Towards Deep Learning for Long-Term Visual Localization in Visual Teach and Repeat

by

Mona Gridseth

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

Towards Deep Learning for Long-Term Visual Localization in
Visual Teach and Repeat

Mona Gridseth
Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2022

Visual path following can facilitate autonomous operation of robots in a large variety of environments while solely relying on an inexpensive camera sensor. Visual Teach and Repeat (VT&R) achieves accurate metric and long-range autonomous path following in unstructured environments. The user teaches a path by driving the robot manually, after which the path can be repeated autonomously. During long-term operation, the camera is affected by lighting changes, and the appearance of the environment changes. This poses a challenge for localizing to the visual map that was built when teaching the path. VT&R can operate across large appearance change by using multi-experience localization. Each time the robot repeats a path, the run is stored as a new experience in the map. When localizing, VT&R can rely on intermediate experiences to bridge the appearance gap. This requires collecting experiences continuously as the environment changes and from scratch whenever a new path is taught.

In this thesis, we tackle long-term localization using deep learning techniques with the aim of removing the need for intermediate bridging experiences. We first show that we can train a neural network to regress relative poses for localization directly from pairs of images. With this method, we localize across large lighting and seasonal change but fail to generalize to new and unfamiliar paths. We improve on this work by learning visual features that can be used in the VT&R pipeline. With these deep learned features, we perform closed-loop path following across a full range of lighting change and also show that the learned features generalize to new areas. We further test the learned features' robustness to seasonal change in an experiment from late summer through autumn. The experiments cover a total of 87.7 km of driving.

Til farmor og farfar
Kaspara og Peder Gridseth

Acknowledgements

This thesis would not be possible without the help and support of many people. First of all, I would like to extend my gratitude to my advisor, Professor Tim Barfoot, for accepting me into the Autonomous Space Robotics Laboratory (ASRL) and for all his support, encouragement, ideas, and insights that have been essential to the completion of this research. The many constructive questions and careful editing of papers and this thesis, have made the work significantly better. Over the last few years, I have learned a lot about robotics, research, and communication of research.

Many thanks to Professor Angela Schöllig and Professor Jonathan Kelly, who served on the Doctoral Examination Comittee, for their assistance over the years by providing feedback on my research and, finally, on this thesis. Your comments have provided direction and inspiration. Thank you also to the external reviewers of the thesis, Professor Ingmar Posner and Professor Sanja Fidler, for reading the thesis and providing insightful comments that further improved this work.

I would like to acknowledge Clearpath robotics for providing funding that made this research possible.

I owe a debt of gratitude to all my colleagues in ASRL whose collaboration and contributions have been key to the work that resulted in this thesis. I have valued the collaboration, discussions of ideas, help with fixing issues, and many enjoyable conversations. I would like to thank Michael Paton, Kirk MacTavish, Patrick McGarey, Peter Berczi, Michael Warren, Katarina Cujic, David Yoon, Tim Tang, Hengwei Guo, Nan Zhang, Keenan Burnett, Jeremy Wong, Rehman Merali, Ben Congram, Mollie Bianchi, Haowei Zhang, Gorden Tseng, Yuchen Wu, Daniel Guo, Hugues Thomas, Sherry Chen, Jordy Sehn, Connor Holmes, and Frederike Dümbgen. It has been a great pleasure working with all of you.

I would also like to thank Lee Clement and Valentin Peretroukhin from the Space and Terrestrial Autonomous Robotic Systems (STARS) lab for including me in collaboration on one of their research projects and for their time spent teaching me about various aspects of their research and robotics in general.

I had a lot of fun during my PhD experience at the University of Toronto Institute for Aerospace Studies (UTIAS) thanks to the many fellow graduate students, whether we enjoyed lunch together, took time to chat in the hallway, or played softball during the summer. Thank you to everyone for creating a great community.

Without the unwavering support from family and friends, I would not have been able to complete this thesis. Thank you to Tanvi for true friendship and many long walks together with Scotty that were always a highlight of my week. Thank you to Aristotle for your encouragement and great times spent both in Edmonton and later Toronto. I am grateful to the Strangway family, Salina, Richard, and Spencer, for your constant care and support throughout my time in Canada. Thank you to my sister, Lene, and parents, Torhild and Jostein, for being an inspiration and for always encouraging my work despite it taking me far away from home. Finally, thank you to my partner, Tyrone, whose knowledge, advice, understanding, and support has kept me on course.

Contents

List of Tables

List of Figures

Acronyms

ATE	Absolute Trajectory Error
BRIEF	Binary Robust Independent Elementary Features
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DOF	degrees of freedom
FAST	Features from Accelerated Segment Test
LiDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory
MAV	Micro Aerial Vehicle
MEL	Multi-Experience Localization
MLP	Multi-Layer Perceptron
NMS	Non-Maximal Suppression
ORB	Oriented FAST and Rotated BRIEF
PnP	Perspective-n-Point
RADAR	Radio Detection and Ranging
RANSAC	Random Sample Consensus
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RPE	Relative Pose Error
SEL	Single-Experience Localization
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SPP	Spatial Pyramid Pooling
SfM	Structure-from-Motion
SURF	Speeded-Up Robust Features
SVD	Singular Value Decomposition
tanh	Hyperbolic Tangent
UAV	Unmanned Aerial Vehicle
UTIAS	University of Toronto Institute for Aerospace Studies
VO	Visual Odometry
VT&R	Visual Teach and Repeat

Notation

a	Symbols in this font are real scalars.
$\mathbf{a}$	Symbols in this font are real column vectors.
$\mathbf{A}$	Symbols in this font are real matrices.
$\mathbf{1}$	The identity matrix.
$\mathbf{0}$	The zero matrix.
$\underrightarrow{\mathcal{F}}_a$	A vectrix representing a reference frame in three dimensions.
$\hat{(\cdot)}$	An estimated quantity.

Chapter 1

Introduction

1.1 Motivation

Vision-based navigation can be applied to a wide variety of robots such as Unmanned Aerial Vehicles (UAV), rovers for space exploration, or autonomous cars. In VT&R (Furgale and Barfoot, 2010) a robot autonomously repeats a known path while only relying on a camera for sensing. A user manually drives the robot to teach the path and build a visual map, ensuring that the route can be safely traversed by the robot. See Figure 1.1 for an illustration of VT&R operation. Route repetition is potentially useful in many real-world applications such as robots monitoring crops in an orchard, operation of trucks in underground mines without access to GPS, or robots in large warehouses.

Cameras are commercially ubiquitous and inexpensive, making them a good option as a sensor for robotic navigation. Moreover, they have low power consumption and few moving parts. Cameras produce texture-rich images of a scene, providing useful information for vision algorithms. Compared to 3D Light Detection and Ranging (LiDAR) and Radio Detection and Ranging (RADAR) sensors, cameras are highly sensitive to lighting. In addition to lighting change, seasonal or structural changes to the environment also pose challenges for visual path following in the long term (see Figure 1.2). For path

(a) Manual teach at 12:56 to build a visual map.

(b) Autonomous repeat after dark at 17:22.

Figure 1.1: In VT&R, the user teaches a path by driving the robot manually (a), after which the path is repeated autonomously (b). For each figure, the UI (left) shows the estimated path and the robot's position. On the right, we see an external view of the Clearpath Grizzly robot (top) together with the live camera frame and matched features (bottom) for Visual Odometry (VO) during the teach phase and localization during the repeat phase.

(a) Lighting change.

(b) Seasonal change.

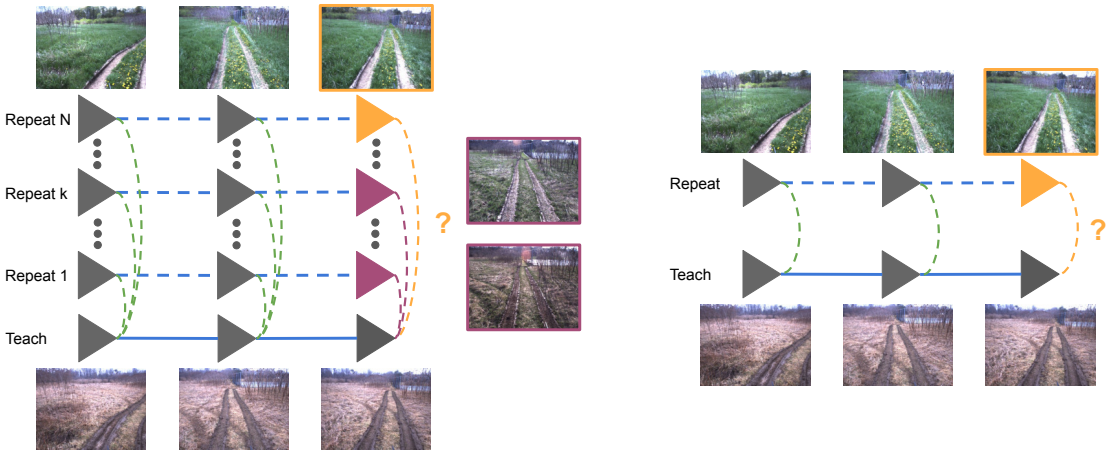
Figure 1.2: In order to be feasible for real-world applications, long-term visual path following must work even when the environment changes.

following to be useful in real-world applications, reliable long-term operation in changing environments is required.

Furgale and Barfoot (2010) perform long-range autonomous path following with VT&R using only a stereo camera. They extract sparse visual features, called Speeded-Up Robust Features (SURF) (Bay et al., 2006), from images in the visual map as well as the live images seen by the robot during a repeat. By matching features between the map and live images, they were able to localize and compute the current offset of the robot with respect to the mapped path. With this information, the path-tracking controller computes the velocities needed to stay on the path. The downside to such handcrafted visual features is their lack of robustness to lighting change or longer-term changes to the environment.

Further work was done to improve the operational time-range of VT&R. Paton et al. (2015) first converted RGB images to colour-constant images (Corke et al., 2013), which extended the operational time from a few hours to being able to repeat a path between sunrise and sunset over multiple days. Later, Paton et al. (2016) introduced Multi-Experience Localization (MEL) (Churchill and Newman, 2013) to VT&R. Each time the robot repeats a path, the run is stored as a new experience in the map. When localizing back to the map, they can use visual features from experiences with the most similar appearance to the current condition instead of matching to features from the initial teach run. In other words, localization relies on intermediate experiences stored in the map to bridge the appearance gap, see Figure 1.3a. This extension to VT&R achieved accurate path following across a full day of lighting change, including driving after dark (MacTavish et al., 2017), and across seasons in an experiment that was run from January until May (Paton, 2018).

The reliance on intermediate bridging experiences in multi-experience VT&R comes with a couple of downsides. Firstly, we must repeat the path often enough to gradually capture the environmental change. In a scene with rapid change, we need to collect a large number of experiences to successfully describe the environment. Secondly, we must collect experiences from scratch for each new path - even if we operate in the same area - since the experiences are specific to each map. We can imagine applications where MEL may fail. If we are teaching a longer path during varying conditions (such as during sunrise,



(a) Multi-experience localization.

(b) Single-experience localization.

Figure 1.3: MEL (a) relies on intermediate bridging experiences (purple) to localize a live frame (orange) where appearance has changed significantly since the mapped teach run. The motivation for our work is to make Single-Experience Localization (SEL) robust for long-term operation, avoiding the use of intermediate experiences.

sunset, or changing weather), we will not be able to repeat the full path relying on traditional hand-crafted features because the environment has changed during operation. An example of this scenario is a UAV that relies on vision-based repetition of an outward route for emergency return if GPS navigation fails. Another case where we want to avoid multiple visits to the same path, is operation in hazardous environments. In our research, we aim to tackle long-term localization in VT&R. In particular, we wish to improve the robustness of Single-Experience Localization (SEL) to large appearance change, such that we can localize directly to the original teach run without relying on intermediate bridging experiences, see Figure 1.3b. We can think of robust SEL and MEL with hand-crafted features where every repeat is added as an experience to the map, as two extremes on a spectrum. Linegar (2016) combined MEL with a small number of representative experiences, different types of hand-crafted features as well as learned place-dependent features to handle long-term localization in an off-road environment. However, they combined one experience from winter and one from summer for their map. This relies on having two experiences from different seasons available when building the map, which would not be the case when starting to teach a new path from scratch.

Our goal is to focus on autonomous visual path following in outdoor scenes, including challenging off-road environments. A substantial part of our work will include localization in areas with undulating terrain together with tall grass, bushes, and trees as seen in Figure 1.2. They also have fewer permanent structures such as buildings or roads. These characteristics can pose challenges for localization - especially during large lighting or seasonal change - that do not arise in more structured urban scenes common in public localization datasets (examples include the Aachen (Sattler et al., 2012) and Oxford Robotcar datasets (Maddern et al., 2017)). For instance, as grass and leaves wither in the autumn, the appearance changes more drastically in a forest area or open field than on a city street surrounded by buildings.

1.2 Research Objectives

There are two main objectives of this thesis as alluded to in the previous section. The first objective is to improve the robustness of visual localization for VT&R such that we can localize across large appearance change without the need for intermediate bridging experiences. The second objective is to include our localization approach in the VT&R system and demonstrate real-time autonomous path following in challenging outdoor environments, including off-road areas with significant vegetation and few permanent structures. Moreover, the experiments should achieve accurate path following across lighting and seasonal change. We emphasize that the accuracy of pose estimation in VT&R is very high (Clement et al., 2017) and so we are not working to improve path-following accuracy, but focus on long-term operation.

1.3 Novel Contributions

This thesis presents the following novel contributions towards tackling the research objective of robust long-term visual localization in VT&R.

- We replace sparse feature-based visual localization in VT&R with a direct method and are the first to demonstrate direct localization in a challenging off-road example with tall grass and dense vegetation, but limited appearance change (Gridseth and Barfoot, 2019).

- We regress relative pose for single-experience visual localization across large appearance change directly from a pair of images using a deep neural network. We also regress poses for VO with the same network architecture and combine localization and VO for visual path following. We are the first to tackle large seasonal appearance change in a challenging off-road environment with deep learned pose regression (Gridseth and Barfoot, 2020).

- We estimate the pose for single-experience visual localization using sparse features extracted with a deep neural network. We include the learned features for localization into the VT&R pipeline and are the first to achieve real-time autonomous path following across a full range of lighting change with a 100% autonomy rate with deep learned features. We extend the experiment to include three months of seasonal change from summer through autumn, where 79% of runs have no path-following failures. We demonstrate successful generalization to new areas not seen in the training data (Gridseth and Barfoot, 2022).

