to use for pose regression. During inference, we use the most likely scene as predicted from the scene prediction branch.

2.5 Evaluation

We show comparisons with our method to the current state-of-the-art methods, PoseNet and MapNet. As is convention in these works, we provide median position and orientation error. All networks, including PoseNet and MapNet, use ResNet-34 as the backbone CNN.

2.5.1 Datasets

We train and evaluate using two common 6DOF camera pose datasets, Microsoft 7Scenes [55] and Cambridge Landmarks [34]. These two datasets have become the standard benchmarks for learning-based direct pose regression. 7Scenes is composed of 7 small indoor scenes ranging in difficulty and size, but all scenes are less than 10 meters wide. Cambridge Landmarks on the other hand is composed of 6 large-scale outdoor scenes on the order of 100s of meters wide. Both datasets were collected with hand-held cameras and ground truth data was computed using highly accurate structure-from-motion algorithms.

2.5.2 Effect of Joint Training

We first test the effects of the scene-specific component in the case of a scene oracle. That is to say, the true scene index is used to query the weights database instead of the output of the scene prediction head. By jointly training several scenes together with our method, we not only match, but in many cases improve upon the accuracy for each scene. By training on multiple scenes, the shared feature extraction network gets orders of magnitude more examples to learn from. Although different scenes can be seen as different domains in the context of localization, increasing the number of training examples may help increase the generality of the feature extraction, perhaps reducing over-fitting to the training sequences

Table 2.1: Results on 7Scenes dataset. Both PoseNet and MapNet are single scene approaches. The naïve approach is a single PoseNet with no modification trained on each of the scenes simultaneously. MSPN is our method. We report median translation / median orientation error. Values are highlighted red when they are significantly worse than what we are able to achieve with our modified network.

Method	Chess	Fire	Heads	Office	Pumpkin	Red Kitchen	Stairs
PoseNet (Single)	0.14/4.50	**0.27**/11.8	0.18/12.10	0.20/5.77	0.25/4.82	0.24/5.52	0.37/**10.60**
MapNet (Single)	**0.08/3.25**	**0.27**/11.7	0.18/13.3	0.17/**5.15**	0.22/**4.02**	0.23/**4.02**	**0.30**/12.10
Naïve (Multiple)	0.15/4.85	0.28/13.13	0.30/**11.54**	0.23/6.34	0.29/5.34	0.29/6.98	0.35/10.63
MSPN (Multiple)	0.09/4.76	0.29/**10.50**	**0.16**/13.10	**0.16**/6.80	**0.19**/5.50	**0.21**/6.61	0.31/11.63

Table 2.2: Results on Cambridge Landmarks. The difference between our method and the naïve approach are less significant here. We postulate that due to the capacity of the network and the small size of the training sets, our method provides less benefit for these scenes.

Method	Kings College	Old Hospital	Shop Facade	St. Mary's Church	Street	Great Court
PoseNet (Single)	**0.99/1.06**	2.17/**2.94**	**1.05/3.97**	**1.49/3.43**	**20.7/25.7**	-
MapNet (Single)	1.07/1.89	**1.94**/3.91	1.49/4.22	2.00/4.53	-	-
Naïve (Multiple)	1.72/3.12	2.58/5.96	1.97/9.25	2.55/6.92	48.37/45.46	12.04/10.99
MSPN (Multiple)	1.73/3.65	2.55/4.05	2.02/7.49	2.67/6.18	26.78/54.22	9.2/10.42

in each scene. Note that the batch size for an individual scene is on average $\frac{batch\ size}{N}$ for N scenes, so the effective batch size for the scene-specific weights is small for a large number of scenes.

A comparison with other methods is shown in Table 2.1 and Table 2.2. To verify that the network capacity is not simply large enough to learn all scenes, we also train a single PoseNet without any scene-specific layers on all scenes simultaneously as a baseline. Although it performs remarkably well, performance is typically much worse than our method. These results show that simply training a single model does not work well in general. Note that one method is not dominant over any other for either dataset. Keep in mind that the fundamental challenge we are trying to solve is parameter reduction for scaling to multiple scenes, but it tends to perform as well as or better than other methods. We believe forcing only the feature extraction to be generic is imposing implicit regularization, reducing the tendency of the network to overfit to the training data. Note that some scenes Cambridge Landmarks do not show the metric gain from our method compared to the baseline that is observed for 7Scenes. Due the the high capacity of the network, low number of training examples per scene, and low number of individual scenes, the naïve baseline can still perform reasonably well on this dataset. However, our methods exhibits a clear benefit for the 7Scenes dataset.

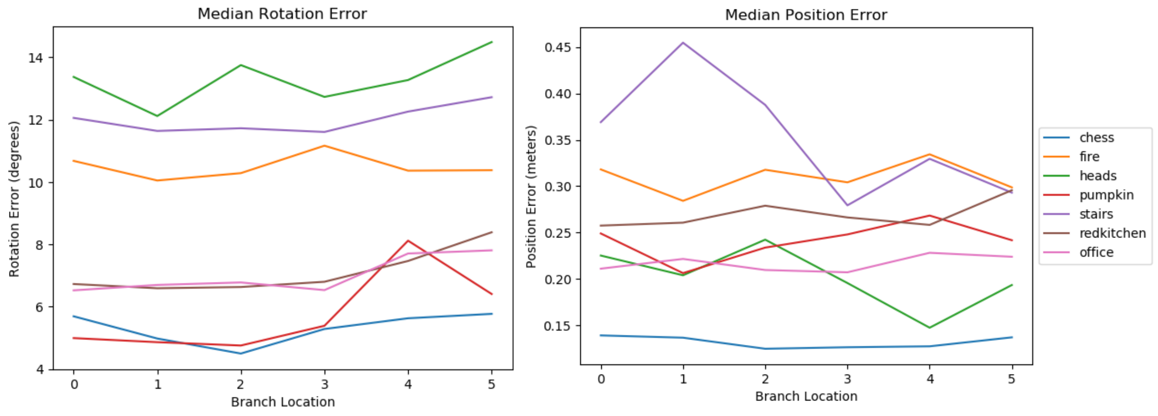


Figure 2.2: Orientation and position error with different branching points for the shared weights. A lower branch location signifies less of the network is shared.

We also experiment with splitting the network at different locations in the network. Results are given in Figure 2.2. For this experiment, we share the weights for each scene up to the the specified branch point. Branch location 1 means after the first residual block, 2 means after the second, etc. While there are some slight deviations due to random initialization and batch shuffling during training, overall we see that nearly all of the network can be shared without effecting performance.

2.5.3 Stability With Different Scenes

As the key component of our method is training on a set of scenes at once, it is of interest to see if it performs similarly regardless of the scenes used for training. To test this, we train multiple networks where each network is trained using a different subset of all scenes in the dataset and each subset is missing a single scene. Table 2.3 and Table 2.4 show the general stability of our method to different scenes. With a few outliers, the error for a given scene remains similar regardless of which scenes we trained on. However, some scenes do seem to make learning difficult. For example, training the Landmarks dataset together without the Street scene results in uniformly improved performance by 10's of centimeters, which is a sizeable improvement. Upon inspection of the dataset, this is somewhat unsurprising as the Street training set was collected only along cardinal directions. While the strangeness of this specific scene is apparent to us, it is not trivial to determine that this scene is challenging for a network to train on. Being able to quantitatively show how a scene effects training is a beneficial and unique ability of the proposed multi-scene pose regression framework.

Table 2.3: We experiment with leaving one scene out during training of the Cambridge Landmarks dataset. We observe relatively stable performance with respect to the left out scene. Note that in some cases, we can see that a particular scene hinders performance for other scenes, most notably the model trained without the Street dataset performs much better on all other scenes than the other training setups. We report median translation / median orientation error.

	Great Court	Kings College	Old Hospital	Shop Facade	St. Mary's Church	Street
w/o Great Court	-	1.13/7.04	2.16/9.41	1.33/13.49	1.59/10.69	15.20/61.03
w/o Kings College	8.16/20.33	-	2.33/9.54	1.56/14.34	1.56/11.75	15.09/58.52
w/o Old Hospital	7.51/18.51	1.01/7.71	-	1.68/16.08	1.61/10.39	14.63/60.02
w/o Shop Facade	6.85/17.08	0.98/6.81	2.15/9.01	-	1.51/10.55	18.63/55.41
w/o St. Mary's Church	6.92/16.04	1.07/5.84	1.79/8.27	1.36/12.74	-	14.28/51.75
w/o Street	6.18/13.33	0.91/5.31	1.96/8.54	1.07/11.52	1.29/8.13	-
standard deviation	0.67/2.36	0.08/0.86	0.19/0.49	0.21/1.53	0.12/1.19	1.57/3.38

Table 2.4: Leave-one-out experiment results for the 7Scenes dataset. We train a network using only 6 of the scenes to show the stability of our method.

	Fire	Heads	Stairs	Office	Red Kitchen	Chess	Pumpkin
w/o Fire	-	0.13/13.86	0.34/12.38	0.16/6.78	0.21/7.43	0.09/4.66	0.20/4.97
w/o Heads	0.27/9.70	-	0.29/11.04	0.16/6.75	0.22/7.92	0.10/4.90	0.21/5.53
w/o Stairs	0.26/10.94	0.15/13.62	-	0.16/6.35	0.23/7.51	0.10/4.90	0.20/5.19
w/o Office	0.26/10.95	0.14/12.95	0.30/11.70	-	0.22/7.47	0.10/5.14	0.19/5.12
w/o Red Kitchen	0.26/11.01	0.13/14.08	0.30/12.31	0.16/6.55	-	0.13/5.67	0.21/5.00
w/o Chess	0.27/10.92	0.14/12.72	0.28/13.00	0.16/6.54	0.22/7.65	-	0.21/5.66
w/o Pumpkin	0.28/10.69	0.14/12.93	0.34/11.88	0.16/6.52	0.22/7.38	0.11/4.95	-
standard deviation	0.007/0.459	0.007/0.516	0.023/0.612	0.000/0.146	0.006/0.181	0.013/0.316	0.007/0.261

2.5.4 Quickly Training New Scenes

The backbone of our method is a generalized localization feature extractor. Though the extracted features may have learned some amount of scene-specific information due to the high capacity of the network, the feature extractor acts as a good starting point for training localization of new scenes. There are two ways to approach this, freezing the feature extractor and training only the final scene-specific layer, or fine-tuning the whole network for the new scene. Unfortunately, fine-tuning only the final layer results in very poor performance, as shown in Table 2.5. As mentioned above, we believe that this is because the network has the capacity to learn scene-specific features for the small number of scenes used in our experiments, meaning the resulting feature extractor is not completely independent of the scenes it was trained on. However, we do find that the resulting models serve as much better initialization for fine-tuning than ImageNet pretraining. Table 2.6 and Table 2.7 shows the final performance of networks trained for one epoch on specific scenes. We chose to train for only one epoch to highlight the improvement in performance on the new scene after just seconds of training. In all cases, one or both of position and orientation has much

Table 2.5: We finetune just the scene-specific layers with different network initialization. For ImageNet, the feature extractor is initialized from ImageNet. For MSPN, we initialize with a network trained with all other scenes.

Initialization	Fire	Heads	Stairs	Office	Red Kitchen	Chess	Pumpkin
ImageNet (finetune)	0.92/101.23	0.87/97.00	0.87/89.91	0.95/84.60	1.00/87.37	0.84/83.11	0.83/87.08
MSPN (finetune)	0.76/31.39	0.44/23.15	0.69/32.95	0.98/45.69	1.32/33.37	0.82/23.28	0.76/29.86
MSPN (full)	0.29/10.50	0.16/13.10	0.31/11.63	0.16/6.80	0.21/6.61	0.09/4.76	0.19/5.50

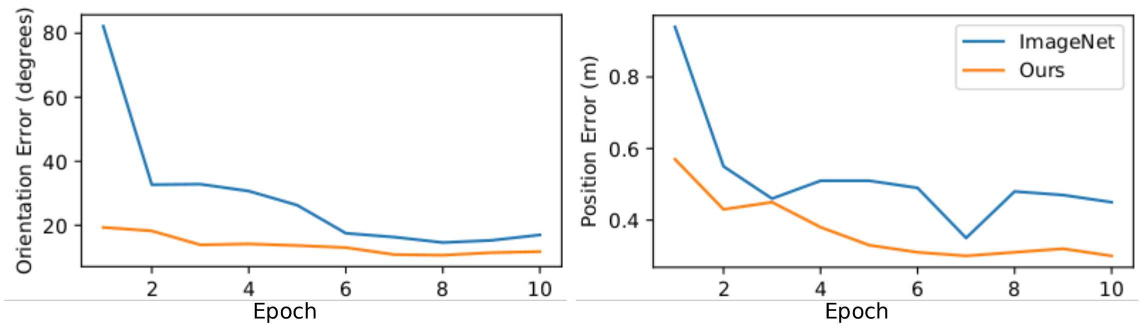


Figure 2.3: Error in orientation and position on the Fire test set as training continues. “ImageNet” signifies the network was initialized from ImageNet pretrained weights. “MSPN” signifies that the network was pretrained for pose regression on other scenes. Fine-tuning a new scene from a network pretrained for pose regression converges much faster than from ImageNet pretraining even if the original scenes are very different from the new scene.

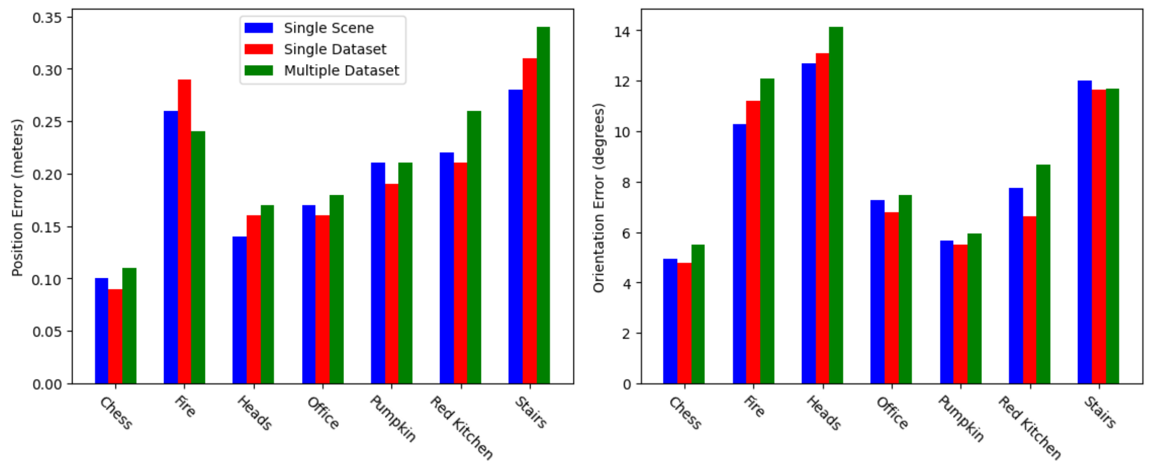
Table 2.6: We trained each 7Scenes scene independently for 1 epoch. Training of a new scene typically converges much faster when initializing from a MSPN trained on several other scenes as opposed to ImageNet pretraining. We report median translation / median orientation error.

Initialization	Fire	Heads	Stairs	Office	Red Kitchen	Chess	Pumpkin
ImageNet	0.94/82.19	1.19/148.11	0.99/92.36	0.79/44.37	0.86/19.54	0.56/30.91	0.79/48.19
MSPN	0.57/19.35	0.41/20.12	0.70/15.38	0.51/16.95	0.43/13.86	0.54/16.34	0.48/16.6

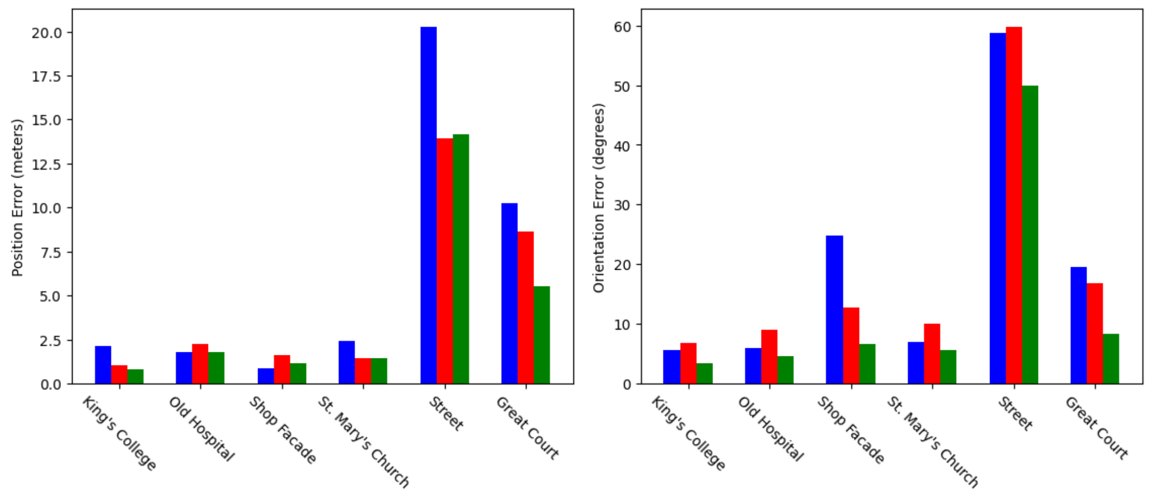
lower error after just one epoch when initialized from a MSPN. Figure 2.3 shows the full loss curve of 10 epochs for the Fire scene using both initialization methods. Overall we see that the model initialized using our method has nearly converged within only 10 epochs compared to still high error with the ImageNet pretrained models. The faster convergence allows for fine-tuning a new scene with significantly less computation. This means new scenes can be trained in a couple of minutes instead of nearly an hour (5 minutes vs 50 minutes for our experiments).

Table 2.7: Training Cambridge Landmarks scenes independently for 1 epoch.

Initialization	Great Court	Kings College	Old Hospital	Shop Facade	St. Mary's Church	Street
ImageNet	72.58/140.71	30.82/102.77	31.43/119.15	9.81/229.48	19.65/96.58	119.69/73.29
MSPN	73.16/45.71	33.75/11.05	30.23/92.15	10.19/47.54	23.45/29.46	123.81/33.79



(a) 7Scenes



(b) Cambridge Landmarks

Figure 2.4: Comparison of final position and orientation error for (a) 7Scenes and (b) Cambridge Landmarks scenes with various training methods. Single scene means that the network was trained with only images from the specified scene. Note that this is trained with our modified network without the intermediate fully connected layer. Single dataset training means all scenes from the original dataset (either 7Scenes or Cambridge Landmarks) were used for training. Similarly, multiple dataset means all scenes from both 7Scenes and Cambridge Landmarks were used. Training across both datasets does not significantly degrade performance even in the worse cases.

2.5.5 Stability Across Datasets

While many applications require accurate localization of the same camera across many different scenes, it is also possible that a method should be agnostic to camera and work across drastically different scenes. To evaluate our method on this scenario, we use all 13 scenes collectively between both datasets to train a single model. Results are shown in Figure 2.4. We show performance for single scene training, training on all scenes from the individual datasets, and training with all scenes from both 7Scenes and Cambridge Landmarks using our method. Performance is generally the same in all cases, with no significant increase in error even when training with all scenes from both datasets. These results are surprising considering the vast difference in appearance and spatial extents between the two datasets. We believe this offers more evidence that the features extracted by our approach are better for the localization task in general.

2.5.6 Scene Detection

We also verify the ability of the network to infer in which scene the image is located for end-to-end inference. During training, scene prediction is trained along with pose regression using (2.4). At inference time, the maximum probability scene from the network is used. This allows for completely end-to-end pose regression framework for an arbitrary number of scenes. We compare this to the scene oracle approach in which the scene id is known at inference time. The results are shown in Table 2.8 and Table 2.9. We found that scene prediction accuracy is very good in general. Even in the case of the Pumpkin scene where accuracy is only around 85%, the resulting change in pose prediction error compared to the oracle approach is not particularly significant. This is likely because images that are hard to correctly classify are also hard to localize, so the resulting pose will be an outlier in each case. Note that there is a difference in error even when the scene accuracy is 100% because one network is trained with the scene detector and the other without. The additional loss provided by the scene classification results in a different intermediate feature vector.

Table 2.8: End-to-end inference for 7Scenes scenes. End-to-end pose error is comparable to the scene oracle case where the scene id is known at inference time.

	Fire	Heads	Stairs	Office	Red Kitchen	Chess	Pumpkin
Scene Oracle	0.29/11.20	0.16/13.10	0.31/11.63	0.16/6.80	0.21/6.61	0.09/4.76	0.19/5.50
End-to-end	0.30/10.88	0.16/13.33	0.30/13.78	0.17/6.89	0.22/7.19	0.12/6.10	0.23/6.95
Scene Accuracy	0.9895	0.9990	1.000	1.000	1.000	0.977	0.847

Table 2.9: End-to-end inference for Cambridge Landmarks scenes. End-to-end pose error is comparable to the scene oracle case where the scene id is known at inference time.

	Great Court	Kings College	Old Hospital	Shop Facade	St. Mary's Church	Street
Scene Oracle	9.2/10.42	1.73/3.65	2.55/4.05	2.02/7.49	2.67/6.18	26.78/54.22
End-to-end	7.0/11.05	1.22/5.82	2.31/6.19	1.17/11.27	2.2/8.45	25.04/53.25
Scene Accuracy	0.9987	0.9971	1.000	0.9709	0.9981	0.6743

2.6 Conclusion

We proposed a multi-scene approach to camera pose regression, Multi-Scene PoseNet, that enables accurate localization across multiple scenes using a single network. Our approach works by learning a set of scene-specific fully connected layers for final pose regression while maintaining a shared feature extractor that is applied across all scenes. This allows a majority of the network to be shared across scenes, and drastically increases the amount of examples that can be used for training. Compared with existing techniques, our approach can support multiple scenes at a fraction of the computational cost of existing methods, and yet still achieves competitive results on standard benchmark datasets, even achieving state-of-the-art performance on some scenes. We showed that this method is stable across various datasets and number of unique scenes. Finally, we showed how our approach captures a general camera localization feature that can be used to quickly understand new scenes, allowing for faster training of unseen scenes than typical ImageNet pretraining. We believe our method can act as a foundation for many useful extensions to absolute pose regression.

Chapter 3

A Structure Aware Method for Direct Pose Estimation

Estimating camera pose from a single image is a fundamental problem in computer vision. Existing methods for solving this task fall into two distinct categories, which we refer to as direct and indirect. Direct methods, such as PoseNet, regress pose from the image as a fixed function, for example using a feed-forward convolutional network. Such methods are desirable because they are deterministic and run in constant time. Indirect methods for pose regression are often non-deterministic, with various external dependencies such as image retrieval and hypothesis sampling. We propose a direct method that takes inspiration from structure-based approaches to incorporate explicit 3D constraints into the network. Our approach maintains the desirable qualities of other direct methods while achieving much lower error in general.

3.1 Background

Camera pose estimation is a fundamental task in computer vision. Often this research is referred to as visual localization, or camera relocalization, and refers to estimating the position and orientation of a camera with respect to some predetermined reference frame. Recently, much work has explored learning-based absolute pose regression which directly regresses camera pose using a single forward pass through a neural network. These methods typically use the same basic pipeline: predict a feature embedding using a convolutional neural network (CNN), and then use features from this embedding to regress the camera pose. This works because the weights of the network implicitly capture understanding of the scene. The main differences between methods of this type are choices of feature extraction architecture or loss function. However, absolute pose regression typically performs much worse than more sophisticated alternatives. A recent study by Sattler et al. [52] found that absolute pose regression essentially interpolates between a set of learned basis poses.

In this work we partition methods into one of two classes, direct or indirect. Direct methods determine pose as a fixed pipeline in a deterministic way with no dependency on external steps such as hypothesis sampling, database querying, pose averaging, correspondence matching, or refinement. These methods are typically a single CNN architecture, such as PoseNet [34], but we show that it is possible to design a more explicit architecture that still resides in the direct pose estimation category. Indirect methods are any method that fall outside of this definition. Examples include structure-based methods such as Active Search [51] and DSAC++ [8] which compute 2D-3D correspondences and require RANSAC and refinement for final pose determination, and retrieval-based methods such as DenseVLAD [58] which require a database query. What we call direct are commonly denoted as absolute pose regression. Here we use a different term to more easily differentiate between what we call indirect methods, which have no unified name in literature.

An alternate way of thinking about the difference between direct and indirect methods is that direct methods generate pose hypotheses while indirect methods use one or more pose hypotheses for further processing. We believe this is an important distinction because improvements to direct methods can result in improvements in future indirect methods. While the more accurate methods typically fall into the indirect pose estimation category, we believe direct methods are an exciting area and wish to provide a direction for future improvement. Direct pose estimation methods have many practical benefits. Specifically, they perform relocalization in a way that is fast, deterministic, and provide a constant runtime guarantee regardless of scene complexity.

We propose an approach for solving the direct pose estimation problem using a CNN that bridges the gap between absolute pose regression and structure-based methods. Our approach has several key components. First, given a single RGB image, we simultaneously perform scene coordinate regression and monocular depth estimation to extract geometric information from the image. Then, we use the known camera intrinsics to convert the estimated per-pixel depth to 3D camera-frame coordinates, resulting in a set of 3D-3D correspondences. Our approach takes advantage of known geometric constraints to frame the problem as 3D-3D alignment of point clouds, integrating a differentiable singular value decomposition (SVD) layer with a learned per-pixel weighting scheme to compute the final camera pose.

We evaluate our method both quantitatively and qualitatively through a variety of experiments using well-known benchmark datasets. We significantly advance the state-of-the-art compared to direct pose estimation methods and decrease the gap to indirect methods.

3.2 Related Work

In this section we provide an overview of related work in pose regression using CNNs, scene coordinate regression, monocular depth estimation, and point cloud registration.

3.2.1 Pose Regression with CNNs

Directly estimating camera pose from an image using convolutional neural networks began with the introduction of PoseNet [34]. This style of approach which computes pose as the output of a CNN is known as absolute pose regression. Absolute pose regression differs significantly from traditional methods in that it does not require explicit image-level or pixel-level correspondences. Many works have explored the application of CNNs to this problem, but differences lie largely in changes to the underlying feature extraction architecture or modified objective functions [32, 33, 41, 61]. Another common approach is to add relative pose constraints on pairs of images during training instead of only absolute pose [11,65]. In general, this class of methods is attractive due to their simplicity and moderate performance, but are far from the most accurate approaches in terms of pose accuracy. Sattler et al. [52] provide an in depth review of existing work in absolute pose regression, with comparisons to more accurate structure-based and image retrieval methods.

Relative pose estimation approaches predict the pose of a query image relative to a set of database images with known pose. RelocNet [2] uses a method for camera pose retrieval using nearest neighbors with learned features. Similarly, CamNet [19] uses coarse-to-fine retrieval with two rounds of retrieval and pose refinement. AnchorNet [49] uses a set of spatial anchors, and the final pose estimate is computed by a weighted average of predicted offsets from the anchors. This method is different from others of this style in that reference poses for relative pose estimation are not based on retrieval, but rather embedded into the network either prior to or during training. The analogue to retrieval in this scenario is a set of anchor point scores, which define the weights for averaging all pose hypotheses.

3.2.2 Scene Coordinate Regression

Recent work has explored the problem of estimating the 3D location of image pixels in the scene's coordinate frame, known as scene coordinate regression. Unlike camera pose, scene coordinate information is often explicit in the image content, as points in space often look similar from varying viewpoints. Early work showed that scene coordinates could accurately be computed from RGB-D imagery in small indoor scenes using random forests [55]. Later, random forests were replaced with CNNs that only require RGB imagery without depth information [7, 8, 9].

Other methods combine image retrieval and scene coordinate regression. SANet [66] retrieves a set of images to construct a local 3D point cloud that is used for scene coordinate regression. Perspective-n-Learned-Point [43] uses image retrieval to find a single nearby image for stereo depth estimation. CNN features are used for computing pixel correspondences which, along with the stereo depth and known pose of the retrieved image, result in a set of 2D-3D correspondences.

From scene coordinates, the camera pose can be computed using perspective-n-point [21] (PnP) methods that use the 2D pixel and 3D scene coordinate correspondences to triangulate the position and orientation of the camera. While these methods produce highly accurate results, they are sensitive to noise and require the use of a robust estimator, such as RANSAC [21], and hypothesis refinement to achieve high accuracy. Though there is active research exploring the use of neural networks for generating samples for pose hypotheses, e.g., NG-RANSAC [10], our goal is to avoid this step altogether. Dang et al. [16] proposed an accurate alternative to RANSAC that predicts correspondence weighting using a CNN. However, the method of Dang et al. only works on a very specific type of problem that is solved by finding the minimum eigenvector of a matrix. There has been work on estimating point weights specifically for scene coordinate regression and pose estimation [12], but accurate results still require RANSAC.

3.2.3 Monocular Depth Estimation

Much progress has been made using CNNs for single image depth estimation [22, 23]. Importantly, recent work has shown that a learned representation for depth is sufficient for visual odometry in simultaneous localization and mapping (SLAM) systems [5, 57, 68]. Other work uses stereo matching to improve depth estimation [59]. Ranftl et al. [46] propose tools for mixing datasets during training, including a robust objective function that is invariant to changes in depth range and scale. Our work takes advantage of the success of monocular depth estimation to frame the absolute pose regression problem as an end-to-end learned alignment of corresponding point clouds.

Estimating Depth using Scene Coordinates It is possible to compute depth explicitly from scene coordinates and camera calibration parameters using PnP algorithms. A detailed overview of solutions to this problem is presented by Xiao [40]. From N 2D-3D correspondences, a common approach involves solving N systems of $\frac{(N-1)(N-2)}{2}$ equations to find the distance from each 3D point to the camera center. With five or more points, there exists a direct solution to the linear system [44]. While it is theoretically possible to perform the PnP depth estimation step in a differentiable way, it is extremely sensitive to noise. This is why the final solution in practice is found using robust estimation schemes, such as RANSAC, for accurate pose regression. Our proposed approach avoids the need for PnP and sampling altogether.

3.2.4 Point Cloud Registration

Point cloud registration is fundamental to feature-matching based image localization. The final step of pose retrieval in many PnP algorithms is absolute alignment of the 3D scene coordinates and recovered 3D camera coordinates [40]. Modern approaches rely on features extracted from neural networks for high accuracy and robustness to noise. While recent work is largely focused on synthetic or single model alignment [25, 50, 62], it has also been demonstrated that large point cloud scans can be accurately aligned using CNN architectures [39]. While we rely on point cloud registration for our method, it is a simpler case where we have explicit correspondences. Wang and Solomon [62] explore registering single object point clouds. They predict correspondences from 3D input and use the Kabsch [31] method for final alignment. We also use the Kabsch algorithm, but tackle the problem of camera pose estimation and extract 3D correspondences from an RBG image only.

3.3 Method

We present a direct pose estimation pipeline that combines scene coordinate regression, monocular depth estimation, and point cloud registration to estimate camera pose from corresponding point clouds extracted from an image.