1.4 Thesis Outline

This thesis is structured as follows; Chapter 2 introduces topics in state estimation and deep learning that we rely on in following chapters of the thesis. The chapter also gives a brief overview of VT&R and describes the datasets that we use to develop and test our research. Chapter 3 describes our first attempt at improving robustness of SEL in VT&R by replacing sparse feature-based localization with a direct method. Seeing the challenges posed by lighting and seasonal change to visual localization, we choose to focus on deep learning for the remainder of our work.

In Chapter 4, we train a deep neural network to regress the relative pose between two input images. With this we are able to localize images across large appearance change. However, generalizing to new

paths not observed in the training data proved challenging. In Chapters 5 and 6, we add more structure to the learning method with the goal of improving generalization. We learn sparse visual features that can be used in a differentiable, classical pose estimation pipeline that allows for end-to-end training. In Chapter 5, we apply the learned features to VO and show that they perform well on previously unseen paths. In Chapter 6, we extend this work to use the learned features for localization. We include the features in the VT&R system and perform path following across a full range of lighting change and three months of seasonal change from summer through autumn. Finally, Chapter 7 summarizes the conclusions and novel contributions of the thesis and discusses possibilities for future work.

In all remaining chapters, except the final one, we follow the same general outline of providing an introduction to motivate the work, a section on related work, and a section to explain the methodology before we present the experiments, results, and conclusions.

1.5 Associated Material

Publications

- Gridseth and Barfoot (2019). Towards Direct Localization for Visual Teach and Repeat. In the *2019 Conference on Computer and Robot Vision (CRV)*.
- Gridseth and Barfoot (2020). DeepMEL: Compiling visual multi-experience localization into a deep neural network. In the *2020 IEEE International Conference on Robotics and Automation (ICRA)*.
- Gridseth and Barfoot (2022). Keeping an Eye on Things: Deep Learned Features for Long-Term Visual Localization. In the *2022 IEEE Robotics and Automation Letters (RAL)*.

Code

The code for training and testing deep learned features for localization in PyTorch is available at https://github.com/utiasASRL/deep_learned_visual_features

Videos

- Deep learned pose estimation: https://youtu.be/lXbTaAM6kSY
- Deep learned pose estimation (conference presentation): https://youtu.be/XrlLDTFiMt8
- Deep learned features: https://youtu.be/mqDivnIQjl0
- Deep learned features (conference presentation): https://youtu.be/JsEBs9Y5XVg
- Deep learned features and VT&R3 (demo): https://youtu.be/ZYkxevdw7DA

Chapter 2

Background

This chapter provides background information relevant to later chapters of the thesis. We start by summarizing some relevant deep learning concepts before discussing topics from three-dimensional geometry and the sensor model that we make use of in our work. We give an overview of VT&R and the challenges of long-term localization. Finally, we give details on the datasets we use to develop our research and how we sample from those dataset. The material on three-dimensional geometry, the sensor model, and VT&R was presented in a similar fashion in the thesis of Paton (2018).

2.1 Deep Learning Primer

In this section we describe some basic concepts form deep learning that are relevant to this thesis. We have based our summary on some of the material presented in the book by Nielsen (2015) and in the introductory review by Emmert-Streib et al. (2020).

Neural Networks

A neural network is a computational model consisting of a set of basic entities, called neurons. The neuron takes an input $\mathbf{x} \in \mathbb{R}^n$, applies weights, $\mathbf{w} \in \mathbb{R}^n$, and adds a bias, $b \in \mathbb{R}$. The result, $z \in \mathbb{R}$, is passed through an activation function (or transfer function), $\phi : \mathbb{R} \to \mathbb{R}$, to get the output, $y \in \mathbb{R}$:

$$y = \phi(z) = \phi\left(\mathbf{w}^T\mathbf{x} + b\right). \tag{2.1}$$

The most used activation functions are non-linear and examples include the Sigmoid function, the Rectified Linear Unit (ReLU) activation function, and the Hyperbolic Tangent (tanh) activation function. In our work we make use of ReLU (Nair and Hinton, 2010), which is defined as:

$$\phi(z) = \begin{cases} 0 & \text{if } x < 0 \\ z & \text{if } x \geq 0. \end{cases}$$

To form a neural network, the neurons are connected to each other, i.e. the output of one neuron is passed as the input to another. In the simplest neural network, a Multi-Layer Perceptron (MLP), the neurons are organized in layers with one input layer, one or more hidden layers, and one output layer, see Figure 2.1. The depth of the network is determined by the number of activation functions connecting

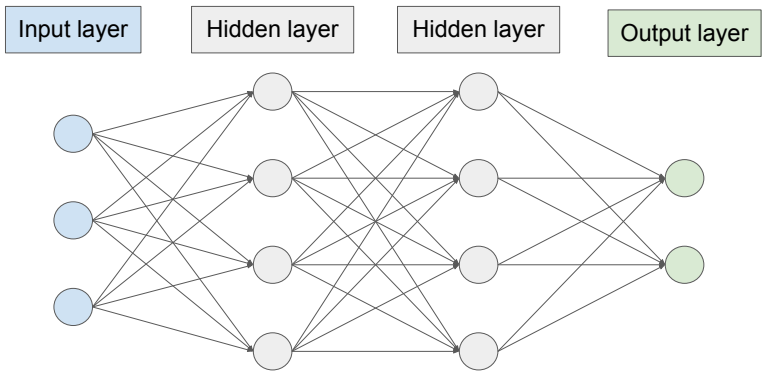


Figure 2.1: A basic MLP neural network with one input layer, two hidden layers, and one output layer. The layers are fully connected.

the layers, while its weight is given by the number of neurons contained in a hidden layer. Networks, where information is passed forward form one layer to the next without loops, are called feedforward neural networks.

When training a neural network, the goal is to find the optimal network parameters. We start by computing the error between the network output and the ground truth labels, which are then used to form a cost function (or loss function). During training we find the optimal network parameters by minimizing the cost function for the training data. This can be done with gradient descent, where backpropagation is used to calculate the gradients.

Convolutional Neural Networks

In networks with fully connected layers, each neuron in a layer is connected to all neurons in the next layer, as seen in Figure 2.1. This can result in a very high number of network parameters. A Convolutional Neural Network (CNN) is a type of neural network that is commonly used when working with images. CNNs are able to handle such high-dimensional data by using alternatives to fully connected layers that use fewer parameters. Moreover, a fully connected MLP does not take into consideration the spatial structure of images, but instead treats pairs of pixels the same way whether they are far apart or close together. CNNs rely on three main tools, namely local receptive fields, shared weights, and pooling, which we describe in the following.

For a CNN, we can imagine a layer as a grid of neurons and the input layer corresponds to the image grid of pixels. We pick a small square region (or window) in the input image and connect it to one neuron in the first hidden layer. This is called a local receptive field. Then we slide this region across the image, moving a chosen number of pixels (stride) over to the right each time and down when we reach the right edge of the image. As we slide the region over the image, we connect each new receptive field to a neuron in the first hidden layer. The same operation is repeated to connect the following layers.

Each neuron in a hidden layer has one bias, and the number of weights equal the size of the local receptive field connected to the neuron. The same bias and weights are used for each of the neurons in a hidden layer. We refer to these as shared weights and a shared bias, which are also commonly called kernels or filters. At a high level, it means that all the neurons in one hidden layer detect the same type of feature, such as an edge, in different locations in the image. Combining several individual layers for feature detection forms one convolutional layer. By not connecting every pixel to every neuron in

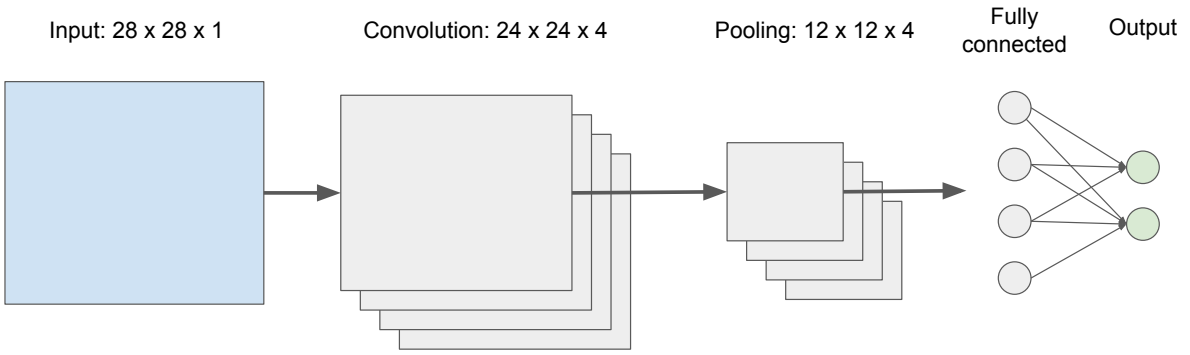


Figure 2.2: A CNN with a convolutional layer with 24x24 neurons across four individual features layers that share weights and a bias within the layer. Pooling reduces the spacial resolution to 12x12 neurons. A fully connected layer is included at the end before the output layer.

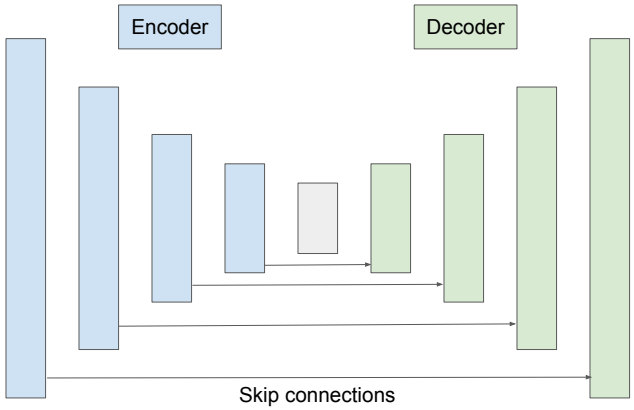


Figure 2.3: Encoder-decoder architecture with strided convolutions in the encoder for downsampling and strided convolutions for upsampling in the encoder. Skip connections between layers preserve information that may otherwise be lost during downsampling.

the next layer, and by using shared weights, the convolutional layers reduce the number of parameters needed.

After a convolutional layer, a pooling layer can be used to summarize and simplify the output. In particular, a region in the output from the convolutional layer can be summarized into one neuron by the pooling operation. Options include, but are not limited to, keeping the maximum value from the region (max-pooling) or taking an average of the values (average pooling). The pooling is done separately for each individual layer making up the convolution layer, and so the number of layers stay the same while the spatial resolution is reduced. This, again, helps reduce the number of parameters needed.

The final CNN is usually a combination of convolutional layers and pooling layers with one or more fully connected layers at the end before the output layer, see Figure 2.2. The components of CNNs can also be used to form a network architecture that contains an encoder and a decoder, see Figure 2.3. UNet, by Ronneberger et al. (2015), is an example of an encoder-decoder. The encoder uses strided convolutions to learn features while reducing the spatial resolution of the input, referred to as downsampling or compression. Afterwards, strided convolutions are used to increase the resolution back to the original input image size, referred to as upsampling or decompression. The layer at the

smallest spatial resolution, where the encoder and decoder connect, is referred to as the bottleneck. Skip connections between layers in the encoder and decoder let the network preserve information that may otherwise be lost during the downsampling.

2.2 State Estimation Primer

This section presents some topics related to state estimation that we use in different chapters of the thesis.

2.2.1 Three-Dimensional Geometry

Vehicles that translate and rotate have three degrees of freedom (DOF) for rotation and three DOF for translation. We refer to their six DOF configuration as a pose, which consists of position and orientation. The following material explains how to represent rotations and poses in three dimensions. For more detailed derivations and explanations, we refer to Chapter 7 of Barfoot (2017).

Matrix Lie Groups

Rotations and poses do not form vectorspaces but can be represented using the special orthogonal group, $SO(3)$, and the special Euclidean group, $SE(3)$, respectively. $SO(3)$ and $SE(3)$ are matrix Lie groups. A group is a set of elements together with an operation that combines any two elements to form a third element in the same set. This must be the case while satisfying four group axioms: closure, associativity, identity, and invertibility. A Lie group is a group that is also a differential manifold. The elements in a matrix Lie group are matrices.

Rotations are represented with the special orthogonal group, which is the set of valid rotation matrices:

$$SO(3) = \left\{ \mathbf{C} \in \mathbb{R}^{3\times 3} | \mathbf{C}\mathbf{C}^T = \mathbf{1}, \det \mathbf{C} = 1 \right\}. \tag{2.2}$$

Poses are represented using the special Euclidean group, which is the set of valid transformation matrices:

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \mathbf{C} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\}. \tag{2.3}$$

Lie Algebra

Every matrix Lie group has an associated Lie algebra. The Lie algebra consists of a vectorspace, $\mathbb{V}$, over a field, $\mathbb{F}$, together with a Lie bracket, $[\cdot, \cdot]$, which is a binary operation that satisfies the properties of closure, bilinearity, alternating, and Jacobi identity.

The Lie algebra for rotations, or $SO(3)$, is given by:

$$\begin{aligned} \text{vectorspace:} \quad & \mathfrak{so}(3) = \left\{ \boldsymbol{\Phi} = \boldsymbol{\phi}^\wedge \in \mathbb{R}^{3\times 3} | \boldsymbol{\phi} \in \mathbb{R}^3 \right\}, \\ \text{field:} \quad & \mathbb{R}, \\ \text{Lie bracket:} \quad & [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2] = \boldsymbol{\Phi}_1\boldsymbol{\Phi}_2 - \boldsymbol{\Phi}_2\boldsymbol{\Phi}_1, \end{aligned} \tag{2.4}$$

where

$$\boldsymbol{\phi}^\wedge = \begin{bmatrix}\phi_1\\ \phi_2\\ \phi_3\end{bmatrix}^\wedge = \begin{bmatrix}0 & -\phi_3 & \phi_2\\ \phi_3 & 0 & -\phi_1\\ -\phi_2 & \phi_1 & 0\end{bmatrix} \in \mathbb{R}^{3\times 3}, \quad \boldsymbol{\phi} \in \mathbb{R}^3. \tag{2.5}$$

The Lie algebra for poses, or $SE(3)$, is given by:

$$\begin{aligned} \text{vectorspace:} \quad & \mathfrak{se}(3) = \left\{\boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge \in \mathbb{R}^{4\times 4} | \boldsymbol{\xi} \in \mathbb{R}^6\right\}, \\ \text{field:} \quad & \mathbb{R}, \\ \text{Lie bracket:} \quad & [\boldsymbol{\Xi}_1, \boldsymbol{\Xi}_2] = \boldsymbol{\Xi}_1\boldsymbol{\Xi}_2 - \boldsymbol{\Xi}_2\boldsymbol{\Xi}_1, \end{aligned} \tag{2.6}$$

where

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\phi}\end{bmatrix}^\wedge = \begin{bmatrix}\boldsymbol{\phi}^\wedge & \boldsymbol{\rho}^\wedge\\ \mathbf{0} & \boldsymbol{\phi}^\wedge\end{bmatrix} \in \mathbb{R}^{6\times 6}, \quad \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3. \tag{2.7}$$

Exponential Map

The exponential map is used to relate a matrix Lie group to the corresponding Lie algebra. For a square matrix, $\mathbf{A} \in \mathbb{R}^{M\times M}$, the exponential map and logarithm are:

$$\exp(\mathbf{A}) = \mathbf{1} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \cdots = \sum_{n=0}^{\infty}\frac{1}{n!}\mathbf{A}^n, \tag{2.8}$$

$$\ln(\mathbf{A}) = \sum_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}(\mathbf{A}-\mathbf{1})^n. \tag{2.9}$$

For rotations, we can relate elements of $SO(3)$ to elements of $\mathfrak{so}(3)$ using the exponential map:

$$\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge) = \sum_{n=0}^{\infty}\frac{1}{n!}\left(\boldsymbol{\phi}^\wedge\right)^n = \cos\phi\mathbf{1} + (1-\cos\phi)\mathbf{a}\mathbf{a}^T + \sin\phi\mathbf{a}^\wedge, \tag{2.10}$$

where $\mathbf{C} \in SO(3)$ and $\boldsymbol{\phi} \in \mathfrak{so}(3)$. This uses an axis-angle formulation, where $\mathbf{a}$ is the axis and ϕ is the angle. We can also convert from $\mathfrak{so}(3)$ to $SO(3)$ with the logarithm map:

$$\phi = \ln(\mathbf{C})^\vee, \tag{2.11}$$

where the operator $(\cdot)^\vee$ is the inverse of (2.5). This conversion is not unique.

For poses, we can relate elements of $SE(3)$ to elements of $\mathfrak{se}(3)$ using the exponential map:

$$\begin{aligned} \mathbf{T} = \exp(\boldsymbol{\xi}^\wedge) &= \sum_{n=0}^{\infty}\frac{1}{n!}\left(\boldsymbol{\xi}^\wedge\right)^n = \sum_{n=0}^{\infty}\frac{1}{n!}\left(\begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\phi}\end{bmatrix}\right)^n = \sum_{n=0}^{\infty}\frac{1}{n!}\begin{bmatrix}\boldsymbol{\phi}^\wedge & \boldsymbol{\rho}\\ \mathbf{0}^T & 1\end{bmatrix}^n \\ &= \begin{bmatrix}\sum_{n=0}^{\infty}\frac{1}{n!}\left(\boldsymbol{\phi}^\wedge\right)^n & \left(\sum_{n=0}^{\infty}\frac{1}{(n+1)!}\left(\boldsymbol{\phi}^\wedge\right)^n\right)\boldsymbol{\rho}\\ \mathbf{0}^T & 1\end{bmatrix} = \begin{bmatrix}\mathbf{C} & \mathbf{r}\\ \mathbf{0}^T & 1\end{bmatrix}, \end{aligned} \tag{2.12}$$

where $\mathbf{T} \in SE(3)$, $\boldsymbol{\xi} \in \mathbb{R}^6$, $\mathbf{r} = \mathbf{J}\boldsymbol{\rho}$, and $\mathbf{J} = \sum_{n=0}^{\infty}\frac{1}{(n+1)!}\left(\boldsymbol{\phi}^\wedge\right)^n$ is the left Jacobian of $SO(3)$. We can

also convert from $\mathfrak{se}(3)$ to $SE(3)$ with the logarithm map, which, again, is not unique:

$$\xi = \ln(\mathbf{T})^{\vee} = \ln\left(\begin{bmatrix}\mathbf{C} & \mathbf{r}\\ \mathbf{0}^T & 1\end{bmatrix}\right)^{\vee} = \begin{bmatrix}\mathbf{J}^{-1}\mathbf{r}\\ \boldsymbol{\phi}\end{bmatrix}, \tag{2.13}$$

and $\mathbf{J}^{-1}$ is the inverse Jacobian of $SO(3)$.

2.2.2 Stereo Camera Model

All the work in this thesis is done using a stereo camera as the only sensor for the robot. Cameras are widely available, inexpensive sensors that can be used to estimate a robot's motion and the geometry of the surrounding environment. In this section, we present the stereo camera sensor model as it is defined in Chapter 6 of Barfoot (2017). We rely on the stereo camera model for tasks such as triangulating 3D points from 2D image coordinates (Chapters 5 and 6) and transforming image coordinates from one frame to another with a warp function (Chapter 3).

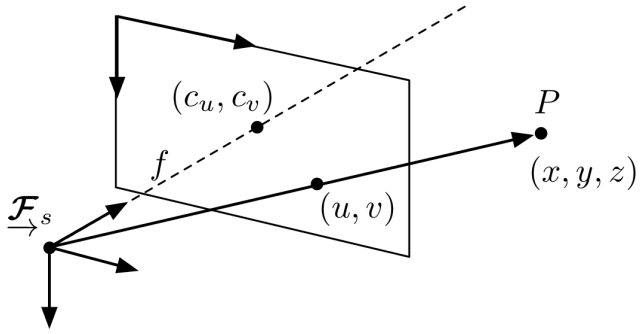


Figure 2.4: This figure is from Barfoot (2017). Illustration of a perspective camera. In reality the image plane is behind the camera, but the frontal projection model avoids working with a flipped image. The intrinsic parameters are focal length, f, and optical centre, (c_u, c_v). With the camera model we can project the point, P, into its image coordinates, (u, v).

Perspective Camera

We start with a perspective camera with a frontal projection model, illustrated in Figure 2.4. Lens effects can distort camera images. However, we assume that undistortion (calibration) has been applied so that the images appear as coming from an idealized pinhole camera. Given a point, P, in the sensor frame, $\underrightarrow{\mathcal{F}}_s$, with coordinates, $\mathbf{p} = \begin{bmatrix}x & y & z\end{bmatrix}^T$, we can project it into its image frame coordinates, $\mathbf{q} = \begin{bmatrix}u & v\end{bmatrix}^T$, with the perspective camera model, $\mathbf{s} : \mathbb{R}^3 \to \mathbb{R}^2$, as follows:

$$\mathbf{q} = \begin{bmatrix}u\\ v\end{bmatrix} = \mathbf{s}(\mathbf{p}) = \underbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix}f_u & 0 & c_u\\ 0 & f_v & c_v\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{K}} \frac{1}{z}\begin{bmatrix}x\\ y\\ z\end{bmatrix}, \tag{2.14}$$

where $\mathbf{P}$ is a projection matrix, and $\mathbf{K}$ holds the intrinsic camera parameters. These are the focal lengths, f_u and f_v, and the coordinates of the optical centre, (c_u, c_v). With one perspective camera

alone, we cannot recover the depth of points in the environment. Combining two cameras into a stereo camera solves this problem.

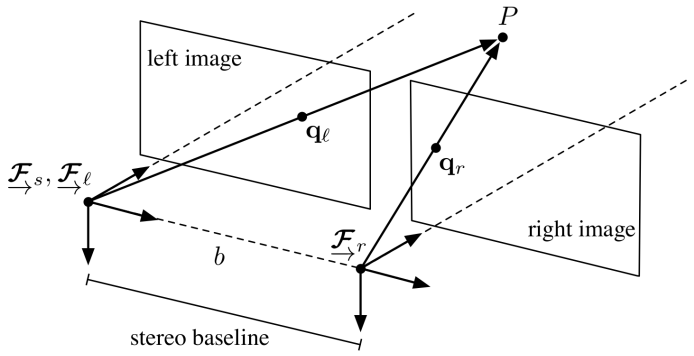


Figure 2.5: This figure is from Barfoot (2017). A stereo camera configuration, where the cameras are separated by the baseline, b, distance along the x-axis. The stereo camera sensor frame, $\underrightarrow{\mathcal{F}}_s$, is chosen to coincide with the left camera sensor frame, $\underrightarrow{\mathcal{F}}_\ell$.

Stereo Camera

A stereo camera consists of two perspective cameras connected by a known, fixed transform. Most commonly they are placed side-by-side along the x-axis pointing in the same direction, and the distance between the cameras is referred to as the baseline, b. For our purposes, we use a left stereo camera model, which means that the sensor frame, $\underrightarrow{\mathcal{F}}_s$, coincides with the sensor frame of the left camera, $\underrightarrow{\mathcal{F}}_\ell$. Figure 2.5 illustrates such a stereo camera configuration.

We can define a stereo camera model, $\mathbf{g} : \mathbb{R}^3 \to \mathbb{R}^4$, that maps a point, P, in the sensor frame, $\underrightarrow{\mathcal{F}}_s$, with coordinates, $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, to stereo image coordinates, $\mathbf{q} = \begin{bmatrix} u_l & v_l & u_r & v_r \end{bmatrix}^T$:

$$\mathbf{q} = \begin{bmatrix} u_\ell \\ v_\ell \\ u_r \\ v_r \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \underbrace{\begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u b \\ 0 & f_v & c_v & 0 \end{bmatrix}}_{\mathbf{M}} \frac{1}{z} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\mathbf{p}} = \mathbf{M} \frac{1}{\mathbf{c}^T \mathbf{p}} \mathbf{p}, \tag{2.15}$$

where $\mathbf{c}^T = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$, and the stereo camera frame coincides with the left camera frame We also assume that both cameras have the same intrinsic properties.

If we wish to recover the coordinates of the point, P, we need the inverse of the stereo camera model that maps stereo image coordinates to a homogeneous point:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{g}^{-1}(\mathbf{q}) = \begin{bmatrix} \frac{b}{u_\ell - u_r}(u_\ell - c_u) \\ \frac{b f_u}{f_v (u_\ell - u_r)}(v_\ell - c_v) \\ \frac{b f_u}{u_\ell - u_r} \\ 1 \end{bmatrix}. \tag{2.16}$$

The disparity, $u_\ell - u_r$, is found by performing stereo matching. In our work, we use the stereo matching

method from Hirschmuller (2008) as implemented in OpenCV (OpenCV, 2019).

Warp Function

When we have a stereo camera together with its inverse, we have the ability to transform 2D image coordinates from one image frame to another. Given a relative pose transformation, $\mathbf{T} \in SE(3)$, between to camera frames, we want to transform the stereo image coordinates, $\mathbf{q}$, from one frame to stereo image coordinates, $\mathbf{q}'$, in the other frame using the warp function, $\mathbf{w} : SE(3) \times \mathbb{R}^4 \to \mathbb{R}^4$:

$$\mathbf{q}' = \mathbf{w}(\mathbf{T}, \mathbf{q}) = \mathbf{g}\left(\mathbf{T}\,\mathbf{g}^{-1}(\mathbf{q})\right) = \mathbf{g}\left(\mathbf{T}\,\mathbf{p}\right) = \mathbf{M}\frac{1}{\mathbf{c}^T\mathbf{T}\mathbf{p}}\mathbf{T}\mathbf{p}. \tag{2.17}$$

2.2.3 Point-Cloud Alignment

In Chapters 5 and 6, we need to solve a point-cloud alignment, meaning that we want to find the best estimate of the pose to align two sets of 3D point measurements. How the rotation or pose is represented (with quaternions, rotation matrices, or transformation matrices) determines the approach used to solve for the pose. If we represent the rotation with rotation matrices, we can use a non-iterative method that relies on Singular Value Decomposition (SVD) (Umeyama, 1991). Although an iterative approach with transformation matrices only requires solving a system of linear equations, our later work requires a non-iterative approach. The following is based on the more thorough explanation in Chapter 8 of Barfoot (2017).

Assume that we have two frames, one attached to a map frame, $\underrightarrow{\mathcal{F}}_m$, and one to the live robot camera, $\underrightarrow{\mathcal{F}}_\ell$. For both of these frames, we have M measurements of a set of landmark points, P_i, given in the respective frames, namely $\mathbf{r}_m^{p_i m}$ and $\mathbf{r}_\ell^{p_i \ell}$, where $i = 1 \dots M$. The goal is to find the rotation matrix, $\mathbf{C}_{\ell m}$, and translation, $\mathbf{r}_m^{\ell m}$, that will align the two sets of points. In order to simplify notation, we define:

$$\mathbf{y}_i = \mathbf{r}_m^{p_i m}, \quad \mathbf{p}_i = \mathbf{r}_\ell^{p_i \ell}, \quad \mathbf{r} = \mathbf{r}_m^{\ell m}, \quad \mathbf{C} = \mathbf{C}_{\ell m}. \tag{2.18}$$

We also define

$$\mathbf{y} = \frac{1}{w}\sum_{i=1}^{M} w_i \mathbf{y}_i, \quad \mathbf{p} = \frac{1}{w}\sum_{i=1}^{M} w_i \mathbf{p}_i, \quad w = \sum_{i=1}^{M} w_i, \tag{2.19}$$

where w_i are scalar weights for each point.

Now that we have all the variables, we define the error term for each point:

$$\mathbf{e}_i = \mathbf{y}_i - \mathbf{C}\left(\mathbf{p}_i - \mathbf{r}\right) \tag{2.20}$$

and use it to create the cost function to minimize,

$$J(\mathbf{C}, \mathbf{r}) = \frac{1}{2}\sum_{i=1}^{M} w_i \mathbf{e}_i^T \mathbf{e}_i = \frac{1}{2}\sum_{i=1}^{M} w_i \left(\mathbf{y}_i - \mathbf{C}\left(\mathbf{p}_i - \mathbf{r}\right)\right)^T \left(\mathbf{y}_i - \mathbf{C}\left(\mathbf{p}_i - \mathbf{r}\right)\right), \tag{2.21}$$

subject to $\mathbf{C} \in SO(3)$.