3.3.1 Problem Statement and Formulation

We address the problem of single-image pose regression, in which we must estimate the camera pose with respect to a scene coordinate frame from a single image. The pose consists of two components, the camera orientation, R, and position, $\vec{T}$. Together, these can be used to transform 3D positions in the camera frame, A, to the scene coordinate frame, B, using $B = RA + \vec{T}$. Given corresponding points between A and B, it is possible to estimate the pose using various point cloud registration approaches. The challenge is that this estimation problem is sensitive to noise and it is difficult to find noise-free corresponding points.

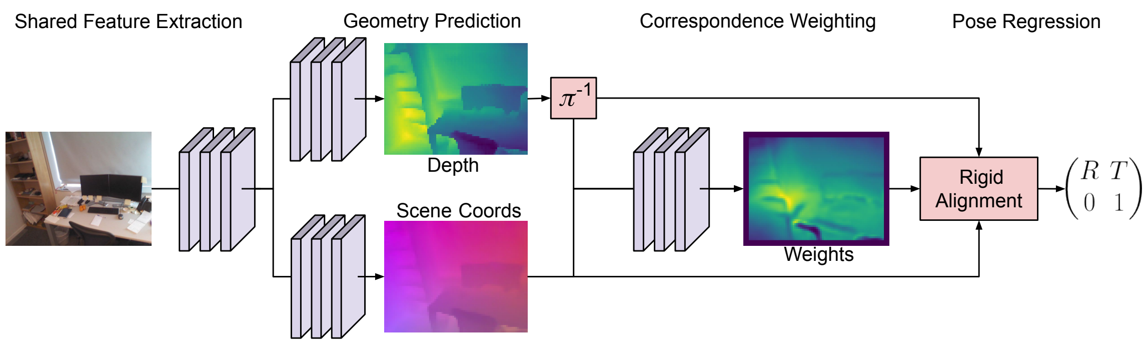


Figure 3.1: Our direct pose estimation pipeline. Given a single RGB input, we jointly estimate scene coordinates and depth. Using the known camera focal length, depth is converted to camera frame coordinates by unprojecting using π^{-1}. The second stage of our network computes a correspondence weight for each point pair. These weights are used for point normalization and final pose regression using weighted rigid alignment to estimate camera pose.

We formulate our approach such that it reduces to the problem of point cloud registration. We directly regress points in the camera and scene coordinate frames for every pixel in the image. Each pixel thus defines a pair of corresponding points between the two coordinate frames. To address the problem of noise, we score each point correspondence using a per-pixel weighting mechanism prior to point cloud registration. Our approach is implemented as a sequence of differentiable neural network layers, enabling end-to-end optimization, deterministic inference, and a simple implementation. Our network architecture is shown in Figure 3.1. We provide a detailed description of each component in the following subsections.

3.3.2 Estimating Depth and Scene Coordinates

We train a CNN to simultaneously estimate depth and scene coordinates directly from image content. We use a shared CNN backbone for initial feature map extraction and then the intermediate feature is passed to two separate sub-networks for independent depth and scene coordinate regression. For scene coordinate regression, we follow DSAC++ [8] and use a fully convolutional approach without upsampling layers. For this, we use the features from the shared feature extractor and pass them through a series of stride 1 convolutions such that the output size is $1/8$th the input image resolution. Smaller resolutions have shown to perform adequately [8] and a smaller output size is more efficient, as computation of the cross-covariance matrix used for the Kabsch algorithm requires a multiplication of two $N \times 3$ matrices.

For estimating monocular depth, we follow recent work [5,24] and perform both coarse and fine-scale depth estimation. We use the features from the shared feature extractor as input to an encoder/decoder network. We predict depth at two different scales in the network, optimizing each individually for accuracy. The largest resolution depth output is $1/8$th the size of the input such that it matches the scene coordinate resolution.

3.3.3 Estimating Camera Pose

We use the estimated depth and known camera geometry to compute 3D camera frame coordinates. For pixel i with homogeneous pixel coordinate $\vec{u_i}$ and depth d_i, the camera frame point is computed using the known camera intrinsic matrix K as follows: $\vec{p_i} = d_i K^{-1} \vec{u_i}$.

Given scene coordinates and 3D camera frame coordinates, each pixel defines a pair of corresponding points in different reference frames. Thus, the problem now becomes pose estimation from corresponding point clouds. We solve this using the Kabsch method [31] which we describe here.

Finding the optimal pose between two point sets A and B amounts to solving the following minimization problem:

$$\underset{R,\vec{T}}{\arg\min} \sum_i (\vec{b_i} - R\vec{a_i} - \vec{T})^2.$$

In the ideal case of noise-free data, we would assume all points have uniform weighting. However, directly optimizing for this results in low accuracy due to noise in the point positions, so we apply a per-point weighting term. The optimal rotation and translation can then be found by solving:

$$\underset{R,\vec{T}}{\arg\min} \sum_i w_i(\vec{b_i} - R\vec{a_i} - \vec{T})^2,$$

where w_i is a weight assigned to the point pair i.

To solve for R and $\vec{T}$, the translation component is first removed by centering both point clouds:

$$\begin{aligned} \vec{\mu_B} &= \frac{\sum_i w_i b_i}{\sum_i w_i} & \vec{\mu_A} &= \frac{\sum_i w_i a_i}{\sum_i w_i} . \\ \bar{B} &= B - \vec{\mu_B} & \bar{A} &= A - \vec{\mu_A} \end{aligned}$$

We recover R and $\vec{T}$ as follows:

$$
\begin{aligned}
USV^{\intercal} &= svd(\bar{A}^{\intercal} W \bar{B}) \\
d &= det(VU^{\intercal}) \\
R &= V \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} U^{\intercal} \\
\vec{T} &= -R\vec{\mu_A} + \vec{\mu_B}.
\end{aligned}
$$

Our output representation for rotation is unique among direct pose estimation methods. Levinson et al. [37] show how this use of SVD to construct a orthonormal matrix is the most accurate among a large variety of rotation representations for deep networks.

Intuitively, the point weights move the point towards ($w_i < 1$) or away ($w_i > 1$) from the origin. Motion towards the camera reduces the contribution of that point to the singular vectors. Likewise, motion away from the camera increases the contribution of that point. This allows for more robustness to noise without the need for hard thresholding or sampling. To generate the weights, we pass the correspondences through a scoring CNN that produces a per-pixel weighting. The input to this network is the concatenation of the output scene coordinates and unprojected camera frame coordinates computed from the output depth. The weighting is vital because the accuracy of the point cloud centroid depends on the accuracy of the points themselves. Since the estimated rotation is dependent on accurate centroid subtraction, and the estimated translation is dependent on *both* the centroid and rotation estimate, it is important to reduce the impact of the noisy points.

3.3.4 Implementation Details

We train our method in two stages, denoted as geometry and pose optimization respectively.

In the first stage, we train the depth estimation and scene coordinate regression networks jointly for accuracy. Given the scene coordinates, C, depth, D, and half resolution depth, $D_{1/2}$, we minimize the following loss function:

$$L_{geom} = ||C - \hat{C}||_1 + ||D - \hat{D}||_1 + ||D_{1/2} - \hat{D}_{1/2}||_1.$$

Notably, we do not use a validity mask during training which is common in depth and scene coordinate regression work. This encourages the network to explicitly learn areas of the image which do not have valid depth data in the training set. We train this stage for 50 epochs with the Adam optimizer [35] using an initial learning rate of $1e^{-4}$. The learning rate is reduced by a factor of 0.1 every 20 epochs. The output depth is mapped to the range

$[0, 1]$ using the *sigmoid* activation and we normalize the target depth about the mean as in CodeSLAM [5]. For each scene, we determine the mean scene coordinate and subtract this value from the target scene coordinates for optimization. At inference, the mean is added to the scene coordinate outputs. We use $\alpha = 0.5$ for weighting the loss on the half resolution depth prediction.

In the second stage, we train the weighting network with scene coordinates and camera coordinates from the previous step. We apply a *sigmoid* activation at the end of our weighting network such that weight values are in the range $[0, 1]$. We train this stage for 10 epochs with the Adam optimizer using a learning rate of $1e^{-3}$. While we do not explicitly have loss terms on the depth and scene coordinate outputs in this stage, we do allow the relevant network weights to update by using a learning rate of $1e^{-6}$. This stage is purely optimized for the accuracy of the final pose rotation, R, and translation, $\vec{T}$ with the following function:

$$L_{pose} = ||R - \hat{R}||_1 + ||\vec{T} - \hat{\vec{T}}||_1.$$

All input images are resized to 640×480, resulting in an output depth and scene coordinate resolution of 80×60.

The backbone of our network is ResNet34 [27]. The depth and scene coordinate sub-networks share the first half of the backbone and split after the second residual block. The scene coordinate regression sub-network consists of a series of 3×3 stride 1 convolutions with ReLU activations. The depth network is based on the LinkNet [14] segmentation network. The weighting network is a series of 3×3 stride 1 convolutions with ReLU activations.

3.4 Evaluation

We evaluate our method both quantitatively and qualitatively through a variety of experiments using well-known benchmark datasets.

3.4.1 Datasets

We report results on two common benchmark datasets. The 7Scenes dataset is a collection of 7 unique indoor scenes of varying size and localization difficulty. Each scene has a set of sequences containing 500 or 1000 frames of RGB, depth, and pose information resulting in 1000 to 7000 frames for training. The 12Scenes [60] dataset is a more difficult indoor dataset for absolute pose regression. It contains few training images relative to the size of the space for each scene which makes training accurate absolute pose regression models difficult, whereas structure-based methods perform very well on this dataset. For each scene in these datasets, ground-truth depth labels are found by ray-casting into structure-from-motion models using the ground-truth pose information. The pose and depth labels are used for computing ground-truth scene coordinates.

3.4.2 Quantitative Evaluation

Table 3.1 shows how our method compares to several RGB only direct pose estimation methods on the 7Scenes dataset. For "SC-conf" we report the 2D-3D, single hypothesis metric which uses only RGB input. We only report for the *heads* scene because this is all that is reported in the paper and a public implementation is not provided by the authors. For "ESAC no RANSAC" we used the scene coordinate regressors from ESAC and used the EPnP [36] algorithm for final pose estimation without RANSAC or refinement. Our method dominates all other methods in rotation accuracy. Our rotation error is often less than 70% of the next best method, and is better by at least 7% in all cases. While not as dominant for position error, we match or outperform the next best method in all but two scenarios, one of which we obtain very similar results. The only scene where our method is significantly outperformed is *stairs*. This scene is very challenging even for indirect methods [55]. Note that while our method does require depth ground-truth for training, we still show significant improvements over other methods that make use of depth for training, such as Geo-PoseNet [33]. Like all of these approaches, we make use of only the input RGB imagery without depth input at test time.

While it is clear that our approach outperforms all direct absolute pose regression methods, it is important to also compare to indirect methods that rely on techniques such as image retrieval, RANSAC, correspondence matching, etc. Table 3.2 shows how our method compares to several recent indirect methods. Our method outperforms several of these methods. It typically takes several additional techniques in order for a method to significantly outperform our direct method.

We show the cumulative histogram of errors for all images from the 7Scenes dataset in Figure 3.2 comparing our method to PoseNet, MapNet, and the scene oracle approach from ESAC [9]. Additionally, Figure 3.2 shows to the same methods for overall accuracy on the complete dataset. While the improvement from PoseNet to MapNet is minor, our method shows a significant improvement over MapNet. These comparisons better illustrate that while our method is still less accurate than the most effective single image localization methods, it provides a sizable improvement over the next best direct method.

Additional results on 12Scenes are shown in Table 3.3. These scenes are very difficult

Table 3.1: Direct pose estimation results on 7Scenes compared to other methods (median position in meters/median rotation in degrees). We outperform all methods in rotation accuracy, typically by several degrees. Our method is only outperformed in position error for 2 scenes, and the difference is minor relative to the improvements in performance overall.

Method	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Avg.
PoseNet [34]	0.32/8.12	0.47/14.4	0.29/12.0	0.48/7.68	0.47/8.42	0.59/8.64	0.47/13.8	0.44/10.44
PoseNet Learned Weights [33]	0.14/4.50	0.27/11.8	0.18/12.1	0.20/5.77	0.25/4.82	0.24/5.52	0.37/10.6	0.24/7.87
Geo PoseNet [33]	0.13/4.48	0.27/11.3	0.17/13.0	0.19/5.55	0.26/4.75	0.23/5.35	0.35/12.4	0.23/8.12
LSTM PoseNet [61]	0.24/5.77	0.34/11.9	0.21/13.7	0.30/8.08	0.33/7.00	0.37/8.83	0.40/13.7	0.31/9.85
GPoseNet [13]	0.20/7.11	0.38/12.3	0.21/13.8	0.28/8.83	0.37/6.94	0.35/8.15	0.37/12.5	0.31/9.95
Hourglass PN [41]	0.15/6.17	0.27/10.8	0.19/11.6	0.21/8.48	0.25/7.01	0.27/10.2	0.29/12.5	0.23/9.54
BranchNet [64]	0.18/5.17	0.34/8.99	0.20/14.2	0.30/7.05	0.27/5.10	0.33/7.40	0.38/10.3	0.29/8.32
MapNet [11]	0.08/3.25	0.27/11.7	0.18/13.3	0.17/5.15	0.22/4.02	0.23/4.93	0.30/12.1	0.21/7.78
MapNet++ [11]	0.10/3.17	**0.20**/9.04	0.13/11.1	0.18/5.38	0.19/3.92	0.20/5.01	0.30/13.4	0.19/7.29
Sequence Enhancement [65]	0.09/3.28	0.26/10.92	0.17/12.70	0.18/ 5.45	0.20/3.66	0.23/4.92	**0.23**/11.3	0.19/7.46
ESAC no RANSAC	0.12/2.96	0.28/7.58	1.04/60.68	0.48/9.05	0.21/4.32	0.31/6.88	0.58/10.25	0.43/14.53
SC-conf no RANSAC [12]	-	-	0.18/10.6	-	-	-	-	-
Ours	**0.08/2.17**	0.21/**6.14**	**0.13/7.93**	**0.11/2.65**	**0.14/3.34**	**0.12/2.75**	0.29/**6.88**	**0.15/4.55**

	Acc.
PoseNet	0.024
MapNet	0.052
Ours	0.116
ESAC	0.752

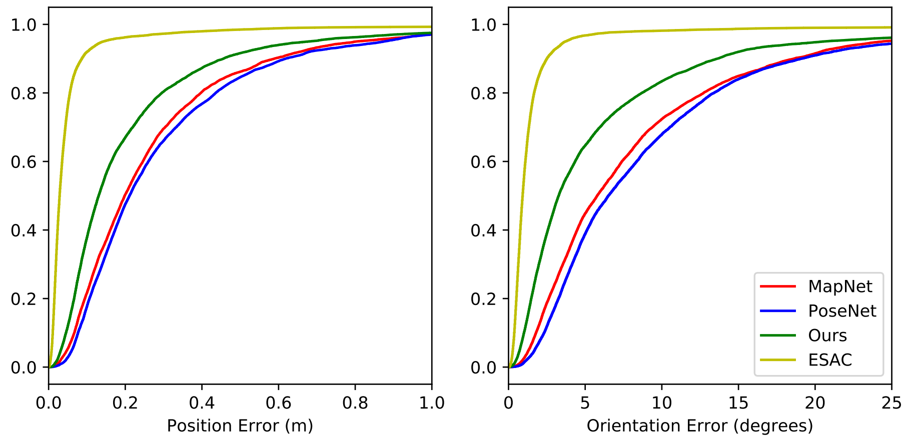


Figure 3.2: Cumulative histograms of error for all scenes from 7Scenes for several methods, truncated to 1 meter and 25 degrees error. We also report the accuracy at 5 cm and 5 degree error thresholds.

for PoseNet-style approaches, leading to very poor performance. However, our method is able to achieve accuracy very similar to PnLP, a method that performs retrieval, correspondence matching, RANSAC, and refinement.