The next step is to make a change of variables for the translation,

$$\mathbf{d} = \mathbf{r} + \mathbf{C}^T\mathbf{y} - \mathbf{p}, \tag{2.22}$$

such that we can rewrite the cost function as

$$J(\mathbf{C},\mathbf{d}) = \frac{1}{2}\sum_{i=1}^{M} w_i \left((\mathbf{y}_i - \mathbf{y}) - \mathbf{C}\left(\mathbf{p}_i - \mathbf{p}\right)\right)^T \left((\mathbf{y}_i - \mathbf{y}) - \mathbf{C}\left(\mathbf{p}_i - \mathbf{p}\right)\right) + \frac{1}{2}\mathbf{d}^T\mathbf{d}, \tag{2.23}$$

where the first term of the cost only depends on $\mathbf{C}$ and the second term only depends on $\mathbf{d}$. The second term can be minimized by setting $\mathbf{d} = \mathbf{0}$, which means that the translation is the difference of the centroids of the point clouds:

$$\mathbf{r} = \mathbf{p} - \mathbf{C}^T\mathbf{y}. \tag{2.24}$$

In order to find the rotation, we need to minimize the first term of the cost with respect to $\mathbf{C}$. We can multiply out parts of the cost term,

$$\begin{aligned}&\left((\mathbf{y}_i - \mathbf{y}) - \mathbf{C}\left(\mathbf{p}_i - \mathbf{p}\right)\right)^T \left((\mathbf{y}_i - \mathbf{y}) - \mathbf{C}\left(\mathbf{p}_i - \mathbf{p}\right)\right)\\ &\qquad = \left(\mathbf{y}_i - \mathbf{y}\right)^T\left(\mathbf{y}_i - \mathbf{y}\right) - 2\left(\left(\mathbf{y}_i - \mathbf{y}\right)^T\mathbf{C}\left(\mathbf{p}_i - \mathbf{p}\right)\right) + \left(\mathbf{p}_i - \mathbf{p}\right)^T\left(\mathbf{p}_i - \mathbf{p}\right),\end{aligned}$$

where only the middle term depends on $\mathbf{C}$. We sum the middle term over all the points to get

$$\begin{aligned}\frac{1}{w}\sum_{i=1}^{M} w_i\left(\left(\mathbf{y}_i - \mathbf{y}\right)^T\mathbf{C}\left(\mathbf{p}_i - \mathbf{p}\right)\right) &= \frac{1}{w}\sum_{i=1}^{M} w_i \operatorname{tr}\left(\mathbf{C}\left(\mathbf{p}_i - \mathbf{p}\right)\left(\mathbf{y}_i - \mathbf{y}\right)^T\right)\\ &= \operatorname{tr}\left(\mathbf{C}\frac{1}{w}\sum_{i=1}^{M} w_i\left(\mathbf{p}_i - \mathbf{p}\right)\left(\mathbf{y}_i - \mathbf{y}\right)^T\right)\\ &= \operatorname{tr}\left(\mathbf{C}\mathbf{W}^T\right),\end{aligned} \tag{2.25}$$

where

$$\mathbf{W} = \frac{1}{w}\sum_{i=1}^{M} w_i\left(\mathbf{y}_i - \mathbf{y}\right)\left(\mathbf{p}_i - \mathbf{p}\right)^T. \tag{2.26}$$

We define a new cost function that we can minimize with respect to $\mathbf{C}$,

$$J(\mathbf{C},\mathbf{\Lambda},\gamma) = -\operatorname{tr}\left(\mathbf{C}\mathbf{W}^T\right) + \operatorname{tr}\left(\mathbf{\Lambda}\left(\mathbf{C}\mathbf{C}^T - \mathbf{1}\right)\right) + \gamma\left(\det\mathbf{C} - 1\right), \tag{2.27}$$

where $\mathbf{\Lambda}$ and γ are Lagrange multipliers, and the associated terms are added to ensure that $\mathbf{C} \in SO(3)$ (i.e. that $\mathbf{C}\mathbf{C}^T = \mathbf{1}$ and $\det\mathbf{C} = 1$). Finally, we find the derivative of the cost,

$$\frac{\partial J}{\partial \mathbf{C}} = -\mathbf{W} + 2\mathbf{\Lambda}\mathbf{C} + \gamma\underbrace{\det\mathbf{C}}_{1}\underbrace{\mathbf{C}^{-T}}_{\mathbf{C}} = -\mathbf{W} + \mathbf{L}\mathbf{C}, \tag{2.28}$$

$$\frac{\partial J}{\partial \mathbf{\Lambda}} = \mathbf{C}\mathbf{C}^T - \mathbf{1}, \tag{2.29}$$

$$\frac{\partial J}{\partial \gamma} = \det\mathbf{C} - 1, \tag{2.30}$$

where $\mathbf{L} = 2\mathbf{\Lambda} + \gamma\mathbf{1}$. If we set the first equation equal to zero, we get

$$\mathbf{L}\mathbf{C} = \mathbf{W}. \tag{2.31}$$

At this point, SVD on the matrix $\mathbf{W}$ to find $\mathbf{C}$:

$$\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{V}^T, \tag{2.32}$$

where $\mathbf{U}$ and $\mathbf{V}$ are square, orthogonal matrices, and $\mathbf{D} = diag(d_1, d_2, d_3)$ is a diagonal matrix of singular values, $d_1 \geq d_2 \geq d_3 \geq 0$. Depending on the properties of $\mathbf{W}$, there can be one or infinitely many global solutions for $\mathbf{C}$. The different cases (Umeyama, 1991; de Ruiter and Forbes, 2013) are described in great detail in Barfoot (2017). If a unique solution for $\mathbf{C}$ exists, it is:

$$\mathbf{C} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{2.33}$$

where $\mathbf{S} = \text{diag}\,(1, 1, \det \mathbf{U}, \det \mathbf{V})$. The necessary and sufficient conditions for this unique global solution to exist are:

1. $\det \mathbf{W} > 0$, or
2. $\det \mathbf{W} < 0$ and $d_1 \geq d_2 > d_3 > 0$, or
3. $\text{rank}\, \mathbf{W} = 2$.

Cases where these conditions do not hold, and we get infinitely many solutions, are rare in practical situations.

We bring everything together to build the final solution. After estimating the rotation, $\hat{\mathbf{C}}_{\ell m}$, we use (2.24) to get the estimated translation:

$$\hat{\mathbf{r}}_m^{\ell m} = \mathbf{p} - \hat{\mathbf{C}}_{\ell m} \tag{2.34}$$

and combine everything in the transformation matrix,

$$\hat{\mathbf{T}}_{\ell m} = \begin{bmatrix} \hat{\mathbf{C}}_{\ell m} & -\hat{\mathbf{C}}_{\ell m}\hat{\mathbf{r}}_m^{\ell m} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{2.35}$$

In Chapters 5 and 6, we will make use of this non-iterative pose estimation technique in a fully differentiable pose estimation pipeline.

2.3 Visual Teach and Repeat

VT&R facilitates accurate metric and long-range autonomous path following in unstructured and GPS-denied environments (Furgale and Barfoot, 2010; Paton et al., 2018). VT&R relies on a local relative pose map removing the need for global localization. The user teaches a path by driving the robot manually, after which the path can be repeated autonomously. The overall work flow of the VT&R system is illustrated at a high level in Figure 2.6. In the teach phase visual features are extracted and used to generate landmarks and estimate poses that are stored in the map. During the repeat phase visual odometry and localization are combined in a predictor/corrector fashion to estimated the robot's offset to the mapped path. In the following sections, we describe the various components of the system in more detail.

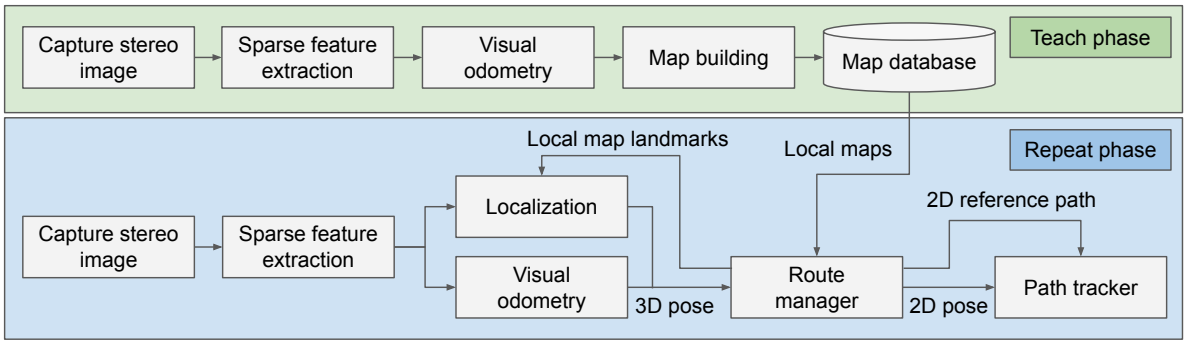


Figure 2.6: This figure is based on a figure from Furgale and Barfoot (2010). An overview of the main blocks of the VT&R system.

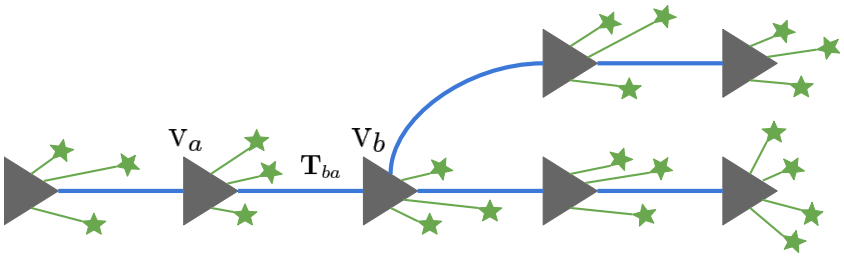


Figure 2.7: Illustration of a pose graph used to store a map in VT&R. The grey triangles represent vertices that store 3D landmarks shown as green stars. The landmark positions are given relative to the vertex reference frame. The blue edges represent the relative $SE(3)$ pose transforms between vertices, for instance T_{ba} is the transform from vertex v_a to vertex v_b.

2.3.1 Map Representation

The map for VT&R is represented as a topometric pose graph, an example of which is shown in Figure 2.7. The pose graph contains a series of vertices (shown as gray triangles), each of which has its own reference frame. The vertices store an image as well as the triangulated 3D landmarks (green stars) observed at that vertex together with their associated uncertainties and descriptors. The landmark positions are given relative to the vertex reference frame. The vertices are connected by edges (blue lines) that represent the relative $SE(3)$ pose between them, which are stored together with their uncertainty. These poses are generated using VO. Since VT&R does not keep track of global poses, only relative poses need to be stored in the graph. It is possible to create a network of paths in VT&R, and therefore the pose graph may contain both branches and loops.

2.3.2 Teach Phase

When a user teaches a new path, they drive the robot manually while a map of the path is created and stored as a pose graph. A high-level overview of the teach phase is shown in Figure 2.8. One of the main components of building the map is VO. When the stereo camera captures a new stereo image, the individual images are rectified and converted to grayscale. Next, visual features (SURF (Bay et al., 2006)) are extracted using the gpusurf library developed by Furgale and Tong (2010). Stereo matching is performed on the extracted keypoints to find the depth values needed to triangulate the 3D landmark associated with each keypoint. Each landmark has an associated descriptor and uncertainty. The VO

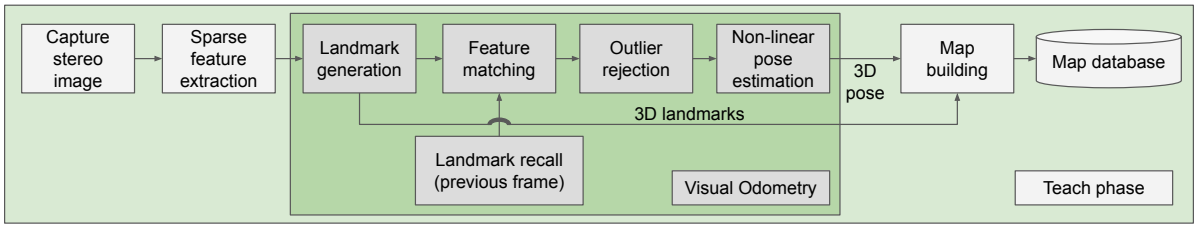


Figure 2.8: An overview of the main blocks of the VT&R teach phase. Image capture and feature extraction is followed by VO for pose estimation. The map stores vertices with 3D landmarks and the relative poses between vertices.

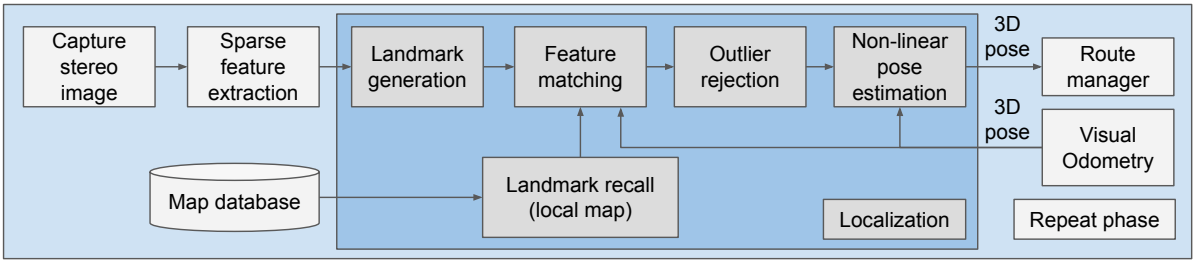


Figure 2.9: An overview of the main blocks of the VT&R repeat phase. Image capture and feature extraction is followed by VO and localization that act as pose predictor and corrector, respectively. The relative pose provides the offset to the path needed for path following.

pipeline recalls landmarks associated with the closest vertex in the map and uses nearest-neighbour matching of descriptors for data association between these and the newly detected landmarks. This means that keypoints for each incoming stereo image are tracked against the landmarks of the closest vertex in the map.

After data association, outlier rejection with Random Sample Consensus (RANSAC) (Fischler and Bolles, 1981; Raguram et al., 2013) is applied to the matched landmarks, and the resulting inliers are passed to pose estimation. The relative pose between the live frame and the map is found by minimizing the sum of squared map landmark reprojection errors after transforming and projecting the landmarks into the image plane. The errors are weighted with uncertainty from the live landmark measurements. Whenever the estimated relative pose between the live frame and the map exceeds a certain distance threshold, or the number of matched feature inliers gets too low, a new vertex is added to the pose graph. The distance threshold ensures vertices are evenly distributed and avoids a bloated map with too much data. More details on the optimization algorithm for pose estimation in VT&R can be found in Furgale and Barfoot (2010) and Paton (2018).

2.3.3 Repeat Phase

In the repeat phase, the robot traverses the taught path autonomously. An overview of the workflow is shown in Figure 2.9. The task is to localize the robot with respect to the map such that commands can be issued to stay on the path. This requires metric localization. We assume that the user provides the initial approximate topological localization by either placing the robot at the beginning of the path or indicating (in the user interface) where along the path the robot is located. An initial localization search is performed to find the closest vertex in the pose graph map. During path following, VO and

localization are run in a predictor/corrector fashion. This means that VO is used to propagate the estimated pose forward as the robot drives, and localization uses sensor measurements to correct the pose estimate. In the following we explain these operations in more detail.

As before, live images are captured with the stereo camera before they are rectified and converted to grayscale. Sparse visual features (SURF) are extracted, and the associated keypoints are triangulated to find 3D landmarks. These landmarks must then be matched to landmarks in the pose graph. Given the last known closest vertex in the graph and the pose estimate from VO, we can find the current closest vertex in the graph. Let this vertex have reference frame $\underrightarrow{\mathcal{F}}_m$. Next, VT&R extracts a local submap by relaxing a window of vertices centred at the closest vertex. The landmarks from all the vertices in the window are then transformed to the frame $\underrightarrow{\mathcal{F}}_m$. Let the frame of the live robot be $\underrightarrow{\mathcal{F}}_l$. Feature matching is performed between the landmarks in $\underrightarrow{\mathcal{F}}_l$ and $\underrightarrow{\mathcal{F}}_m$ using nearest neighbour matching of the descriptors. As in the case with VO, RANSAC is used for outlier rejection, and the pose, $\mathbf{T}_{lm}$, is found using the reprojection error of map landmarks transformed into the image plane. Additionally, the pose propagated forward with VO is used as a prior for the pose estimation. Hence, we can see the estimated pose from localization as a correction, based on feature measurements, to the pose predicted by VO. If localization fails, VT&R relies on the pose propagated by VO. Path following fails if localization has not recovered after 20 metres of driving.

The estimated $SE(3)$ relative offset to the mapped path is projected into $SE(2)$. With knowledge of the projected 2D pose, the 2D reference path, and the target velocity, the path tracker carries out path following using Model Predictive Control (Rawlings and Mayne, 2009).

2.3.4 Long-Term Localization

When relying on handcrafted features such as SURF, feature matching will fail once the difference in appearance between the live and the map images becomes too large. VT&R has been extended with new work to make localization more robust to large environmental change. Paton et al. (2015) extended the operation of VT&R from a few hours to repeating from sunrise until sunset across multiple days. They improved the localization's robustness to lighting change by transforming RGB images into colour-constant images (Corke et al., 2013).

Paton et al. (2016) further extended the long-term localization ability by introducing MEL (Churchill and Newman, 2013). Each time the robot repeats a path is referred to as an experience. The data for each repeat is then added to the pose graph map such that we get a spatio-temporal pose graph, see Figure 2.10. This means that we get vertices in the graph for each repeat. In addition to the temporal edges (in blue) that record the relative transforms between vertices on the teach, or privileged, run, we also get spatial edges (green dashed line) that represent the relative pose transforms between vertices from each new experience back to the closest vertex on the teach run. These poses are obtained with localization during repeats. Finally, the temporal relative transformations (blue dashed line) between adjacent vertices, obtained with VO, are also recorded for each experience.

MEL tackles drastic appearance change by using intermediate experiences to bridge the appearance gap. We assume that the robot has been repeating a path as the environment gradually changes and storing the data from each experience. When localizing across significant environmental change, VT&R can pick a subset of the stored experiences that are most similar to the current conditions and use the visual features from these experiences to localize instead of or together with the features stored with the first mapping run. MacTavish et al. (2018) provide details on different ways to pick the relevant

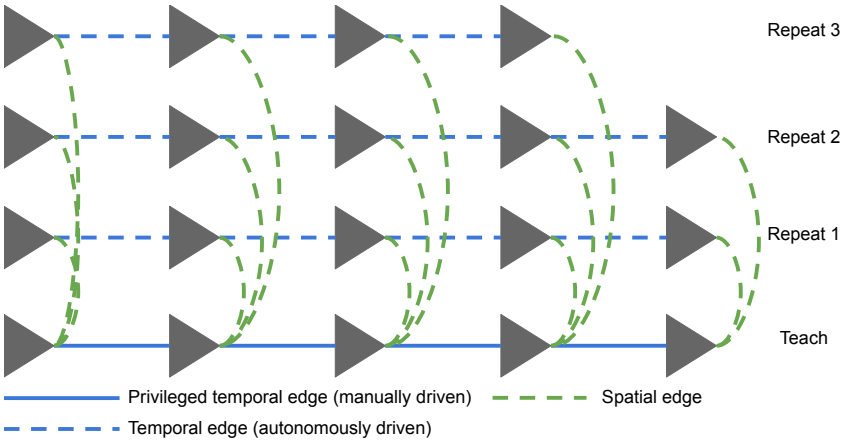


Figure 2.10: The spatio-temporal pose graph for MEL stores the data from each repeat as an experience in the pose graph. Vertices on the teach are connected by temporal edges (blue line) that represent the relative $SE(3)$ pose between them. Repeat vertices are connected to a vertex on the map via a spatial edge (green dashed line) that represents the relative pose from localization. Finally, the temporal edges (dashed blue line) between adjacent vertices on the repeats represent the relative pose provided by VO.

experiences. MEL has extended VT&R operation to a full day of lighting change, including driving after dark (MacTavish et al., 2017), and localization across seasons in an experiment that was run from January until May (Paton, 2018). The downside to MEL is the fact that experiences must be collected continually as the environment changes. Moreover, they must be gathered independently for each path even though two paths may be collected in similar environments. In this thesis, we work towards making single-experience localization more robust to appearance change with the goal of localizing without intermediate bridging experiences.

2.4 Datasets

Throughout the work for this thesis, we make use of datasets that were collected outdoors using multi-experience VT&R. In this section, we provide details on each dataset and describe how we use the data in our work. As explained in Section 2.3.4, multi-experience VT&R stores sensor data with vertices and relative poses as edges between the vertices in a spatio-temporal pose graph. It contains temporal edges between adjacent keyframes on a given run and spatial edges from repeat vertices back to the teach vertices to which they are localized.

The data used in this thesis were all collected outdoors at the University of Toronto Institute for Aerospace Studies (UTIAS). Figure 2.11 shows an overhead view of the area. The data were gathered during autonomous path following with the Clearpath Grizzly robot (see Figure 2.12), which uses a factory-calibrated FLIR Bumblebee XB3 stereo camera with 24 cm baseline and 16 Hz frame rate as the only sensor for VT&R. The camera is mounted on a mast and the extrinsic transformation between the robot base and the camera is hand-measured. The robot has two forward-facing and two backward-facing LED lights that can be used while driving during the dark. There is also a Navtech GPS unit, which was not used for the data we work with.

The VT&R data is stored in a pose graph structure that can be accessed with Python code. In order to make the data easier to handle for our own work, and to make it more accessible to others, we

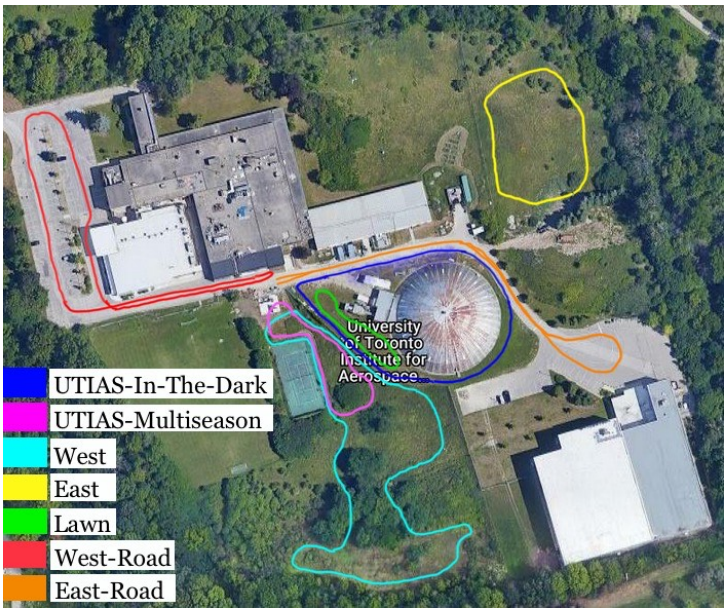


Figure 2.11: Aerial view of UTIAS, with the approximate path for each dataset overlayed.

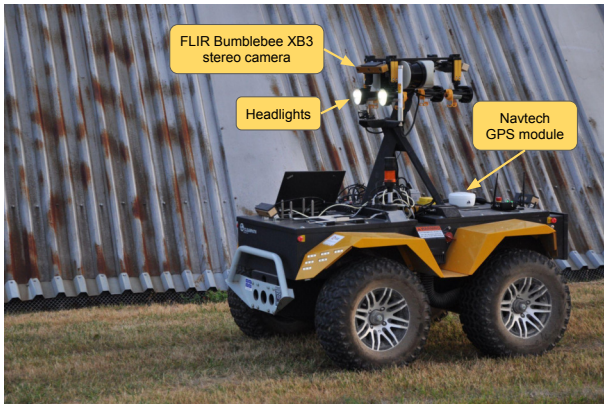


Figure 2.12: The Clearpath Grizzly robot with a forward-facing FLIR Bumblebee XB3 stereo camera, headlights, and a Navtech GPS.

worked together with another student, Ben Congram, to store the VT&R data in easier-to-use formats. The UTIAS Long-Term Mapping and Localization dataset is accessible online [1] and comes with Python tools for reading vertices and transforms from the pose graph. The first two datasets that we explain below, are available under the UTIAS Long-Term Mapping and Localization dataset.

2.4.1 UTIAS In-The-Dark

In summer 2016, MacTavish et al. (2017) used multi-experience VT&R to repeat a path across all lighting conditions over a 31-hour time period between 09:00 on July 19th and 15:30 the following day. We refer to the data from this experiment as the UTIAS In-The-Dark dataset. The path is approximately 250 metres long and was collected on the road and lawn around the Mars Dome building. Small traffic cones were set up to cordon off the experiment from other vehicles along the road. Figure 2.13 shows the overhead

[1] Data available at http://asrl.utias.utoronto.ca/datasets/2020-vtr-dataset.

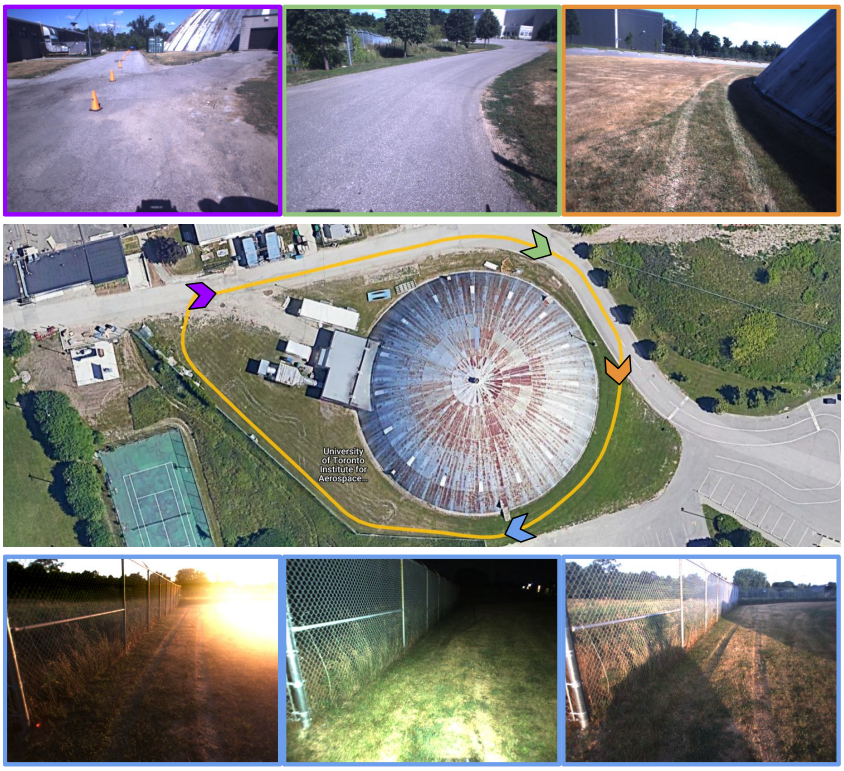


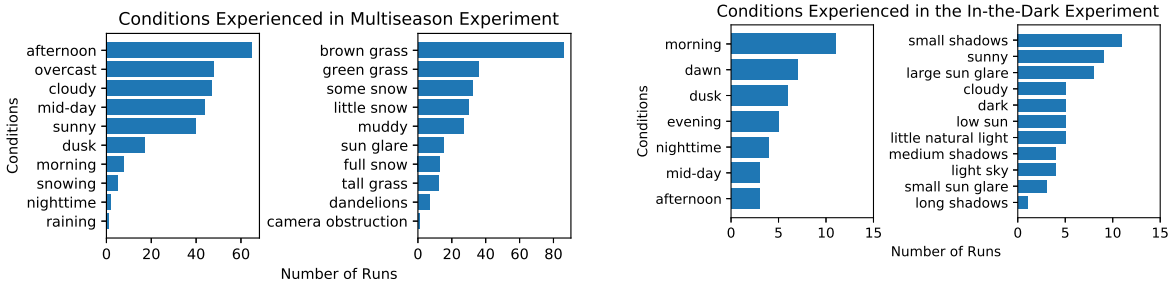
Figure 2.13: Overhead view of the UTIAS In-The-Dark path. Example images from different places along the path are indicated with coloured markers on the map. The images on the bottom row with blue frames show different appearance conditions from the same location.

view of the path and some example images from different locations along the path and from different lighting conditions. Figure 2.15 summarizes the distribution of different conditions in the data. This dataset contains a full range of lighting, including challenging conditions such as strong sun flares during sunrise and sunset, long shadows, and driving with headlights after dark. Since the weather remains the same throughout the experiment, apart from a few instances with cloud cover causing some more diffuse lighting, this dataset nicely isolates the lighting change from longer-term weather and structural changes. All together the dataset contains 39 experiences and a total of 72,666 stereo image pairs. The number of stereo pairs per run varies between 862 and 4042. In general, more keyframes were created in runs with more challenging visual conditions as new vertices are created in the pose graph when the number of matched features during VO drop below a certain threshold.