Table 3.2: We compare to indirect pose estimation methods that rely on some combination of retrieval (Ret), image correspondence (Cor), hypothesis averaging (Avg), RANSAC (RAN), and refinement (Ref). We highlight in red instances where a method is outperformed by our approach. Results are shown as the average median error across all scenes from 7Scenes.

Method	Error	Additional Steps
Ours	0.15/4.55	
Bayesian PN [32]	0.47/9.81	Avg
MapNet+PGO [11]	0.18/6.56	Ref
DenseVLAD [58]	0.26/13.11	Ret
AnchorNet [49]	0.10/6.74	Avg
RelocNet [2]	0.21/6.73	Ret, Avg
PnLP [43]	0.12/3.93	Ret, Cor, RAN, Ref
Active Search [51]	0.05/2.46	Cor, RAN, Ref
SCoRe [55]	0.08/1.60	RAN, Ref
SC-conf [12]	0.06/3.06	RAN, Ref
CamNet [19]	0.04/1.69	Ret, Avg
SANet [66]	0.05/1.68	Ret, Cor, RAN, Ref
ESAC [9]	0.03/0.95	RAN, Ref

Table 3.3: Comparison of median position and orientation error for several methods on the 12Scenes dataset.

	PoseNet	Ours	PnLP	ESAC
Kitchen1	0.29/15.48	0.08/4.45	0.09/4.1	0.01/0.44
Living1	0.29/15.31	0.08/2.50	0.08/2.9	0.01/0.43
Kitchen2	0.21/18.18	0.08/2.96	0.10/3.7	0.01/0.46
Living2	0.31/23.58	0.09/3.13	0.10/4.7	0.01/0.40
Bed	0.57/17.85	0.07/3.87	0.12/5.7	0.01/0.46
Luke	0.35/20.07	0.12/4.82	0.14/5.5	0.01/0.59
Office 5a	0.57/14.55	0.10/5.08	0.09/3.6	0.01/0.59
Office 5b	0.47/15.49	0.09/2.57	0.10/3.7	0.02/0.59
Lounge	0.29/18.42	0.07/2.53	0.10/3.5	0.02/0.61
Manolis	0.22/17.45	0.09/3.52	0.09/3.7	0.01/0.53
Gates362	0.27/16.71	0.06/2.04	0.10/4.7	0.01/0.46
Gates381	0.37/20.52	0.12/5.02	0.11/4.4	0.01/0.67

3.4.3 Ablation Study

We evaluate several different configurations of our approach. First we evaluate the pose performance of the output depth and scene coordinates before the pose optimization step has occurred. Next, we evaluate the effect of our correspondence weighting network by evaluating pose both with and without the predicted weighting. Additionally, we show results which use a masked version of L_{geom} which only considers valid regions of the image in loss computation. Ablation study results are given in Table 3.4.

Effect of the Pose Optimization Stage First we test the pose optimization stage. This is the second stage of training that optimizes directly for pose accuracy through L_{pose}. Surprisingly, the outputs prior to this stage are reasonable. However, in many cases we see a slight decrease in median error from the pose optimization stage even when not applying the correspondence weighting, indicating that depth and scene coordinate estimates were improved.

Effect of Correspondence Weighting Next, we see the effect of using the predicted correspondence weighting. As expected, the application of the weights drastically reduces the error.This shows that the weighting network is learning to accurately segment incorrect correspondences.

Effect of Validity Mask Training It is common in dense prediction tasks such as depth estimation and scene coordinate regression to train only against "valid" data, that is, pixels where ground-truth labels are available. Our final test shows the results of applying a validity mask to L_{geom}. While performance is similar, we found that training without the

Table 3.4: We compare the effect of each component of our system for all scenes of 7Scenes. Pose opt indicates that the pose optimization stage that optimizes L_{pose} has been performed. Cor weighting indicates that the learned correspondence weights are being applied. Loss mask indicates that a validity mask was used with L_{geom} during training.

Pose Opt	Cor Weighting	Loss Mask	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
			0.36/10.72	0.29/9.06	0.23/12.69	0.15/3.81	0.21/4.92	0.17/4.10	0.35/8.15
✓			0.36/10.70	0.26/9.10	0.22/12.4	0.14/3.68	0.20/4.86	0.16/4.08	0.34/7.89
	✓		0.08/2.27	0.23/6.01	0.14/**7.75**	0.11/2.70	0.15/3.47	0.13/2.93	0.31/7.14
✓	✓		**0.08/2.17**	**0.21**/6.14	**0.13**/7.93	**0.11/2.65**	**0.14/3.34**	**0.12/2.75**	**0.29**/6.88
		✓	0.14/3.70	0.25/8.41	0.23/15.52	0.14/3.59	0.17/3.86	0.16/4.21	0.30/7.89
✓	✓	✓	0.08/2.46	0.21/**5.51**	0.16/10.67	0.11/2.75	0.18/4.17	0.14/2.99	0.29/**6.36**

validity mask results in improved performance in most cases. An interesting result to note is that when training with the loss mask it is possible to get results on par with, and in many cases much better than, many other direct pose estimation methods from Table 3.1 even without correspondence weighting. Performance in this case is better than in the case of not using a validity mask during training due to the mean subtraction of the point clouds in the alignment phase. The unmasked loss results in zeros being predicted, which while they can be easily detected and ignored by the correspondence weighting CNN, negatively effect the point cloud centroids. On the other hand, loss masking prevents zero values from being predicted. Though error still decreases in this case with the use of the predicted weights. We show example results of training with or without the validity mask in L_{geom} in Figure 3.3.

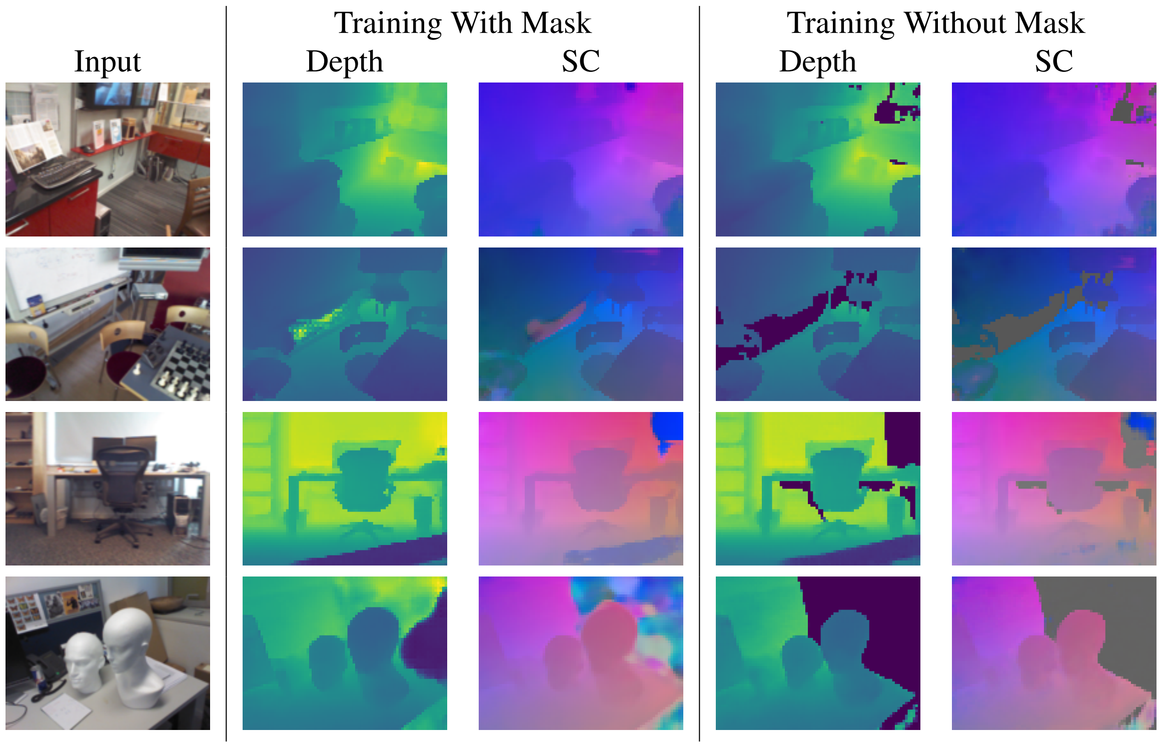


Figure 3.3: Qualitative results from training with or without a validity mask: (left) input image, (middle) output depth and scene coordinates (SC) trained using the validity mask, and (right) outputs trained without this mask. Note that these are not ground-truth labels. Training without the mask forces the network to recognize unknown areas.

3.4.4 Qualitative Results

Qualitative outputs of depth, scene coordinates, and correspondence weights are shown in Figure 3.4. To assess the quality of the estimated scene geometry and final pose estimate, the endpoint error after applying the regressed pose to the predicted point clouds is also given. The endpoint error is clamped to $1m$ for visualization purposes, with dark colors representing lower values and bright colors representing high values. Overall, it appears that the weighting is learning to recognize regions which are inconsistent with the rest of the point clouds. This is apparent as the output weights are often similar and opposite to the endpoint error after applying the pose. For example, the second row shows an example where the predicted scene coordinates are accurate, but the depth prediction is incorrect. This is correctly captured in the weighting network. The bottom row shows a difficult failure case. The depth estimates are reasonable, but the high noise present in the scene coordinates confuses the weighting network, leading to an incorrect pose.

We show qualitative results on images from the 12Scenes dataset in Figure 3.5. Compared to the 7Scenes dataset, it is less common for images to contain large regions that have no depth information during training. As such, the weighting CNN produces more varied weights in general because it can no longer depend on obviously incorrect points. This can result in seemingly strange results, such as row 5 in Figure 3.5. Note, however, that even with these strange correspondence weights, the final re-projection error is typically low.

3.4.5 Results on Outdoor Scenes

We show quantitative results on common outdoor scenes from the Cambridge Landmarks dataset [34] in Table 3.5. While our approach does not work as well in this scenario as it does for the indoor scenes from the main paper due to the low quality depth labels, it still performs competitively on all scenes, and is the best method for the Hospital scene by a large margin. On average our method is best for position error, but the ResNet based PoseNet [33] performs best on orientation error. This is surprising since PoseNet was not competitive even against other similar methods on 7Scenes. This shows the difficulty of this dataset for direct pose estimation methods. Note that this is among the most challenging scenarios for our method because of the poor quality of the depth labels generated from rendering from sparse structure-from-motion (SfM) keypoints. Due to the explicit nature of our method, utilizing a better SfM tool to generate more accurate depth images will directly lead to better performance. Examples of depth labels found in the dataset are shown in Figure 3.6. While there are many areas that have missing depth (depth=0), this is not an issue for our method as we can ignore these pixels during training. However,

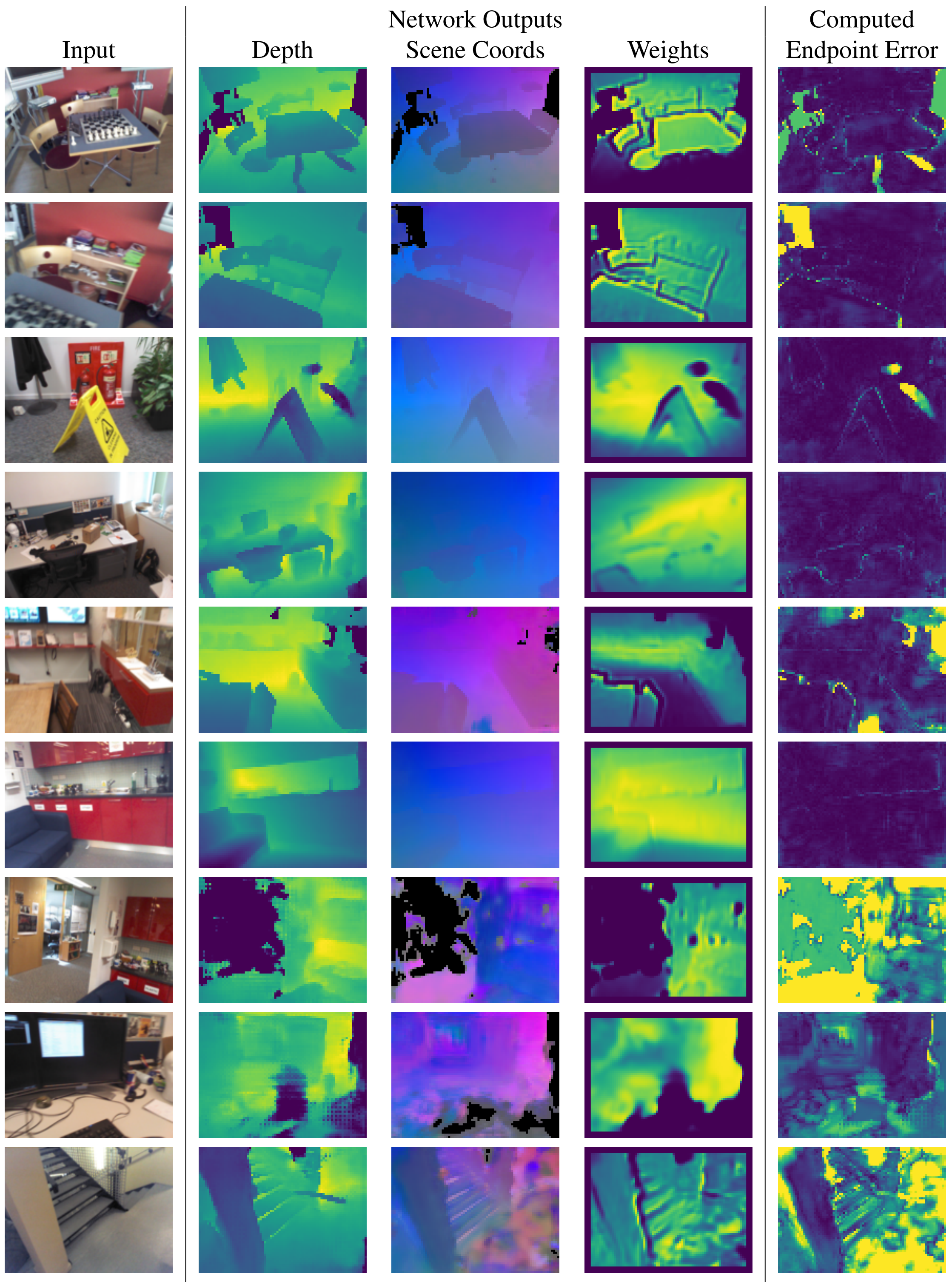


Figure 3.4: Example network outputs on the 7Scenes dataset. Depth, scene coordinates, and weights are direct outputs of our network. The endpoint error is computed by applying the regressed pose to the estimated camera coordinates and comparing to the estimated scene coordinates. The bottom three row shows failure cases with a bad final pose results caused by poor scene coordinate estimates.