2.4.2 UTIAS Multiseason

Over 17 weeks from January 31st til May 11th, 2017, Paton (2018) repeated the same path across different lighting, weather, and seasonal conditions. We refer to the data from this experiment as the UTIAS Multiseason dataset. The 165 metre path is driven off-road in an area with more undulating terrain, sharper turns, denser vegetation, and fewer permanent structures compared to the UTIAS In-The-Dark dataset. The dataset comprises conditions with varying degrees of snow accumulation, snow melting, muddy conditions, and green grass. Different lighting conditions, including driving during the dark, and different weather conditions are also present in the 136 total runs of the path. Figure 2.15

Figure 2.14: Overhead view of the UTIAS Multiseason path. Example images from different places along the path are indicated with coloured markers on the map. The images on the right with blue frames show different appearance conditions from the same location.



(a) UTIAS Multiseason dataset.

(b) UTIAS In-The-Dark dataset.

Figure 2.15: Summary of weather, lighting, and ground conditions experienced during the 39 runs of the In-The-Dark dataset (a) and 136 runs of the UTIAS Multiseason dataset (b).

summarizes the distribution of different conditions. A total of 86,883 stereo image pairs are provided. An overhead view of the path together with example images are can be seen in Figure 2.14.

2.4.3 Additional Paths

During spring and early summer in 2019, we collected data from new paths around UTIAS to use as additional testing data. In particular, for learning-based algorithms, we want to test generalization to paths and areas not seen in the larger UTIAS In-The-Dark and UTIAS Multiseason datasets we use for training. We have named the paths East, West, Lawn, West-Road, and East-Road. Each of these paths have fewer repeats and also less appearance change between repeats compared to the UTIAS In-The-Dark and UTIAS Multiseason datasets. However, they were intended to test generalization to new areas and not for tackling a large range of appearance conditions.

The West path is approximately 470 metres long and driven off-road in a vegetated area with few permanent structures. We taught the path on May 15th, 2019 and collected six repeats on May 15th and May 16th, some of which include sun flares from low evening sun. See Figure 2.16 for an overhead view of the path and example images. The East path is approximately 190 metres long and is also collected in an off-road field with surrounding vegetation. In particular, it has tall grass along the entire route. We

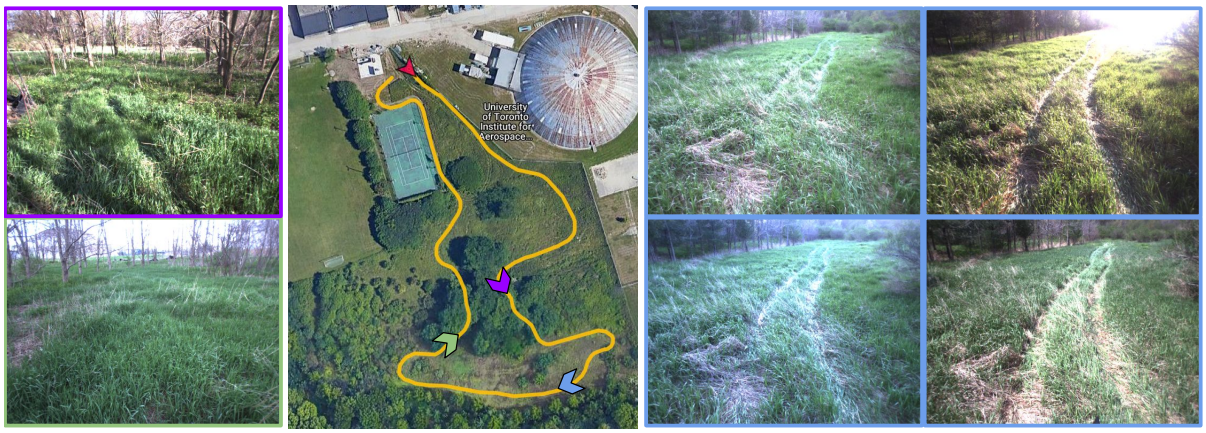


Figure 2.16: Overhead view of the West path. Example images from different places along the path are indicated with coloured markers on the map. The images on the right with blue frames show different appearance conditions from the same location.

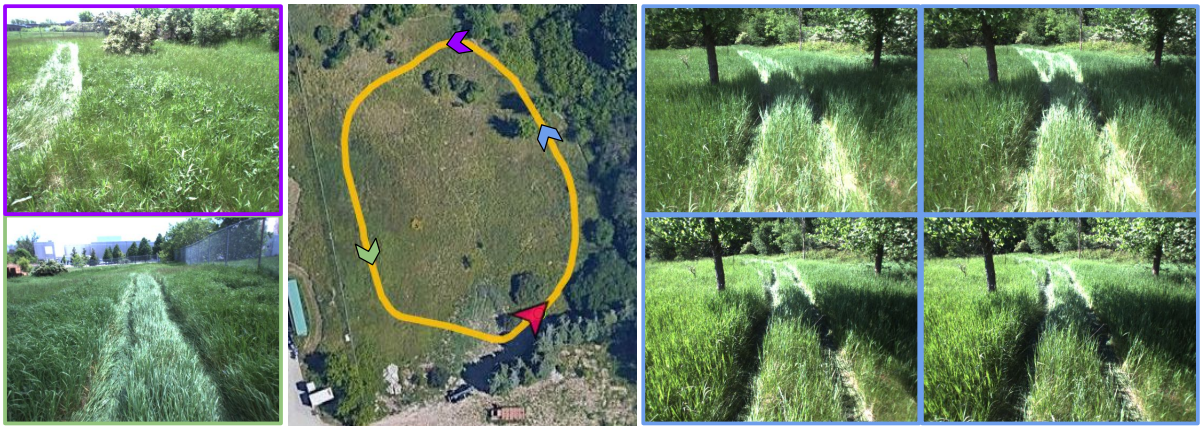


Figure 2.17: Overhead view of the East path. Example images from different places along the path are indicated with coloured markers on the map. The images on the right with blue frames show different appearance conditions from the same location.

taught the path on June 6th, 2019 and repeated it four times in the afternoon of the same day. Figure 2.17 provides a summary of the path with example images.

The Lawn path is a shorter, approximately 120 metre, route collected on the lawn next to the Mars Dome building. The path taught on June 12th, 2019, was repeated once on June 12th, three times on June 14th, and seven times on June 21st. Some of the data contains long shadows. An overhead view of the path and images from different locations along the path are provided in Figure 2.18.

The West-Road path was taught in the evening on June 21st, 2019 and repeated once on June 21st and twice on June 25th. The robot drives along the road and the path drives around a parking lot. Some of the runs are gathered during the evening with low sun and long shadows. Figure 2.19 displays an overhead view of the path and example images from the robot's camera. The East-Road path was also collected along a road and contains a large turn next to a smaller parking lot. The path is approximately 410 metres. It was taught on June 14th, 2019 and repeated twice on June 14th and twice more on June 21st. See Figure 2.20 for a summary including an overhead view and example images.

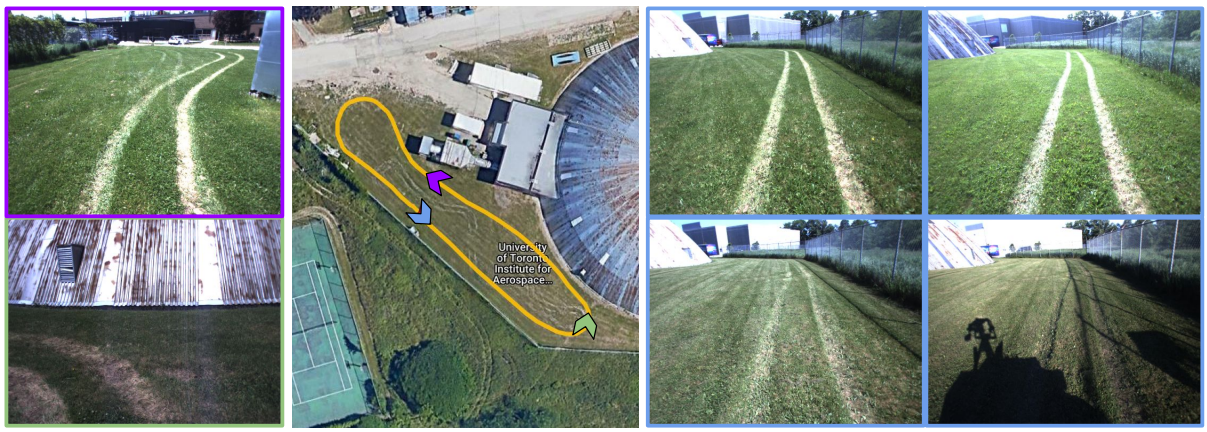


Figure 2.18: Overhead view of the Lawn path. Example images from different places along the path are indicated with coloured markers on the map. The images on the right with blue frames show different appearance conditions from the same location.

Figure 2.19: Overhead view of the West-Road path. Example images from different places along the path are indicated with coloured markers on the map. The images on the right with blue frames show different appearance conditions from the same location.

2.4.4 Generating Datasets for Deep Learning

For the work on deep learning in Chapters 4, 5, and 6, we sample training, validation, and test data from the VT&R spatio-temporal pose graph. We use the relative poses estimated with multi-experience VT&R as ground truth for training. Although this ground truth is computed using VT&R and not a typical GPS-based system such as RTK-GPS (using a base station) or GPS-INS (inertial navigation system), we believe it can be used to supervise training due to the high accuracy of VT&R path following with centimeter-level error on kilometer-scale repeats (Clement et al., 2017). A data sample consists of two sets of stereo images together with the relative pose between them. For a VO sample, the images are chosen from two vertices on the same repeat, while a localization sample contains images from vertices on two different repeats, see Figure 2.21.

Since the UTIAS In-The-Dark and UTIAS Multiseason dataset have many experiences (39 and 136, respectively), we can generate a large number of samples for training and validation by randomly sampling from the spatio-temporal pose graph. In the case of VO, we can vary the size of the relative

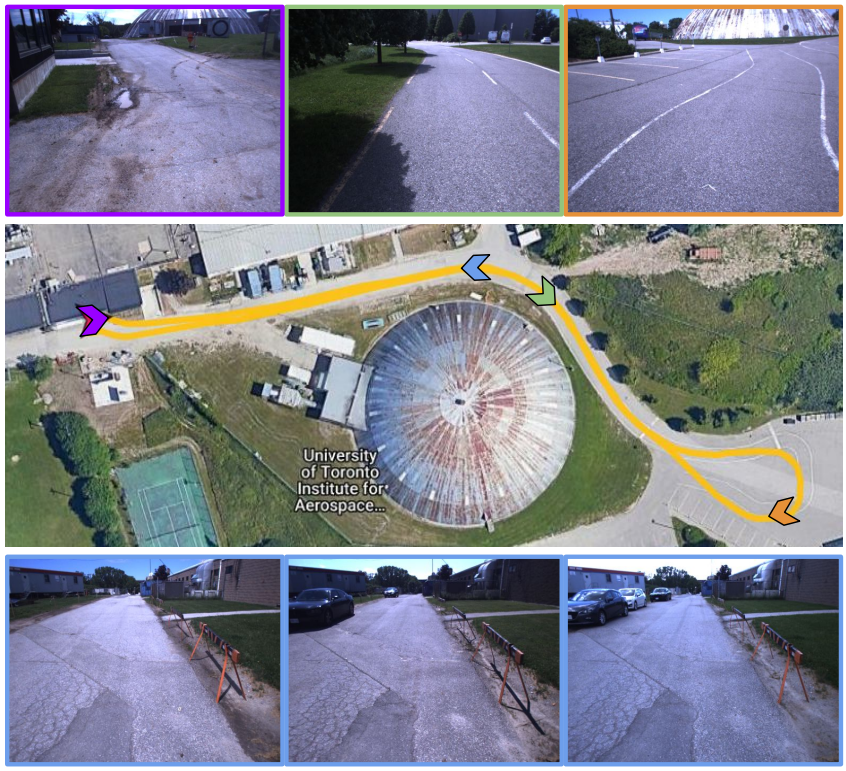


Figure 2.20: Overhead view of the East-Road. Example images from different places along the path are indicated with coloured markers on the map. The images on the bottom row with blue frames show different appearance conditions from the same location.

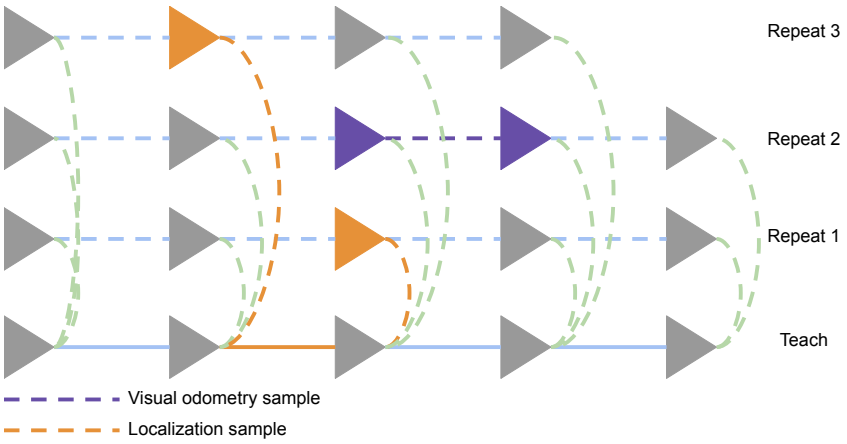


Figure 2.21: Sampling relative poses for VO and localization from the spatio-temporal pose graph. The sample in purple shows an example of how we get the temporal relative pose between two vertices on the same run. This can be used as VO ground truth for training. An example of getting a relative pose for localization ground truth is shown in orange.

pose by either choosing adjacent keyframes (purple sample in Figure 2.21) or compound poses between keyframes that are farther apart. Since VT&R localizes all repeats back to the teach run, we can find the relative pose between vertices from any two repeats. We do so by compounding the relative poses from the vertex on one repeat, via the teach vertex, to a vertex on a another repeat, see the sample

marked in orange in Figure 2.21. Similarly to VO, we can choose two vertices connected directly via the teach vertex, or we can move temporally along the teach run to find a vertex farther away and generate a relative pose with larger magnitude. The combination of many repeats and the ability to move both spatially and temporally in the pose graph, lets us generate large numbers of localization samples. Since VT&R accomplishes very accurate path following, many samples, apart from on turns along the path, will not provide a large change in viewpoint. We easily find samples with drastically different appearance, but a small portion of potential samples have large viewpoint change. Advancing along the graph temporally to sample vertices that are farther apart helps remedy this challenge to a certain extent.

When extracting data for experiments to test the performance of deep learning methods, we do not sample randomly from the pose graph but rather collect all of the samples sequentially. Specifically, the samples come from the same repeat for VO or from a pair of repeats for localization, where we use one run as the teach and the other as the repeat. Further details on the training, validation, and test datasets that we create will be provided in Chapters 4, 5, and 6.

2.4.5 Comparison to Public Datasets

As described above the data we use captures stereo images and associated poses from different outdoor paths across lighting, weather, and seasonal change. There exist many datasets that can be used to test localization and mapping. Some are based on images of city scenes or landmarks taken from many different viewpoints (Li et al., 2012; Sattler et al., 2012; Kendall et al., 2015; Li and Snavely, 2018) or a static camera (Chen et al., 2017) that captures change over time, but do not collect data along a path. Other datasets provide data from driving, though the paths are never or only seldom traversed more than once (Smith et al., 2009; Geiger et al., 2012; Blanco et al., 2014; Wang et al., 2017). We focus on comparing the datasets we use to others that follow a similar approach, i.e. datasets for localization from paths that have been traversed more than once.

The Oxford Robotcar Dataset (Maddern et al., 2017) was gathered while driving a car through central Oxford during different seasons, weather, and lighting conditions over a total distance of approximately 1000 km. The 4Seasons dataset (Wenzel et al., 2020) records 350 km cross-seasonal driving in nine areas ranging from urban to countryside environments. The University of Michigan North Campus Long-Term Vision and Lidar Dataset (Carlevaris-Bianco et al., 2015) covers 27 runs across 15 months from indoor and outdoor environments at the University of Michigan campus. The Ford Campus Vision and Lidar Data Set (Pandey et al., 2011) also collects data in an urban environment, however the paths are only traversed a few times during the span of a month. In their Bovasia dataset, which is a part of the Rawseed Project, Ceriani et al. (2009) provide six sessions of mixed indoor and outdoor paths on a campus. Another campus dataset is the Cross Season University Campus Dataset (Masatoshi et al., 2015). The authors collect data from a hand-held camera along three different paths over four seasons with a total of 12 traversals per path. The CMU Visual Localization Dataset repeats the same path 16 times during a year (Badino et al., 2011). Later, the CMU Seasons Dataset (Sattler et al., 2018) has been created using data from (Badino et al., 2011). Two traversals of an 8 km path through a suburb, one collected during heavy rain at night time and the other during a clear morning, is presented in the Alderley Day/Night Dataset (Milford and Wyeth, 2012). The Nordland dataset consist of images from a train journey completed during four different seasons and has been applied to robotics research (Sünderhauf et al., 2013). The Symphony Lake dataset provides 121 surveys of an approximately 1.3 km

lakeshore and includes appearance change across lighting, weather, and seasons (Griffith et al., 2017). Finally, (Sattler et al., 2018) have created a benchmark for challenging visual localization. It contains several different datasets that provide a range of different scenes and associated ground truth poses.

Since the datasets collected at UTIAS were captured while running multi-experience VT&R in the loop, the robot follows the taught path closely and the viewpoint change is small over time (MacTavish et al., 2018). This is different from datasets where paths are collected manually for each session. The Nordland Dataset was collected by a train and therefore also provides minimal viewpoint change. In contains a total of four traversals, one for each season.

The UTIAS In-The-Dark dataset systematically captures lighting change across 31 hours by traversing the path approximately once per hour. The lighting change is not conflated with seasonal, weather, and structural change as in other long-term localization and mapping datasets. Moreover, the data cover challenging scenarios such as driving during the night and sun flares, see Figure 2.13.

Similarly to the New College Dataset (Smith et al., 2009) and other datasets collected in a campus environment, the UTIAS In-The-Dark dataset (along with Lawn, West-Road and East-Road) provide imagery of a mixed environment with road, buildings, fencing, and greenery. The UTIAS Multiseason dataset (along with West and East) contain off-road areas with sharp turns, undulating terrain, and vegetation. These off-road conditions, ranging from thick snow cover to tall grass, facilitate extreme appearance change across seasons, as seen in Figure 2.14. The absence of road together and fewer permanent structures than seen in urban datasets (Carlevaris-Bianco et al., 2015; Maddern et al., 2017; Wenzel et al., 2020) lead to additional challenges for localization. Vegetation or heavy snow cover can also create difficulties within a traversal, e.g., moving grass and vegetation are demanding for stereo-matching and feature matching for VO. The Symphony Lake Dataset also captures data in a natural scenery, but the route is only repeated approximately and stereo imagery is not provided.

Chapter 3

Direct Localization in Visual Teach & Repeat

In this chapter we make a first attempt at improving robustness of single-experience localization in VT&R by replacing sparse feature-based pose estimation with a direct method that uses the whole image without relying on feature extraction and matching.

3.1 Introduction

Attention has been brought to direct methods for visual navigation (Comport et al., 2010; Newcombe et al., 2011; Engel et al., 2014) as an alternative to feature-based ones and have been shown to have several advantages. Direct methods minimize a photometric error based on direct image registration as opposed to sparse feature matching. This removes the intermediate steps of feature computation and matching and their associated errors. There is variation in how much of the image is used by different direct methods ranging from dense approaches using all the pixels to (Newcombe et al., 2011), to semi-dense (Engel et al., 2014) or even sparse approaches (Engel et al., 2018). Most direct methods lack the sparseness of feature-based approaches, but with the use of code parallelization they can run in real-time. Direct methods are advantageous for images with motion blur and camera defocus (Newcombe et al., 2011) as well as for low-texture areas that make feature detection challenging.

We want to use a direct method for localization in VT&R to investigate whether using the intensities from the whole image can provide some improved robustness to appearance change and improved accuracy to pose estimation over using sparse features. In this chapter, we implement direct localization for VT&R using the image alignment method presented in Baker and Matthews (2001). VO remains unchanged from the original feature-based implementation. The map is created by using VO to build a pose graph, where vertices are linked by relative pose transformations. During autonomous repeats, we combine feature-based VO and direct localization to obtain pose estimates relative to the path.

We test our direct localization approach for VT&R on two real-world outdoor paths. The first path covers a loop on flat terrain over asphalt and grass, while the second path runs off-road over undulating terrain. Because the data was collected using the existing VT&R system, we have the opportunity to directly compare feature-based localization and direct localization on the same data. We show that direct localization with a basic photometric cost function can provide increased accuracy for repeats

under nominal lighting conditions. However, the direct method struggles in the presence of large lighting change and will require further work to handle the more extreme cases well.

To summarize, the novel contribution of this chapter is the implementation of direct pose estimation for localization in VT&R. As opposed to the existing localization approach, it does not rely on sparse feature extraction and matching. The content in this chapter appeared in the Conference on Computer and Robot Vision (CRV) (Gridseth and Barfoot, 2019).

3.2 Related Work

Direct methods for visual navigation differ from feature-based approaches by relying on image registration directly instead of calculating and matching sparse visual features. They allow us to use more data and avoid the step of computing either manually designed or learned features. As mentioned in the introduction, direct methods can vary by the amount of pixels they use varying from dense (Newcombe et al., 2011), to semi-dense (Engel et al., 2014), and sparse approaches (Engel et al., 2018). Direct methods provide an advantage in areas with low texture, where it is difficult to compute features, as well as for images with motion blur or camera defocus (Newcombe et al., 2011). Despite using more data, direct methods can run in real time with the help of parallelization, choosing pixels with strong gradients, and algorithm approximations such as the inverse compositional formulation (Baker and Matthews, 2001) that significantly reduces the number of Jacobian calculations.

The goal of the original Lucas-Kanade algorithm (Lucas and Kanade, 1981; Baker and Matthews, 2004) was to align a template image to an input image. Later, Baker and Matthews (2001) developed the inverse compositional algorithm, which improved computational speed because it avoided calculating the image gradient and warp derivative for every iteration of the optimization. Most subsequent works in the area of direct image alignment are based on this algorithm.

3.2.1 Direct Visual Odometry

Several researchers have used the direct approach for VO. Comport et al. (2010) use a quadrifocal relationship between image intensities in sequential stereo pairs to construct their iterative non-linear pose estimation. Engel et al. (2018) present their Direct Sparse Odometry, in which they optimize a direct photometric error without incorporating a geometric prior. A common problem with the photometric error formulation in direct methods is the brightness constancy assumption (Newcombe et al., 2011). Alismail et al. (2017) improve robustness to illumination change by densely calculating a binary descriptor for each pixel instead of using only intensity. This involves computing descriptors, but does not require data association by descriptor matching as is the case in sparse feature-based methods. Jaramillo et al. (2017) also rely on dense matching of descriptors, but experiment with existing descriptors such as Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) and learned descriptors extracted from CNNs. In their paper, Forster et al. (2014) combine direct and feature-based methods to form a Semi-Direct Visual Odometry.

3.2.2 Direct Visual Localization and Mapping

Direct localization and mapping has also received some attention. With their Dense Tracking and Mapping system, Newcombe et al. (2011) perform real-time tracking of a monocular camera and reconstruct a

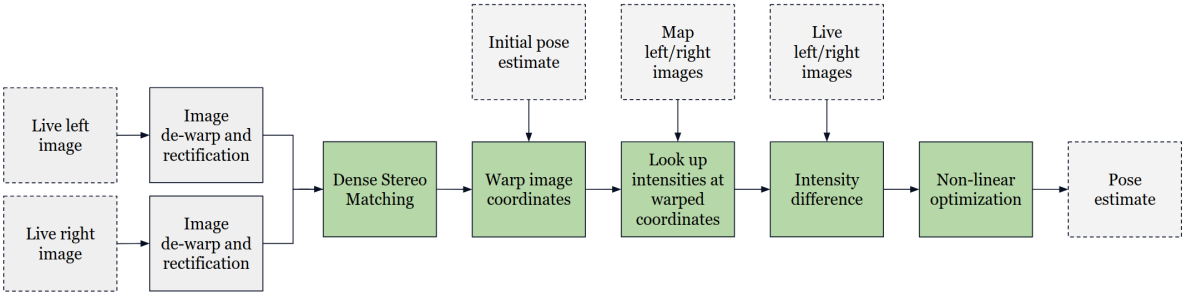


Figure 3.1: The localization pipeline illustrated from input live stereo images on the left to the final pose estimate on the right. Boxes marked in green correspond to components that are new for direct localization compared to the original feature-based implementation.

dense model of the scene. The authors show that using a dense model provides superior camera tracking during rapid motion compared to feature-based methods. With LSD-SLAM (Engel et al., 2014, 2015), the authors demonstrate that they can use direct Simultaneous Localization and Mapping (SLAM) to build consistent maps of large-scale environments. In order to tackle illumination and long-term environmental change in an outdoor urban environment, Pascoe et al. (2017) align images using Normalized Information Distance, which is derived from mutual information. Park et al. (2017) compare several approaches that update the cost function for pose estimation and use dense binary descriptors to improve robustness. Clement and Kelly (2018) use a neural network to transfer the input images to new illumination and seasonal conditions before pose estimation. Finally, Dame and Marchand (2013) perform path following using a monocular camera. They use mutual information to register images of the current view to previously mapped images. Unlike VT&R, they control only 1 DOF using visual servoing instead of performing explicit 6 DOF localization.

The direct approach has also been used to localize camera images to synthetic images created from a prior model. In this case mutual information is used to compare different image modalities, a technique that has been used in the medical imaging field (Viola and Wells III, 1997). Caron et al. (2014) track objects and localize a vehicle by combining real and rendered synthetic images. Pascoe et al. (2015) align real images with synthetic images rendered from a textured 3D mesh created using LIDAR point clouds and camera images, while Stewart and Newman (2012) uses a textured point cloud directly. Wolcott and Eustice (2014) create synthetic views of the ground plane for localization. Unlike Pascoe et al. (2015) and Stewart and Newman (2012), they use LIDAR reflectance and perform a grid search instead of gradient-based optimization.

3.3 Methodology

This section explains the pipeline used for direct localization in VT&R. We start by setting up a warp function required in the inverse compositional method for pose optimization. We also discuss using a robust cost function together with some implementation considerations.

3.3.1 System Overview

As discussed in Section 2.3.4, VT&R builds a map during the teach phase that is represented as a pose graph. The vertices in the pose graph store images and landmarks and are connected by edges that store the transformation between vertices. During a repeat, VO and localization are fused to execute path following. Sparse features are used to match landmarks in the current view to landmarks in the previous frame for VO and to landmarks in the map for localization. Localization estimates a transformation back to the closest vertex on the pose graph. The initial pose guess for localization is provided by using VO to propagate forward the most recently localized pose. In our approach we will substitute the feature-based localization, that matches landmarks in the current frame to landmarks in the map, with a direct method that uses the current and map images only. The rest of the VT&R system remains the same.

Our direct visual localization approach is based on the inverse compositional method proposed by Baker and Matthews (2001). We construct a cost function from the intensity difference between the live stereo images and those associated with the corresponding vertex on the mapped path. Minimizing this cost function with iterative Gauss-Newton optimization yields the relative pose transform between the robot and the mapped path. See Figure 3.1 for an overview of the localization pipeline.