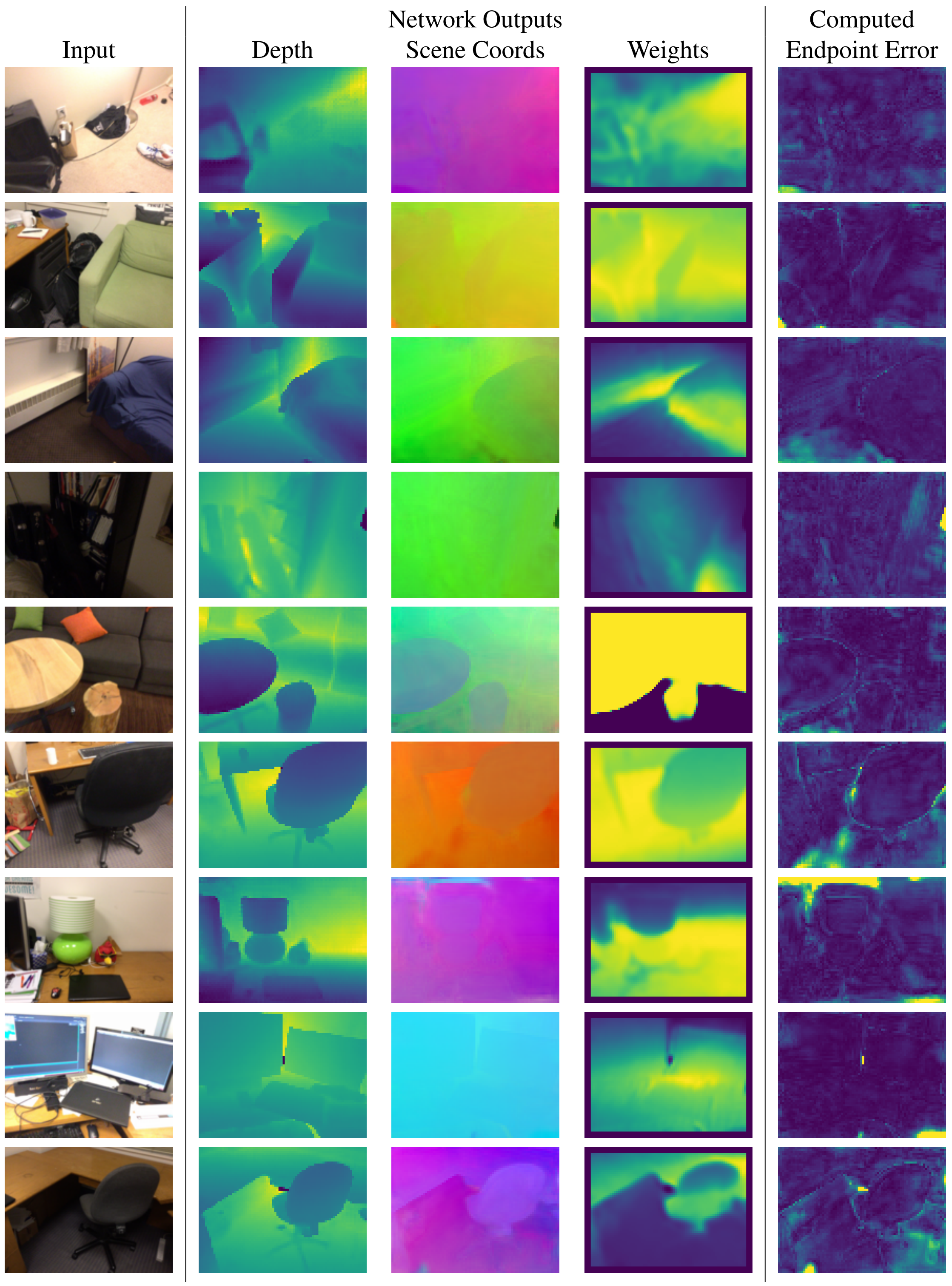


Figure 3.5: Example network outputs for 12Scenes images. Depth, scene coordinates, and weights are direct outputs of our network. The endpoint error is computed by applying the regressed pose to the scene coordinates and clamped to 1 for visualization. For depth, weights, and endpoint error, brighter means a higher value. Row labels are shown on the left for referencing in text.

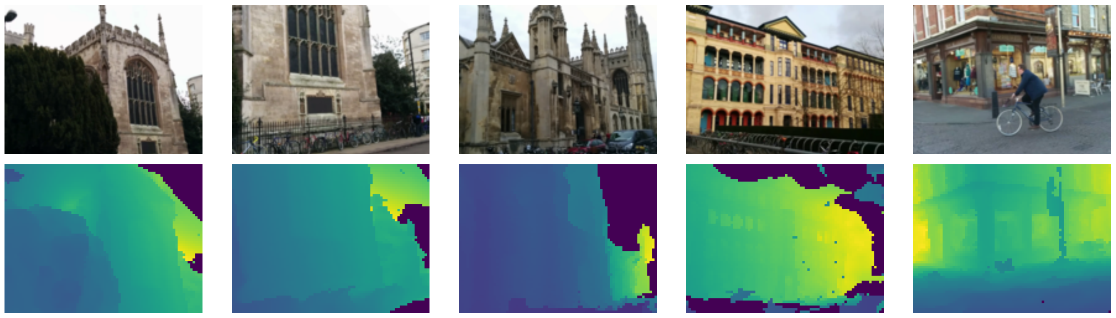


Figure 3.6: Examples of errors in depth labels for Cambridge Landmarks images. Many areas that should have invalid depth are assigned depth values. There are many incorrect depth assignments in general. Dark areas are regions with no depth label (depth=0).

Table 3.5: Direct pose estimation results on Cambridge Landmarks compared to other methods (median position in meters/median rotation in degrees).

			Sequence		
Method	College	Hospital	Shop	Church	Avg
PoseNet [34]	1.92/5.40	2.31/5.38	1.46/8.08	2.65/8.48	2.08/6.83
PoseNet Learned Weights [33]	0.99/1.06	2.17/2.94	1.05/3.97	1.49/3.43	1.43/**2.85**
Geo PoseNet [33]	**0.88/1.04**	3.20/3.29	0.88/**3.78**	1.57/**3.32**	1.63/2.86
LSTM PoseNet [61]	0.99/3.65	1.51/4.29	1.18/7.44	1.52/6.68	1.30/5.51
GPoseNet [13]	1.61/2.29	2.62/3.89	1.14/5.73	2.93/6.46	2.08/4.59
SVS-Pose [42]	1.06/2.81	1.50/4.03	**0.63**/5.73	2.11/8.11	1.32/5.17
MapNet [11]	1.07/1.89	1.94/3.91	1.49/4.22	2.00/4.53	1.62/3.64
Ours	1.19/2.16	**1.11/1.92**	0.95/6.82	**1.37**/4.45	**1.16**/3.84

there are a large number of sky pixels which are incorrectly labeled with depth, as well as erroneous depth values in general. These errors are an issue for our method because they result in incorrect supervision during training. However, as mentioned earlier, even with these labels our method performs well, and there is a clear path to improvement from better label generation alone.

Due to the poor depth quality and fewer training examples per scene, we use the masked version of L_{geom} and train for more epochs compared to indoor scenes. For the geometry optimization phase, we train for 100 epochs with an initial learning rate of $1e^{-4}$ and reduce the learning rate by a factor of 0.5 every 40 epochs. For the pose optimization phase, we train for 20 epochs with a learning rate of $1e^{-3}$ on the weighting CNN parameters and $1e^{-4}$ on the depth and scene coordinate CNN parameters. The higher learning rate on the geometry prediction parameters is similar to the the re-projection error optimization phase of DSAC++ [8] due to the error in ground-truth scene coordinate labels.

We show visualizations of network outputs on several Cambridge Landmarks inputs in

Figure 3.7. Notice that even in areas where depth and scene coordinate predictions seem good, the predicted weights tend to focus on a smaller area. This is apparent mostly in the Kings College scene, examples of which are shown in rows 2 and 8. Also, in row 6 we can see a difficult case where most of the image is a tree, leading to bad predictions. This is reflected in the weights as all predicted correspondence weights for this example are very low. Overall, even with the noisy depth labels, the weighting mechanism is able to capture which points are more reliable for final pose computation.

3.5 Discussion

While it is uncertain in general how exactly CNNs see depth in monocular images [18], it is clear in our case that both the depth and scene coordinate networks are effectively memorizing scene layout and geometry. Luckily, in the case of single image localization, this is precisely what is desired. This type of memorization, unlike PoseNet, works well for pose estimation because the memorized objects have semantic and structural meaning. Instead of interpolating between a set of learned poses, we are able to exploit the high capacity of the network to effectively store two 3D copies of the scene which can then be recalled and used for pose estimation.

We believe this is advantageous over the less explicit PoseNet approaches for several reasons. First, as we have shown, this allows for much lower error in general across a wide variety of scenes. Second, this allows for explainable pose estimation. For PoseNet-style methods, it is difficult to quantify what about an image makes it difficult for pose regression. On the other hand, the intermediate outputs of our method allow for explicit evaluation of geometric consistency. We believe this added benefit of explainability will be useful for investigating the shortcomings of direct pose estimation, leading to better architecture and optimization design choices.

3.6 Conclusion

We presented a pose estimation approach that has many of the desirable properties of PoseNet-style approaches in that it is fully differentiable, uses only feed-forward processing, and has a constant runtime, but significantly improves accuracy. Unlike PoseNet, which uses a generic CNN to perform pose estimation, our approach is composed of modules with specific geometric functions. While our approach has not achieved the accuracy of the SOTA indirect pose estimation methods, it begins to close the performance gap.

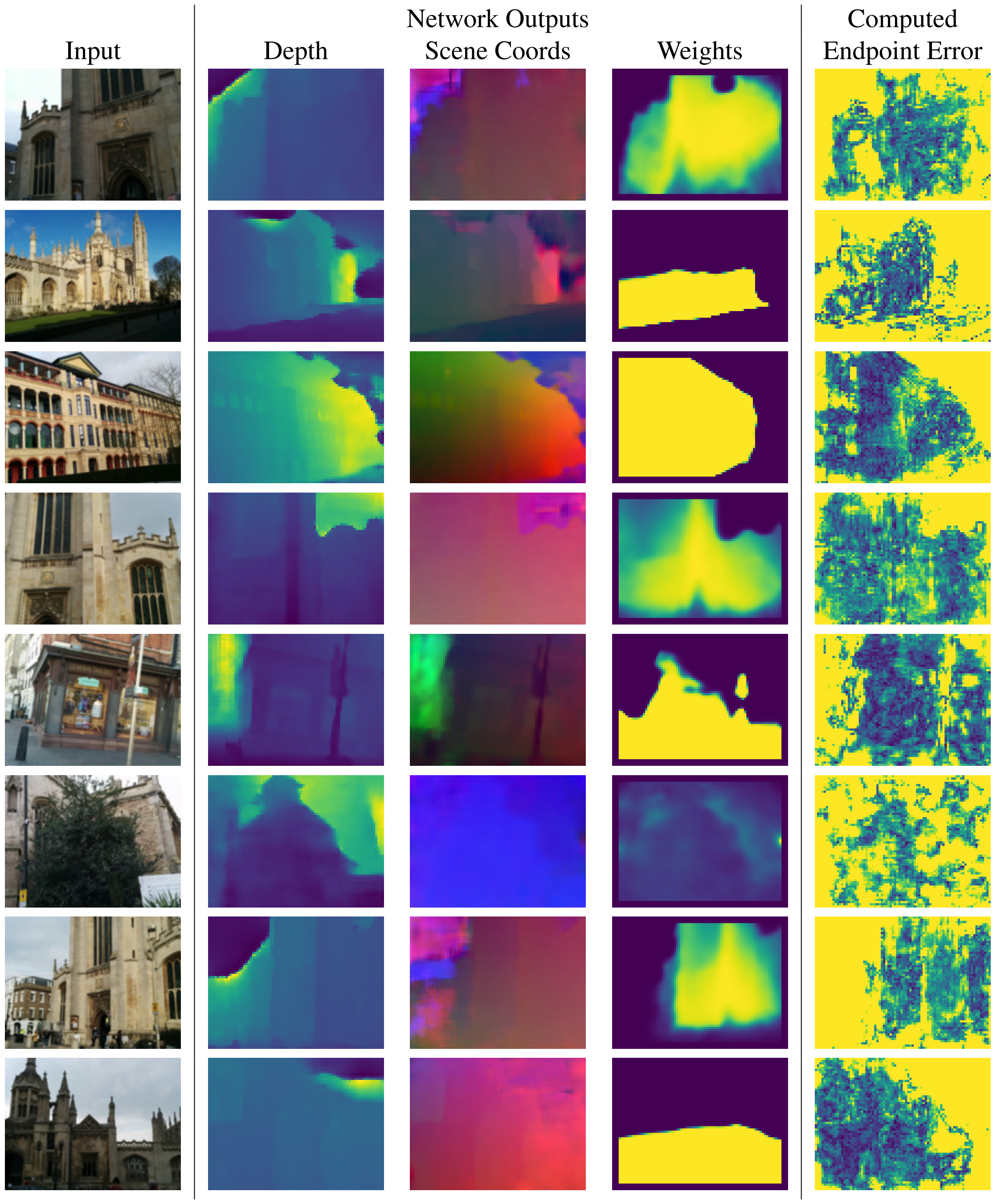


Figure 3.7: Example network outputs for Cambridge Landmarks images. Depth, scene coordinates, and weights are direct outputs of our network. The endpoint error is computed by applying the regressed pose to the scene coordinates and clamped to 1 for visualization. For depth, weights, and endpoint error, brighter means a higher value. Row labels are shown on the left for referencing in text.

Chapter 4

Factoring Out Scene Dependence in Absolute Pose Regression

We address the problem of absolute pose regression across multiples scenes using a mostly shared network. We build off of the results of Chapter 3. With a straightforward modification, our architecture explicitly partitions scene-dependent and scene-agnostic components, which means we can simultaneously train for pose estimation across multiple scenes, as well as add new scenes without retraining the entire CNN. We extensively evaluate our approach on an existing benchmark. We find that our approach is more accurate than existing absolute pose regression networks, and the scene-independent components are useful for localizing in novel scenes.

4.1 Background

The task of localizing a single image has a long history in computer vision. With applications ranging from autonomous navigation to mixed reality, it plays a vital role in several established and emerging technologies. As such, this is a highly active research area. Traditional approaches rely on feature matching at either pixel [51] or image scale [29] to determine correspondences between texture information present in the image and known 3D landmarks of the scene. Recently, much work has focused instead on applying deep convolutional neural networks (CNN) to this task [34]. This class of methods, referred to as absolute pose regression, requires no feature matching steps, instead regressing camera pose directly as the output of a CNN. This approach has many benefits over traditional methods, namely constant time performance regardless of scene size or complexity.

However, there are also several weaknesses of absolute pose regression. Perhaps the most apparent is the often low accuracy of the regressed pose relative to more expensive structure-based methods [52]. An important, but less obvious, issue is the requirement to train each network for a specific scene. Whereas traditional approaches rely on very explicit map representations such as dense point clouds or image databases, pose regression networks work by learning an implicit mapping of the scene, represented by the parameters of the network. Thus, the weights obtained from training on one scene will not in general be applicable to a new scene. While training separate networks for each scene is simple to do, it requires significant time and storage to obtain high quality pose regression models for a large corpus of scenes. Though some work focuses on the use of neural networks for scene-agnostic pose estimation with CNNs [19, 43, 66], these methods rely on image retrieval as a first step. Extending absolute pose regression to multiple scenes is a problem that has largely been ignored. Our work in Chapter 2 is the first work in this direction.

We propose a method for performing absolute pose regression across multiple scenes that is extendable to novel scenes, and has no need for an initial retrieval step or hypothesis sampling. We use the architecture proposed in Chapter 3. Our approach represents pose estimation as the rigid alignment of two 3D point sets. From a single image, we perform dense scene coordinate regression, which results in a set of 3D scene-frame coordinates, and dense monocular depth estimation, which can be combined with camera geometry to compute a set of 3D camera-frame coordinates. These point sets form explicit correspondences from which we extract camera pose by registration using the Kabsch method [31]. A benefit of our approach is that the majority of our network is explicitly scene-agnostic; only the subnetwork related to scene coordinate regression requires per-scene optimization.

Our proposed method has many advantages over existing absolute pose regression approaches: 1) it achieves lower pose error than competing single-scene absolute pose regression approaches, 2) it is structured to enable multi-scene training, instead of optimizing one scene at a time as is typical, and 3) the pretrained model can be applied to new scenes with only the need to optimize a scene-specific subnetwork.

- Application of a shared network that is used for pose regression across multiple scenes.
- A method for absolute pose regression in new scenes without the need for end-to-end pose training which maintains accuracy on previously trained scenes.

4.2 Related Work

Absolute pose regression from single images began with PoseNet [34]. This seminal work showed the feasibility of learning implicit representations of scenes to directly regress pose.

Unlike traditional approaches, absolute pose regression requires no expensive image retrieval or correspondence marching steps. While initial results exhibited relatively high pose error, performance quickly improved with network modification [33,41,61] and more robust training [11, 33, 65]. However, these works have focused on single scene localization, ignoring the problem of using a common network to localize across multiple scenes at once. In Chapter 2 we proposed a method that extends the standard PoseNet architecture to multiple scenes. However, the trained models do not generalize to novel scenes. In this work, we propose a method for multi-scene absolute pose regression that naturally extends to new scenes.

Alternatives to absolute pose regression address this problem of scene-independence implicitly. By changing the problem to image retrieval followed by relative pose estimation, localization can be performed in any environment as long as it is represented in the image database. An extremely effective approach that follows this pattern is CamNet [19], which retrieves similar images based on a CNN image feature and uses the retrieved images along with the query to perform relative pose estimation with a Siamese CNN architecture. This approach achieves accuracy on par with traditional geometric approaches, but requires two rounds of image retrieval. This, along with a forward pass through the CNN for each retrieved image, makes this very expensive compared to the single forward pass of absolute pose regression. Alternative methods [43,66] use the retrieved images to perform scene coordinate regression for each pixel of the input image. Once the scene coordinates are found, the 2D-3D correspondences are used with the perspective-n-point (PnP) algorithm [21] to determine the camera pose.

Scene coordinate regression is currently among the state of the art for single image localization with a CNN. Scene specific approaches such as DSAC++ [8] map each pixel of the input image directly to 3D scene coordinates using a CNN. Similar to absolute pose regression, the CNN effectively learns a functional map of the scene, which is then recalled during inference. However, we differentiate these methods from absolute pose regression because scene coordinates do not provide pose directly, and PnP with RANSAC [21] must be applied to get the final pose. Because of this, this method is relatively slow, often taking $> 100ms$ for final pose determination, compared to only around $5ms$ for an absolute pose regression method such as PoseNet. Our method is functionally equivalent to a CNN alternative to PnP algorithms. Unlike traditional PnP algorithms, we use a CNN to compute per pixel depth from both the image and estimated scene coordinates. Thus, we view or approach as an absolute pose regression network that uses geometrically meaningful intermediate features.

Metric correct depth estimates from a single image have been used successfully for visual odometry and SLAM [15, 57, 67]. These works show that depth estimates from a CNN are good enough to perform accurate pose estimation. Unlike scene coordinates, depth is independent of camera location, and a single monocular depth CNN can be trained for images from arbitrary environments. By jointly learning depth and scene coordinates, we effectively turn absolute pose regression into a point cloud alignment problem with correspondences. Wang and Solomon [62] show that this problem can be effectively solved in an end-to-end CNN setting using the Kabsch method [31].

4.3 Approach

We describe our single image localization approach. This network was first proposed in Chapter 3, but we review it here and discuss the scene dependence of each component. We factor pose regression into three components, (1) dense scene coordinate regression, (2) dense monocular depth estimation, and (3) point cloud alignment. Both point clouds used for registration are estimated from a single image, with no need for multiple image inputs or image retrieval for correspondence estimation. We begin by giving an overview of our network architecture, then provide a detailed discussion of each component.

4.3.1 Architecture

An overview of our approach is shown in Figure 4.1. Our network is divided into three major components. The first performs scene coordinate regression. These scene coordinates, along with the input image and camera geometry, are then used to estimate camera-frame coordinates. This is essentially monocular depth estimation followed by a perspective unprojection step. Finally, we compute the final camera pose estimate through rigid point cloud alignment. Each of these steps is modeled by differentiable network components, allowing for end-to-end optimization for camera pose.

4.3.2 Scene Coordinate Regression

Scene coordinate regression computes a dense map of 3D scene-frame coordinates from pixels in the input image. This has been studied extensively in recent years [7, 8, 9, 55]. Because the correspondence between image texture and 3D position in space is largely stable across changes in view direction, this is effectively a landmark detection task. Given a single image, the network is optimized directly to regress the dense, 3 channel image, where each pixel of the output represents the (x, y, z) scene-frame coordinate of the corresponding input pixel. Given that not all input pixels have known scene coordinate labels,

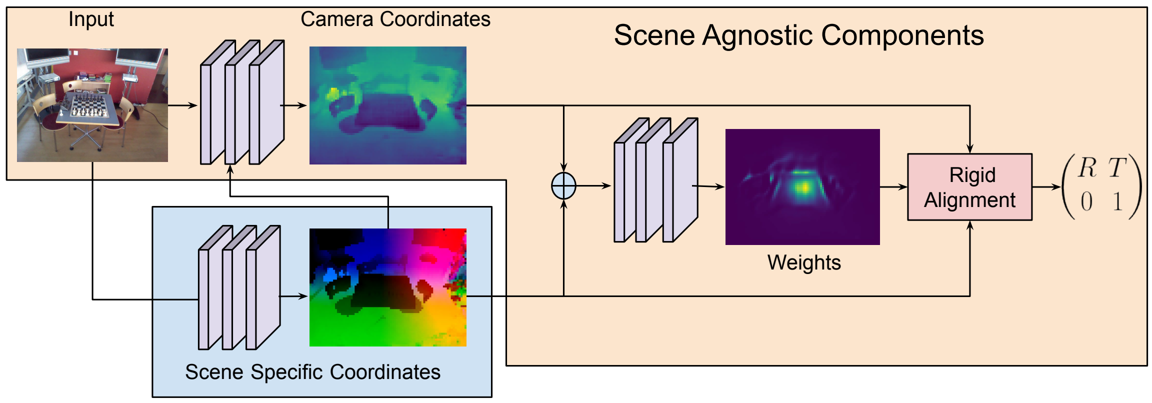

Input
Camera Coordinates
Scene Agnostic Components
Rigid Alignment
$\begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}$
Weights
Scene Specific Coordinates

Figure 4.1: Our absolute pose estimation pipeline. The scene coordinate regression network is the only component that is dependent on the scene. All other parts of the network, namely depth estimation, correspondence weighting, and rigid alignment are generic and are applied to all scenes.

the network is optimized using a weighted robust loss [9]:

$$x = M||S^2 - S'^2||_2 \tag{4.1}$$

$$L_s(x) = \{ \begin{array}{cc} x & x \leq \alpha \\ \sqrt{\alpha x} & x > \alpha. \end{array} \tag{4.2}$$

The weighting mask, M, is set to zero for pixels with no label and one otherwise.

We use the CNN proposed in ESAC [9] for this work. Since scene coordinates are a property of a unique reference frame, each scene requires a separate network. While a scene-agnostic scene coordinate regression CNN was proposed by Yang et al. [66], their method requires an image retrieval step, which we wish to avoid. The scene coordinate CNN takes as input a $480 \times 640 \times 3$ RGB image and produces a lower resolution $60 \times 80 \times 3$ scene coordinate map.

4.3.3 Single Image Depth Estimation

We use the original RGB image, along with the estimated scene coordinates from the previous section, to predict camera-frame coordinates. Camera-frame coordinates are the point cloud computed from unprojecting depth, using camera focal length, f, and optical center, (c_x, c_y). For a pixel with coordinates (u, v) and depth d, the camera frame coordinate p_c is computed as:

$$p_c = \frac{d}{f} \begin{pmatrix} u - c_x \\ v - c_y \\ f \end{pmatrix}. \tag{4.3}$$

These coordinates are independent of any scene reference frame, instead depending only on the relative position of the observed geometry to the camera. Thus, we use a single camera-frame coordinate regressor for all scenes.

The backbone of this network is a modified LinkNet [14]. We modify the network to concatenate scene coordinate information into feature maps after the third downsampling block and the second upsampling block. We chose these locations to concatenate scene coordinate information because these correspond to the feature maps that match the resolution of the regressed scene coordinates. Instead of concatenating the true scene coordinates, we first subtract their mean. This removes coordinate frame specific information, while still providing relative depth information to the network. Because we only need depth at the same resolution as the scene coordinate image, the final two upsampling blocks are removed. We place a *sigmoid* activation on the final layer to output a mean normalized depth as in CodeSLAM [5]. We optimize the depth network using a weighted L1 loss. Given ground truth depth D, network prediction D', average depth a, and validity mask, M:

$$L_d = \frac{\sum_{pixels} M||D - (a/D' - a)||_1}{\sum_{pixels} M}. \tag{4.4}$$

4.3.4 Camera Pose from Point Estimates

We follow the the approach from Section 3.3.3. We give a bried overview here. Given the predicted scene coordinates and the camera-frame coordinates obtained from unprojecting the estimated depth, we perform registration using the Kabsch method [31]. This is possible because the pixel coordinate maps produce direct correspondences between scene coordinates and camera-frame coordinates. Since all components of Kabsch are differentiable, we can optimize our network in an end-to-end manner directly for pose.

After both scene coordinate and camera-frame coordinate networks are trained, we used the Kabsch algorithm for final pose training. For ground-truth rotation matrix R and position vector $\vec{T}$, with corresponding network predictions R' and $\vec{T}'$, we optimize the following objective

$$L_{pose} = ||R - R'||_1 + \beta||\vec{T} - \vec{T}'||_1,$$

where β is a loss weighting term.

4.4 Evaluation

We evaluate our method for absolute pose regression across multiple scenes from a common benchmark dataset. We first evaluate in the single scene scenario to compare against existing absolute pose regression methods. Next, we show performance when using the shared network architecture across multiple scenes. Finally, we show how the scene-independent components of our method can perform accurate pose estimation in novel scenes, comparing to traditional perspective-n-point algorithms. We report all error metrics as median position/orientation error in meters/degrees.

4.4.1 Datasets

We evaluate on the common 7Scenes [55] dataset for single image localization. 7Scenes contains seven unique scenes of varying difficulty and spatial size. Each scene contains thousands of training images taken across different video sequences. For every scene, all frames contain RGB color, full resolution depth, and 6 degree-of-freedom (DOF) pose information computed using an accurate structure-from-motion (SfM) method. For this task, the raw depth maps were not used. Instead, the depth is generated using the ground-truth poses along with the dense SfM model to ray cast accurate depth values for each pixel of the RGB image. We use the labels provided from the official ESAC source code [9]. Pixels with valid depth ($\text{depth} > 0$) are used to produce a per-pixel validity mask used during training for loss weighting.

4.4.2 Single Scene Pose Regression

First, we evaluate our proposed approach for absolute pose regression in individual scenes. In order to compare against other state-of-the-art methods, we train a single network for each scene individually. Results are shown in Table 4.1. This is similar to results from Chapter 3, but we use a different network so we report the results here. For reference we also report the results from Chapter 3 to show that the choice of network is not of significant importance. Overall, our method performs very well against these other approaches. In many cases, we significantly outperform the next best method in both position and orientation error. Even in the worst case, we outperform nearly all other methods. We attribute this to the inclusion of explicit geometric reasoning in our approach.

Table 4.1: Absolute pose regression results on 7Scenes compared to other methods (median position in meters/median rotation in degrees). We train our method individually on each scene for fair comparison. In most cases, our method is the best.

				Sequence				
Method	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Avg
PoseNet [34]	0.32/8.12	0.47/14.4	0.29/12.0	0.48/7.68	0.47/8.42	0.59/8.64	0.47/13.8	0.44/10.44
PoseNet Learned Weights [33]	0.14/4.50	0.27/11.8	0.18/12.1	0.20/5.77	0.25/4.82	0.24/5.52	0.37/10.6	0.24/7.87
Geo PoseNet [33]	0.13/4.48	0.27/11.3	0.17/13.0	0.19/5.55	0.26/4.75	0.23/5.35	0.35/12.4	0.23/8.12
LSTM PoseNet [61]	0.24/5.77	0.34/11.9	0.21/13.7	0.30/8.08	0.33/7.00	0.37/8.83	0.40/13.7	0.31/9.85
GPoseNet [13]	0.20/7.11	0.38/12.3	0.21/13.8	0.28/8.83	0.37/6.94	0.35/8.15	0.37/12.5	0.31/9.95
Hourglass PN [41]	0.15/6.17	0.27/10.8	0.19/11.6	0.21/8.48	0.25/7.01	0.27/10.2	0.29/12.5	0.23/9.54
BranchNet [64]	0.18/5.17	0.34/8.99	0.20/14.2	0.30/7.05	0.27/5.10	0.33/7.40	0.38/**10.3**	0.29/8.32
MapNet [11]	0.08/3.25	0.27/11.7	0.18/13.3	0.17/5.15	0.22/4.02	0.23/4.93	0.30/12.1	0.21/7.78
MapNet++ [11]	0.10/3.17	0.20/9.04	**0.13**/11.1	0.18/5.38	0.19/3.92	0.20/5.01	0.30/13.4	0.19/7.29
Sequence Enhancement [65]	0.09/3.28	0.26/10.92	0.17/12.70	0.18/ 5.45	0.20/**3.66**	0.23/4.92	**0.23**/11.3	0.19/7.46
Ours	**0.08/2.07**	**0.19/6.09**	0.16/**10.98**	**0.12/3.08**	**0.15**/4.15	**0.13/3.74**	0.30/11.00	**0.16/5.87**
Ours (Chapter 3)	0.08/2.17	0.21/6.14	0.13/7.93	0.11/2.65	0.14/3.34	0.12/2.75	0.29/6.88	0.15/4.55

Table 4.2: Results on multi-scene training. Each method was trained on all scene simultaneously. We also compare against a variant of our method, *Ours* ($RGB \rightarrow D$), which does not use scene coordinates as input for depth estimation, only the image.

				Sequence				
Method	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Avg
PoseNet (all)	0.15/4.85	0.28/13.13	0.30/**11.54**	0.23/6.34	0.29/5.34	0.29/6.98	0.35/10.63	0.27/8.40
MSPN	0.09/4.76	0.29/10.50	**0.16**/13.10	0.16/6.80	0.19/5.50	0.21/6.61	0.31/11.63	0.20/8.41
Ours (RGB $\rightarrow D$)	0.08/2.23	0.19/5.93	0.20/13.48	0.14/3.61	0.15/4.06	0.14/3.87	**0.30/8.73**	0.17/5.99
Ours	**0.07/2.00**	**0.15/5.19**	0.19/12.29	**0.12/3.40**	**0.14/4.00**	**0.13/3.69**	0.39/9.99	**0.17/5.79**

4.4.3 Multi-scene Pose Regression

Next, we evaluate our method on pose regression across multiple scenes. Note that we assume that the general location of the camera is known, i.e., the camera is in the Chess scene, and instead focus on the ability to use a shared CNN for absolute pose regression. Other work has shown that even simple scene classifiers can achieve extremely high accuracy [9,55], and these methods could easily be incorporated into our work for course-to-fine localization if desired.

We compare against two multi-scene absolute pose regression baselines. The first is a single PoseNet trained on all scenes at once, which we refer to as *PoseNet (all)*. For this, we use the PoseNet architecture and training proposed by Kendall et al [33]. It uses a ResNet-34 feature extractor and learned loss weighting. We also use the MSPN approach from Chapter 2. This second baseline uses a shared backbone amongst all scenes and scene-specific final regression layers. Table 4.2 shows the results of this experiment. Our approach is, in most cases, a significant improvement over the baselines.

We also show the results of our method if we choose to not use scene coordinates as input to the depth estimation network, denoted as Ours ($RGB- > D$). In this case, we perform standard monocular depth estimation using only the RGB image input, as in Chapter 3. While this version of our approach performs better in some cases, overall adding scene coordinates improves performance. We believe this is because the scene coordinates provide the network with estimates of relative depth, which aid in absolute depth prediction.

4.4.4 Evaluation on Novel Scenes

A benefit of our method is the ability to easily use the shared portion of the network on new scenes without the need for retraining. To test this, we first train a network on 6 out of 7 scenes, and evaluate on the held out scene. For the held out scene, only the scene-specific scene coordinate regression network has been trained. Results are shown in Table 4.3. In the case of the MSPN finetuned baseline, the shared feature extractor is frozen, and only the scene-specific regression layers are optimized. This is done so that performance on previously trained scenes is not lost. From the very high error in this MSPN baseline, it is clear that the shared feature extractor is not good enough for localizing in new scenes, even when the final scene-specific layers are optimized for the new scene. However, because our method is based on explicit geometric predictions, the shared network can still be effectively applied to new scenes. In this sense, our approach is a CNN based alternative to general PnP algorithms. To show this, we also compare to the EPnP algorithm [36] applied to the full scene-coordinate output. This method, unlike the common P3P methods, takes an arbitrary number of points, making it a fair comparison as a non-RANSAC based approach. We also tried to use a differentiable implementation of the EPnP algorithm to train a weighting network. We tried the method of Dang et al. [17] as well as the weighting network used for our method, but neither method converged. While performance for our method is reduced compared to the case where the evaluated scene was part of the training set, it is still comparable to several single-scene pose regression approaches, such as the original PoseNet [34] and LSTM PoseNet [61]. More importantly, it is competitive with EPnP, especially in the difficult scenes such as Heads and Stairs. Note that since we do no finetuning of the shared network components with the new scene, the pose error on previously trained scenes is unaffected.

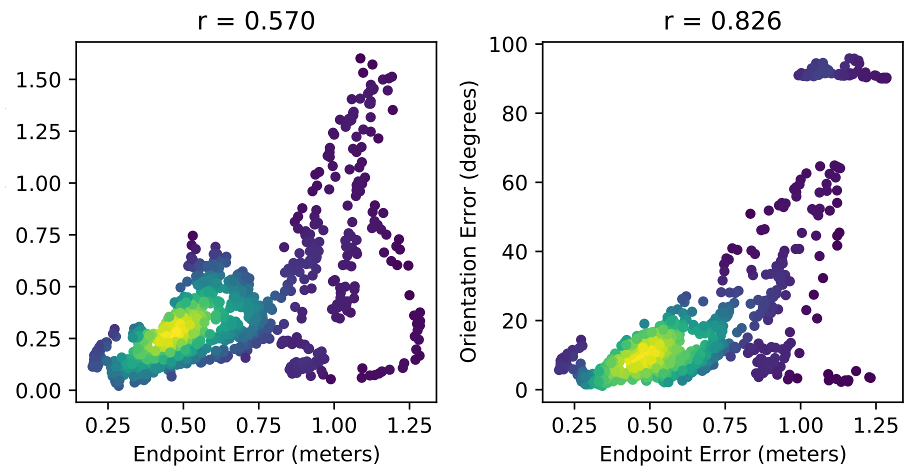
Table 4.3: Pose error results for held out scenes (median position in meters/ median orientation in degrees). In this case, each scene was evaluated using depth and weighting networks trained without examples from the given scene. Our method significantly outperforms MSPN, and manages to beat EPnP in many cases. For reference, we show results from the original PoseNet paper, trained on single scene. In nearly all cases, our method outperforms PoseNet even though PoseNet is trained specifically for the single scene.

	Chess	Fire	Heads	Office	Pumpkin	Red Kitchen	Stairs	Avg
PoseNet [34]	0.32/8.12	0.47/14.4	0.29/12.0	0.48/7.68	0.47/8.42	0.59/8.64	0.47/13.8	0.44/10.44
MSPN (finetune)	0.82/23.28	0.76/31.39	0.44/23.15	0.98/ 45.69	0.76/29.86	1.32/33.37	0.69/32.95	0.82/31.38
EPnP [36]	**0.12/2.96**	**0.28/7.58**	1.04/60.68	0.48/9.05	**0.21/4.32**	0.31/6.88	0.58/10.25	0.43/14.53
Ours	0.24/6.58	0.32/10.40	**0.29/18.97**	**0.28/6.31**	0.35/7.81	**0.28/6.51**	**0.37/8.28**	**0.30/9.27**

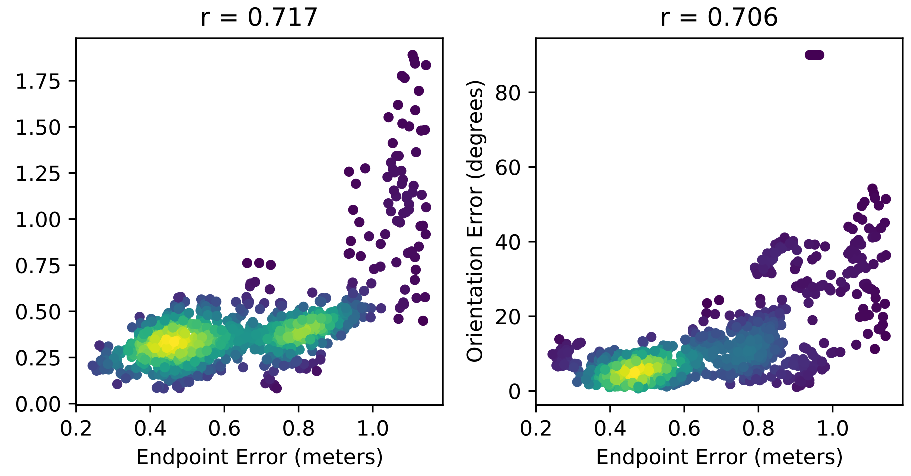
4.4.5 Endpoint Error vs. Pose Error

A major benefit of our method is the ability to estimate pose quality without the need for ground truth pose data. To do this, we compute the Pearson correlation coefficient r between the average endpoint error and estimated pose. The endpoint error is computed by taking the average euclidean distance between the estimated scene coordinates and the camera frame coordinates transformed with the estimated pose. We did this using all test samples from the entire datset. Results are shown in Figure 4.2. We also compare to the more accurate PnP RANSAC method, were we compute the pixel reprojection error instead of endpoint error. For both cases of our method, there is a moderately strong correlation between the final endpoint error with both the position error and rotation error. The PnP RANSAC approach has effectively zero correlation, and the final reprojection error is not informative about the error of the estimated pose.

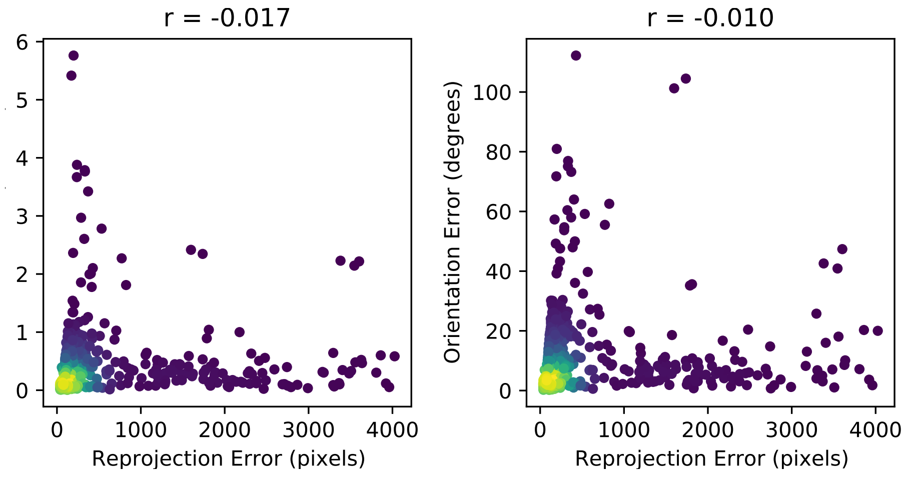
Since the correlation between endpoint error and pose error is relatively high for our method, we can use this relationship to estimate the quality of the pose. In other words, a high average endpoint error leads to less confidence in the final pose prediction. Our method is the only absolute pose regression method where this is possible, as all other methods in literature simply use a black box CNN approach which does not have intermediate features that can naturally be used for this kind of analysis.



(a) Ours, In Training Set



(b) Ours, Held Out



(c) PnP RANSAC

Figure 4.2: Correlation between endpoint error or reprojection error and pose error. We show results for (a) our method when trained on the scene, (b) our method when the scene is not in the training set, and (c) the typical PnP with RANSAC used on the scene coordinates. For both the in-set and out-of-set cases of our method, there is a moderately strong correlation between endpoint error and pose error. This is not the case for the PnP method.

4.4.6 Depth Accuracy

A key part of our method is the choice of depth estimation network. While we could have chosen a large network and trained it for generic depth estimation, we instead chose a more shallow network and trained on a per-scene basis for pose estimation. Table 4.4 shows the average depth error for each scene. We compare the depth estimation accuracy in the case of 1) training with a single scene, 2) training with all scenes, and 3) holding out the scene. This last case tests the potential ability of our depth networks to transfer to other scenes. We report mean absolute error, as well as depth accuracy values for different error thresholds. As expected, we typically observe a gradual decline in depth estimation quality as we move from the single scene case where much of the scene structure can be memorized, to the held out case, where no information about the scene was observed during training. Also, we show that a high percentage of pixels have a depth error of less than 0.125 meters, so we believe our simple network is sufficient for this task.

4.4.7 Qualitative Results

Examples network outputs are shown in Figure 4.3. For the given input image, we show the network output scene coordinates, as well as the depth and correspondence weights for three different inference scenarios: one where we trained only on the single scene, one where the scene was part of the training set, and one where the scene was held out of training. Notice that eh depth is sharpest in the single scene training scenario and noisiest in the held out scenario. However, the depth and weights estimated by the network are similar in all cases.

4.4.8 Implementation Details

We use pretrained ESAC networks, obtained from the official source repository [6], for our scene coordinate regression branches. To train the depth branch, we optimize using the Adam optimizer for 30 epochs with an initial learning rate of $1e^{-3}$. The learning rate is decreased by a factor of 10 after every 10 epochs. We use the mean depth value for the entire 7Scenes dataset for depth normalization. The weighting network is trained for 10 epochs with a learning rate of $1e-3$. We set $\beta = 1$ for L_{pose}.

Table 4.4: Depth error on each scene in three scenarios: training with only on scene, training with all 7 scenes, and holding one scene out during training. Results are reported in meters.

	Sequence						
Method	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
Single Scene							
Abs Error	0.2318	0.1954	0.1491	0.2367	0.2244	0.2570	0.3515
$\delta < 0.5^3$	0.4883	0.4743	0.6206	0.3651	0.4164	0.3616	0.3260
$\delta < 0.5^2$	0.7482	0.7461	0.8371	0.6554	0.7030	0.6349	0.5610
$\delta < 0.5$	0.8968	0.9364	0.9385	0.9011	0.9261	0.8870	0.7959
All Scene							
Abs Error	0.2271	0.1842	0.1730	0.2325	0.2591	0.2654	0.4374
$\delta < 0.5^3$	0.4909	0.4945	0.5817	0.3876	0.3599	0.3580	0.2559
$\delta < 0.5^2$	0.7499	0.7710	0.7842	0.6694	0.6607	0.6280	0.4535
$\delta < 0.5$	0.8980	0.9493	0.9216	0.9029	0.9044	0.8787	0.7196
Held Out							
Abs Error	0.2768	0.2157	0.2425	0.3333	0.3549	0.3386	0.4800
$\delta < 0.5^3$	0.3759	0.4060	0.3014	0.2212	0.1996	0.2615	0.2316
$\delta < 0.5^2$	0.6372	0.6893	0.5940	0.4526	0.4479	0.4814	0.4173
$\delta < 0.5$	0.8666	0.9156	0.9119	0.8048	0.8251	0.7910	0.6766

4.5 Conclusions

We proposed a novel approach for absolute pose regression that represents the problem as the rigid alignment of point sets. From a single image, we perform scene coordinate regression and depth estimation, producing corresponding point sets from which we extract camera pose via differentiable registration. An advantage of our approach is that the majority of the network is scene-independent, enabling training across multiple scenes. To add new scenes, there is only the need to optimize a scene-specific subnetwork, leaving the majority of the network untouched. We evaluated our approach on a standard benchmark dataset, demonstrating improved performance relative to baseline approaches. We also show that our approach can be useful for accessing the quality of the pose estimate directly by computing endpoint error without needing ground truth pose.

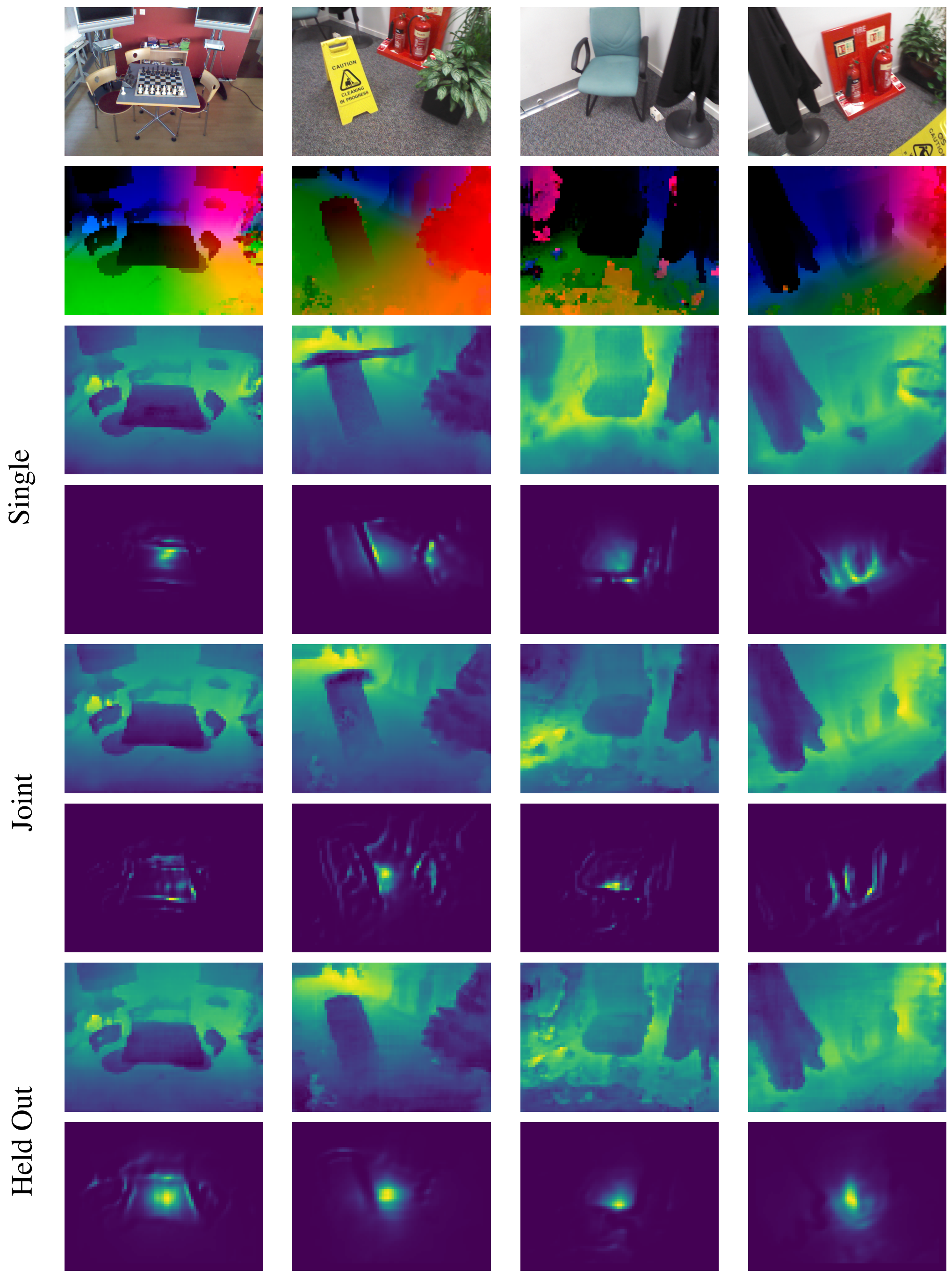


Figure 4.3: Example network outputs in different scenarios. Given the input image (first row), we show the estimated scene coordinates (second row), as well as depth and correspondence weights for single scene trianing (rows 3 and 4), joint training of all scenes (rows 5 and 6), and held out evaluation (rows 7 and 8). The depth results are slightly noisier for held out training, though all results are similar.

Chapter 5

Discussion

Estimating camera pose from a single image is a challenging problem. While methods for image retrieval and scene coordinate regression have had great success, direct regression of the pose without further refinement steps has yet to catch up. Our thesis focused on improvements to absolute pose regression. We proposed a novel method for absolute pose regression. Our method improves pose error metrics across a variety of scenes. Also, we show how to share weights so that a large portion of the network can be shared across scenes.

5.1 Findings

In Chapter 2 we explored how to use a standard PoseNet [34] in a multi-scene setting. We showed that a majority of the network can be shared, with only the final layer needing to be unique for each scene. This modification allows for faster training and greatly reduced model size while allowing for inference across multiple scenes without an increase in median error. Specifically, compared to baseline methods such as PoseNet and MapNet [11], the proposed approach shows roughly equivalent error in many cases while uses on the order of 10 times fewer parameters. However, this work also highlights an issue with APR in which the trained network can not generalized to new scenes. After training in a multi-scene setting, it is not enough to simply train the final layer for a new scene. Instead the full network must be optimized to localize within new scenes.

In Chapter 3 we focused on a new approach for absolute pose regression. While previous APR methods simply used CNNs designed for image classification, we describe a network that is explicitly designed for pose estimation. Unlike the typical PoseNet architecture, we use geometric intermediates. We jointly estimate single image depth as well as scene coordinates to transform the problem into corresponding point cloud alignment.

Aligning corresponding point clouds has a closed form and differentiable solution which fits nicely into an end-to-end network for training. While several methods exist that use scene coordinate estimation and full image depth [12, 43], this is the first work to perform scene coordinate regression and depth estimation jointly from a single image, as well as the first to use both in a absolute pose regression method.We show how this significant departure from previous APR work leads to much better pose estimates. Our method is typically much more accurate than the next best APR method and in some cases outperforms indirect methods that require multiple hypothesis sampling or post processing techniques for final estimation.

In Chapter 4 we combine concepts from Chapters 2 and 3. We explore how the improved architecture for absolute pose regression offers an explicit partitioning of network parameters that are scene-dependent or scene-agnostic. We show that you can train using multiple scenes simultaneously without a significant impact in error at inference. However, unlike with Multi-scene PoseNet, the pretrained network can be applied to a new scene by simply training the scene dependent network for the new scene. While the performance in this novel scene scenario is worse than if the scene was used for training, it is still better in most cases than the scene specific PoseNet. Importantly, we make use of pretrained, publicly available ESAC [9] networks. Taken with the results from Chapter 3, this shows that the specific architecture and training strategy are not significant, but rather the key idea of using estimated depth and scene coordinates along with the Kabsch [31] method are what result in improved absolute pose regression over black-box CNNs.

5.2 Future Work

This thesis proposed several methods for absolute pose regression. Our work culminated in a geometric architecture for APR that can be used for camera pose estimation across multiple scenes. There are several possible future research directions for extending this work. One line of research that has become popular is pose estimation over video sequences. KFNet [69] proposes a method for pose estimation through video by using scene coordinates, optical flow, and a Kalman filter. A similar technique could be used with our approach to potentially improve results on video frames. There are also many possible avenues to explore around the concept of uncertainty estimation. We make use of uncertainty estimation in the correspondence weighting from Chapters 3 and 4, but we could also explore uncertainty over pose [32], depth [5], or scene coordinates [69] directly. Finally, it would be interesting to explore other network architectures for feature extraction. Vision transformers are becoming popular and have shown promising for both several relevant tasks such as depth estimation [45], correspondence estimation [30], and absolute pose regression, including in the multiscene case [54]. Exploring these direction would hopefully improve the network accuracy overall but specifically could help with difficult datasets such as Cambridge Landmarks. Overall, we hope that our work will inspire the vision community to continue exploring absolute pose regression as an alternative to traditional pose estimation techniques.

Bibliography

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 8

[2] Vassileios Balntas, Shuda Li, and Victor Prisacariu. Relocnet: Continuous metric learning relocalisation using neural nets. In *European Conference on Computer Vision*, 2018. 23, 32

[3] Hunter Blanton, Scott Workman, and Nathan Jacobs. Extending absolute pose regression to multiple scenes. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2020. 4

[4] Hunter Blanton, Scott Workman, and Nathan Jacobs. A structure-aware method for direct pose estimation. *arXiv preprint arXiv:2012.12360*, 2020. 4

[5] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. Codeslam — learning a compact, optimisable representation for dense visual slam. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 24, 26, 29, 46, 56

[6] Eric Brachmann. ESAC. `https://github.com/vislearn/esac`, 2019. 52

[7] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac - differentiable ransac for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 9, 24, 44

[8] Eric Brachmann and Carsten Rother. Learning less is more - 6d camera localization via 3d surface regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 7, 9, 22, 24, 26, 38, 43, 44

[9] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *International Conference on Computer Vision*, 2019. 9, 11, 24, 30, 32, 44, 45, 47, 48, 56

[10] Eric Brachmann and Carsten Rother. Neural- Guided RANSAC: Learning where to sample model hypotheses. In *International Conference on Computer Vision*, 2019. 24

[11] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3, 7, 8, 10, 13, 23, 31, 32, 38, 43, 48, 55

[12] Mai Bui, Shadi Albarqouni, Slobodan Ilic, and Nassir Navab. Scene coordinate and correspondence learning for image-based localization. In *British Machine Vision Conference*, 2018. 24, 31, 32, 56

[13] Ming Cai, Chunhua Shen, and Ian D. Reid. A hybrid probabilistic model for camera relocalization. In *British Machine Vision Conference*, 2018. 31, 38, 48

[14] Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *IEEE Visual Communications and Image Processing*, 2017. 29, 46

[15] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *arXiv preprint arXiv:2001.05049*, 2020. 44

[16] Zheng Dang, Kwang Moo Yi, Yinlin Hu, Fei Wang, Pascal Fua, and Mathieu Salzmann. Eigendecomposition-free training of deep networks with zero eigenvalue-based losses. In *European Conference on Computer Vision*, 2018. 24

[17] Zheng Dang, Kwang Moo Yi, Yinlin Hu, Fei Wang, Pascal Fua, and Mathieu Salzmann. Eigendecomposition-free training of deep networks with zero eigenvalue-based losses. In *ECCV*, 2018. 49

[18] Tom van Dijk and Guido de Croon. How do neural networks see depth in single images? In *International Conference on Computer Vision*, 2019. 39

[19] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. Camnet: Coarse-to-fine retrieval for camera re-localization. In *International Conference on Computer Vision*, 2019. 2, 8, 23, 32, 42, 43

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2, 9, 24, 43

[22] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 24

[23] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 24

[24] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *International Conference on Computer Vision*, 2019. 1, 26

[25] Hunter Goforth, Yasuhiro Aoki, Arun Srivatsan Rangaprasad, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 25

[26] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 8

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 12, 29

[28] Arnold Irschara, C. Zach, Jan-Michael Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. *CVPR*, 2009. 2

[29] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 41

[30] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: correspondence transformer for matching across images. *CoRR*, abs/2103.14167, 2021. 57

[31] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 25, 27, 42, 44, 46, 56

[32] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *IEEE International Conference on Robotics and Automation*. IEEE, 2016. 7, 8, 10, 23, 32, 56

[33] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 7, 10, 12, 13, 23, 30, 31, 35, 38, 43, 48

[34] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. 2015. 1, 2, 3, 7, 8, 9, 10, 13, 22, 23, 31, 35, 38, 41, 42, 48, 49, 50, 55

[35] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014. 28

[36] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155, 2009. 30, 49, 50

[37] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *arXiv preprint arXiv:2006.14616*, 2020. 28

[38] D.G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 2

[39] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *International Conference on Computer Vision*, 2019. 25

[40] Xiao Xin Lu. A review of solutions for perspective-n-point problem in camera pose estimation. In *Journal of Physics: Conference Series*, volume 1087, page 052009, 2018. 24, 25

[41] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Image-based localization using hourglass networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017. 8, 23, 31, 43, 48

[42] Tayyab Naseer and Wolfram Burgard. Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. 38

[43] Nathan Piasco, Désiré Sidibé, Cédric Demonceaux, and Valérie Gouet-Brunet. Perspective-n-learned-point: Pose estimation from relative depth. In *British Machine Vision Conference*, 2019. 24, 32, 42, 43, 56

[44] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, 1999. 25

[45] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *CoRR*, abs/2103.13413, 2021. 57

[46] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 24

[47] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[48] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[49] Soham Saha, Girish Varma, and C. V. Jawahar. Improved visual relocalization by discovering anchor points. In *British Machine Vision Conference*, 2018. 9, 23, 32

[50] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. Pcrnet: Point cloud registration network using pointnet encoding. *arXiv preprint arXiv:1908.07906*, 2019. 25

[51] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1744–1756, 2016. 2, 22, 32, 41

[52] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3, 7, 8, 10, 22, 23, 42

[53] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2217–2225, New York, New York, USA, 20–22 Jun 2016. PMLR. 12

[54] Yoli Shavit, Ron Ferens, and Yosi Keller. Learning multi-scene absolute pose regression with transformers. *CoRR*, abs/2103.11468, 2021. 57

[55] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 2, 9, 13, 24, 30, 32, 44, 47, 48

[56] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 2

[57] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 24, 44

[58] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2, 22, 32

[59] Fabio Tosi, Filippo Aleotti, Matteo Poggi, and Stefano Mattoccia. Learning monocular depth estimation infusing traditional stereo knowledge. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 24

[60] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *International Conference on 3D Vision*, 2016. 29

[61] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature

correlation. In *International Conference on Computer Vision*, 2017. 8, 23, 31, 38, 43, 48, 49

[62] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *International Conference on Computer Vision*, 2019. 25, 44

[63] Scott Workman, Richard Souvenir, and Nathan Jacobs. Wide-area image geolocalization with aerial reference imagery. In *International Conference on Computer Vision*, 2015. 8

[64] Jian Wu, Liwei Ma, and Xiaolin Hu. Delving deeper into convolutional neural networks for camera relocalization. In *IEEE International Conference on Robotics and Automation*, 2017. 31, 48

[65] Fei Xue, Xin Wang, Zike Yan, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Local supports global: Deep camera relocalization with sequence enhancement. In *International Conference on Computer Vision*, 2019. 23, 31, 43, 48

[66] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. Sanet: Scene agnostic network for camera localization. In *International Conference on Computer Vision*, 2019. 8, 24, 32, 42, 43, 45

[67] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. *arXiv preprint arXiv:2003.01060*, 2020. 44

[68] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision*, 2018. 24

[69] Lei Zhou, Zixin Luo, Tianwei Shen, Jiahui Zhang, Mingmin Zhen, Yao Yao, Tian Fang, and Long Quan. Kfnet: Learning temporal camera relocalization using kalman filtering. In *CVPR*, 2020. 56

Vita

Hunter Blanton

Education

2012–2016	B.S. in Mathematics *S*umma Cum Laude Minor in Physics	University of Kentucky

Research Interests

Computer vision: Camera Pose Estimation, 3D Reconstruction, Novel View Synthesis

Appointments

Graduate Research Assistant, Computer Science Department	University of Kentucky
2017–Present	*L*exington, KY
Undergraduate Research Assistant, Physics Department	University of Kentucky
2015–2016	*L*exington, KY

Honors and Awards

- Outstanding Ph.D. Student in Computer Science, University of Kentucky, 2021
- Dean's List, University of Kentucky, 2013-2016

- Kentucky Governor's Scholar Presidential Scholarship, University of Kentucky, 2012-2016

Publications

[1] Xiaoqin Wang, Gongbo Liang, Yu Zhang, Hunter Blanton, Zachary Bessinger, and Nathan Jacobs. Inconsistent performance of deep learning models on mammogram classification. In *Journal of the American College of Radiology*, 2020.

[2] M. Usman Rafique, Hunter Blanton, Noah Snavely, and Nathan Jacobs. Generative appearance flow: A hybrid approach for outdoor view synthesis. In *British Machine Vision Conference (BMVC)*, 2020.

[3] Yu Zhang, Xiaoqin Wang, Hunter Blanton, Liang Gongbo, Xin Xing, and Nathan Jacobs. Convolutional neural networks for 3d digital breast tomosynthesis classification. In *IEEE International Conference on Bioinformatics and Biomedicine*, 2019.

[4] Gongbo Liang, Xiaoqin Wang, Yu Zhang, Xin Xing, Hunter Blanton, Tawfiq Salem, and Nathan Jacobs. Joint 2d-3d breast cancer classification. In *IEEE International Conference on Bioinformatics and Biomedicine*, 2019.

[5] Hunter Blanton, Connor Greenwell, Scott Workman, and Nathan Jacobs. Extending absolute pose regression to multiple scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[6] Hunter Blanton, Sean Grate, and Nathan Jacobs. Surface modeling for airborne lidar. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2020.

[7] Armin Hadzic, Hunter Blanton, Weilian Song, Mei Chen, Scott Workman, and Nathan Jacobs. Rasternet: Modeling free-flow speed using lidar and overhead imagery. In *CVPR Workshop: Large Scale Computer Vision for Remote Sensing Imagery (EARTHVISION)*, 2020.

[8] Scott Workman, M. Usman Rafique, Hunter Blanton, Connor Greenwell, and Nathan Jacobs. Single image cloud detection via multi-image fusion. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2020.

[9] Xin Xing, Gongbo Liang, Hunter Blanton, M. Usman Rafique, Chris Wang, Ai-Ling Lin, and Nathan Jacobs. Dynamic image for 3d mri image alzheimer's disease classification. In *ECCV Workshop on BioImage Computing*, 2020.

[10] M. Usman Rafique, Hunter Blanton, and Nathan Jacobs. Weakly supervised fusion of multiple overhead images. In *CVPR Workshop: Large Scale Computer Vision for Remote Sensing Imagery (EARTHVISION)*, 2019.

[11] Tawfiq Salem, Connor Greenwell, Hunter Blanton, and Nathan Jacobs. Learning to map nearly anything. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2019.

[12] Weilian Song, Tawfiq Salem, Hunter Blanton, and Nathan Jacobs. Remote estimation of free-flow speeds. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2019.

[13] Hunter Blanton, Scott Workman, and Nathan Jacobs. A structure-aware method for direct pose estimation. In *arXiv preprint arXiv:2012.12360*, 2020.

Professional Service

- Reviewing for Conferences and Journals:
  - British Machine Vision Conference (BMVC) (2020)
  - IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019-)
  - IEEE International Conference on Computer Vision (ICCV) (2019-)
  - IEEE Winter Conference on Applications of Computer Vision (WACV) (2020)
  - IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2019)
  - IEEE Transactions on Image Processing (TIP) (2020)

Teaching

Teaching Assistant

- *Discrete Mathematics*, CS 275, (F2016, S2017, S2017), University of Kentucky