3.3.2 Warp Function

First we construct a function that warps image coordinates. For this we need the stereo camera model that was defined in Section 2.2.2, $\mathbf{g} : \mathbb{R}^3 \rightarrow \mathbb{R}^4$, that maps a 3D point, $\mathbf{p} \in \mathbb{R}^3$, to stereo image coordinates, $\mathbf{q} \in \mathbb{R}^4$:

$$\mathbf{q} = \begin{bmatrix} u_\ell \\ v_\ell \\ u_r \\ v_r \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \underbrace{\begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u b \\ 0 & f_v & c_v & 0 \end{bmatrix}}_{\mathbf{M}} \frac{1}{z} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\mathbf{p}} = \mathbf{M} \frac{1}{\mathbf{c}^T \mathbf{p}} \mathbf{p}, \tag{3.1}$$

where $\mathbf{c}^T = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$ and the stereo camera frame coincides with the left camera frame. Each stereo image coordinate pair is assigned intensity values

$$\mathbf{y}(\mathbf{q}) = \begin{bmatrix} \mathcal{I}_\ell(u_\ell, v_\ell) \\ \mathcal{I}_r(u_r, v_r) \end{bmatrix}, \tag{3.2}$$

where $\mathcal{I}_\ell$ and $\mathcal{I}_r$ are the left and right images, respectively. The inverse of the stereo camera model maps stereo image coordinates to a homogeneous point:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{g}^{-1}(\mathbf{q}) = \begin{bmatrix} \frac{b}{u_\ell - u_r}(u_\ell - c_u) \\ \frac{b f_u}{f_v (u_\ell - u_r)}(v_\ell - c_v) \\ \frac{b f_u}{u_\ell - u_r} \\ 1 \end{bmatrix}. \tag{3.3}$$

We perform stereo matching to obtain disparity by using Hirschmuller (2008) as implemented in OpenCV. We recall the warp function defined in Section 2.2.2. Given a pose transformation, $\mathbf{T} \in SE(3)$, we want

to transform the stereo image coordinates, $\mathbf{q}$, from one frame to stereo image coordinates, $\mathbf{q}'$, in another frame using the warp function, $\mathbf{w} : SE(3) \times \mathbb{R}^4 \to \mathbb{R}^4$:

$$\mathbf{q}' = \mathbf{w}(\mathbf{T}, \mathbf{q}) = \mathbf{g}\left(\mathbf{T}\,\mathbf{g}^{-1}(\mathbf{q})\right) = \mathbf{g}\left(\mathbf{T}\,\mathbf{p}\right) = \mathbf{M}\frac{1}{\mathbf{c}^T\mathbf{T}\mathbf{p}}\mathbf{T}\mathbf{p}. \tag{3.4}$$

In order to use the warp function in the direct optimization, we need to find its derivative. While doing so, we rely on some of the three-dimensional geometry described in Section 2.2.1. The direct optimization algorithm solves iteratively for the pose, $\mathbf{T}$, by making small perturbations, $\boldsymbol{\xi} \in \mathbb{R}^6$, to the current pose estimate (or pose at the operating point), $\mathbf{T}_{\text{op}}$. The pose update is applied as follows (Barfoot, 2017):

$$\mathbf{T} = \exp\left(\boldsymbol{\xi}^\wedge\right)\mathbf{T}_{\text{op}} \approx \left(\mathbf{1} + \boldsymbol{\xi}^\wedge\right)\mathbf{T}_{\text{op}}, \tag{3.5}$$

with

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\phi}\end{bmatrix}^\wedge = \begin{bmatrix}\boldsymbol{\phi}^\wedge & \boldsymbol{\rho}\\ \mathbf{0}^T & 0\end{bmatrix} \tag{3.6}$$

and

$$\boldsymbol{\phi}^\wedge = \begin{bmatrix}\phi_1\\ \phi_2\\ \phi_3\end{bmatrix}^\wedge = \begin{bmatrix}0 & -\phi_3 & \phi_2\\ \phi_3 & 0 & -\phi_1\\ -\phi_2 & \phi_1 & 0\end{bmatrix}. \tag{3.7}$$

Further expanding the expression, we find that

$$\mathbf{T}\mathbf{p} \approx \left(\mathbf{1} + \boldsymbol{\xi}^\wedge\right)\mathbf{T}_{\text{op}}\mathbf{p} = \mathbf{T}_{\text{op}}\mathbf{p} + \left(\mathbf{T}_{\text{op}}\mathbf{p}\right)^\odot\boldsymbol{\xi}, \tag{3.8}$$

using the identity

$$\boldsymbol{\xi}^\wedge\mathbf{T}_{\text{op}}\mathbf{p} \equiv \left(\mathbf{T}_{\text{op}}\mathbf{p}\right)^\odot\boldsymbol{\xi}, \tag{3.9}$$

where $\odot$ is a homogeneous-point operator (Barfoot, 2017) given by

$$\begin{bmatrix}\boldsymbol{\epsilon}\\ \eta\end{bmatrix}^\odot = \begin{bmatrix}\eta\mathbf{1} & -\boldsymbol{\epsilon}^\wedge\\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}, \tag{3.10}$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^3$ and $\eta \in \mathbb{R}$. By combining a first order Taylor expansion with (3.8) we get that:

$$\frac{1}{\mathbf{c}^T\mathbf{T}\mathbf{p}} \approx \frac{1}{\mathbf{c}^T\mathbf{T}_{\text{op}}\mathbf{p}} - \frac{1}{\left(\mathbf{c}^T\mathbf{T}_{\text{op}}\mathbf{p}\right)^2}\mathbf{c}^T\left(\mathbf{T}_{\text{op}}\mathbf{p}\right)^\odot\boldsymbol{\xi}. \tag{3.11}$$

Substituting (3.8) and (3.11) into the warp in (3.4) lets us update the warp to include the pose perturbation $\boldsymbol{\xi}$:

$$\begin{aligned}
\mathbf{w}(\mathbf{T}, \mathbf{q}) &\approx \mathbf{M}\left(\frac{1}{\mathbf{c}^T\mathbf{T}_{\text{op}}\mathbf{p}} - \frac{1}{\left(\mathbf{c}^T\mathbf{T}_{\text{op}}\mathbf{p}\right)^2}\mathbf{c}^T\left(\mathbf{T}_{\text{op}}\mathbf{p}\right)^\odot\boldsymbol{\xi}\right)\left(\mathbf{T}_{\text{op}}\mathbf{p} + \left(\mathbf{T}_{\text{op}}\mathbf{p}\right)^\odot\boldsymbol{\xi}\right)\\
&= \underbrace{\mathbf{M}\frac{1}{\mathbf{c}^T\mathbf{T}_{\text{op}}\mathbf{p}}\mathbf{T}_{\text{op}}\mathbf{p}}_{\mathbf{w}(\mathbf{T}_{\text{op}},\mathbf{q})} + \underbrace{\mathbf{M}\frac{1}{\mathbf{c}^T\mathbf{T}_{\text{op}}\mathbf{p}}\left(\mathbf{1} - \frac{1}{\mathbf{c}^T\mathbf{T}_{\text{op}}\mathbf{p}}\mathbf{T}_{\text{op}}\mathbf{p}\mathbf{c}^T\right)\left(\mathbf{T}_{\text{op}}\mathbf{p}\right)^\odot}_{\mathbf{W}(\mathbf{T}_{\text{op}},\mathbf{q})}\boldsymbol{\xi}\\
&= \mathbf{w}(\mathbf{T}_{\text{op}}, \mathbf{q}) + \mathbf{W}(\mathbf{T}_{\text{op}}, \mathbf{q})\,\boldsymbol{\xi},
\end{aligned} \tag{3.12}$$

where we have kept only terms linear in $\boldsymbol{\xi}$, and $\mathbf{W}(\cdot,\cdot)$ is the derivative (Jacobian) of the warp function with respect to pose.

3.3.3 Inverse Compositional Approach

We can set up our pose optimization problem using the inverse compositional approach introduced by Baker and Matthews (2001). The purpose of the optimization is to find the pose, $\mathbf{T}$, that minimizes the following cost function:

$$J(\mathbf{T}) = \frac{1}{2}\sum_i \mathbf{e}(\mathbf{T},\mathbf{q}_i)^T\mathbf{e}(\mathbf{T},\mathbf{q}_i), \tag{3.13}$$

where the residual error, $\mathbf{e}(\mathbf{T},\mathbf{q}_i)$, is defined as

$$\mathbf{e}(\mathbf{T},\mathbf{q}_i) = \mathbf{y}'(\mathbf{w}(\mathbf{T},\mathbf{q}_i)) - \mathbf{y}(\mathbf{q}_i). \tag{3.14}$$

$\mathbf{y}'(\cdot)$ and $\mathbf{y}(\cdot)$ are stereo images taken from two different camera poses.

Assume that we have an initial guess for the pose, $\mathbf{T}_{\text{op}}$. Using the compositional property of the warp function, we can write write the error as

$$\begin{aligned}\mathbf{e}(\mathbf{T},\mathbf{q}_i) &= \mathbf{y}'(\mathbf{w}(\mathbf{T},\mathbf{q}_i)) - \mathbf{y}(\mathbf{q}_i) \\ &= \mathbf{y}'(\mathbf{w}(\exp(\boldsymbol{\xi}^\wedge),\mathbf{w}(\mathbf{T}_{\text{op}},\mathbf{q}_i))) - \mathbf{y}(\mathbf{q}_i).\end{aligned} \tag{3.15}$$

In order to use a change of variables, we define

$$\mathbf{q}_{i,\text{op}} = \mathbf{w}(\mathbf{T}_{\text{op}},\mathbf{q}_i), \tag{3.16}$$

from which we get

$$\mathbf{q}_i = \mathbf{w}(\mathbf{T}_{\text{op}}^{-1},\mathbf{q}_{i,\text{op}}). \tag{3.17}$$

We can substitute this into the error to arrive at

$$\mathbf{e}(\mathbf{T},\mathbf{q}_i) = \mathbf{y}'(\mathbf{w}(\exp(\boldsymbol{\xi}^\wedge),\mathbf{q}_{i,\text{op}})) - \mathbf{y}(\mathbf{w}(\mathbf{T}_{\text{op}}^{-1},\mathbf{q}_{i,\text{op}})). \tag{3.18}$$

Inserting the linearized warp function from (3.12), we can linearize the error function:

$$\begin{aligned}\mathbf{e}(\mathbf{T},\mathbf{q}_i) &\approx \mathbf{y}'(\mathbf{w}(\mathbf{1},\mathbf{q}_{i,\text{op}}) + \mathbf{W}(\mathbf{1},\mathbf{q}_{i,\text{op}})\,\boldsymbol{\xi}) - \mathbf{y}(\mathbf{w}(\mathbf{T}_{\text{op}}^{-1},\mathbf{q}_{i,\text{op}})) \\ &\approx \underbrace{\mathbf{y}'(\mathbf{q}_{i,\text{op}}) - \mathbf{y}(\mathbf{w}(\mathbf{T}_{\text{op}}^{-1},\mathbf{q}_{i,\text{op}}))}_{\mathbf{e}(\mathbf{1},\mathbf{q}_{i,\text{op}})} + \underbrace{\left.\frac{\partial \mathbf{y}'}{\partial \mathbf{q}}\right|_{\mathbf{q}_{i,\text{op}}} \mathbf{W}(\mathbf{1},\mathbf{q}_{i,\text{op}})}_{\mathbf{E}(\mathbf{1},\mathbf{q}_{i,\text{op}})}\boldsymbol{\xi} \\ &= \mathbf{e}(\mathbf{1},\mathbf{q}_{i,\text{op}}) + \mathbf{E}(\mathbf{1},\mathbf{q}_{i,\text{op}})\,\boldsymbol{\xi},\end{aligned} \tag{3.19}$$

where $\frac{\partial \mathbf{y}'}{\partial \mathbf{q}}$ is the image gradient.

We make the further approximation that $\mathbf{q}_{i,\text{op}}$ is constant between iterations, which is then equivalent to Baker and Matthews (2001). This allows us to calculate the image gradient and warp derivative once at the beginning of the optimization instead of recalculating them at each step. Substituting the linearized

Figure 3.2: The live image (left), the map image (middle), and a weight image (right) displaying the weights computed by the Geman-McClure robust cost function. Dark pixels correspond to low weights. We see that areas with changing shadows are downweighted. Red pixels are excluded.

error (3.19) into the cost function (3.13) gives:

$$J(\mathbf{T}) \approx \frac{1}{2}\sum_i \left(\mathbf{e}(\mathbf{1}, \mathbf{q}_{i,\text{op}}) + \mathbf{E}(\mathbf{1}, \mathbf{q}_{i,\text{op}})\,\boldsymbol{\xi}\right)^T \left(\mathbf{e}(\mathbf{1}, \mathbf{q}_{i,\text{op}}) + \mathbf{E}(\mathbf{1}, \mathbf{q}_{i,\text{op}})\,\boldsymbol{\xi}\right). \tag{3.20}$$

This is minimized by the solution, $\boldsymbol{\xi}^\star$, to the following system of linear equations:

$$\left(\sum_i \mathbf{E}(\mathbf{1}, \mathbf{q}_{i,\text{op}})^T \mathbf{E}(\mathbf{1}, \mathbf{q}_{i,\text{op}})\right) \boldsymbol{\xi}^\star = -\sum_i \mathbf{E}(\mathbf{1}, \mathbf{q}_{i,\text{op}})^T \mathbf{e}(\mathbf{1}, \mathbf{q}_{i,\text{op}}). \tag{3.21}$$

Finally, we apply the pose perturbation to the initial guess as follows:

$$\mathbf{T}_{\text{op}} \leftarrow \exp\left(\boldsymbol{\xi}^{\star\wedge}\right) \mathbf{T}_{\text{op}}, \tag{3.22}$$

and iterate to convergence.

3.3.4 Robust Optimization

We aim to make our optimization more robust (e.g., to scene changes such as shadows) by using a robust cost function. We do so by modifying our cost function to be

$$J'(\mathbf{T}) = \frac{1}{2}\sum_i \rho\left(u(\mathbf{T}, \mathbf{q})\right), \tag{3.23}$$

with

$$u(\mathbf{T}, \mathbf{q}_i) = \sqrt{\mathbf{e}(\mathbf{T}, \mathbf{q}_i)^T \mathbf{e}(\mathbf{T}, \mathbf{q}_i)}. \tag{3.24}$$

The derivative of the cost is:

$$\frac{\partial J'(T)}{\partial \boldsymbol{\xi}} = \sum_i \frac{\partial \rho}{\partial u_i} \frac{\partial u_i}{\partial \mathbf{e}_i} \frac{\partial \mathbf{e}_i}{\partial \boldsymbol{\xi}}, \tag{3.25}$$

where

$$\frac{\partial u_i}{\partial \mathbf{e}_i} = \frac{1}{u(\mathbf{T}, \mathbf{q}_i)} \mathbf{e}(\mathbf{T}, \mathbf{q}_i)^T, \quad \frac{\partial \rho}{\partial u_i} = \frac{u(\mathbf{T}, \mathbf{q}_i)}{(1 + u(\mathbf{T}, \mathbf{q}_i)^2)^2}. \tag{3.26}$$

This gives us

$$\frac{\partial J'(T)}{\partial \boldsymbol{\xi}} = \sum_i \alpha_i \mathbf{e}(\mathbf{T}, \mathbf{q}_i)^T \frac{\partial \mathbf{e}(\mathbf{T}, \mathbf{q}_i)}{\partial \boldsymbol{\xi}}, \tag{3.27}$$

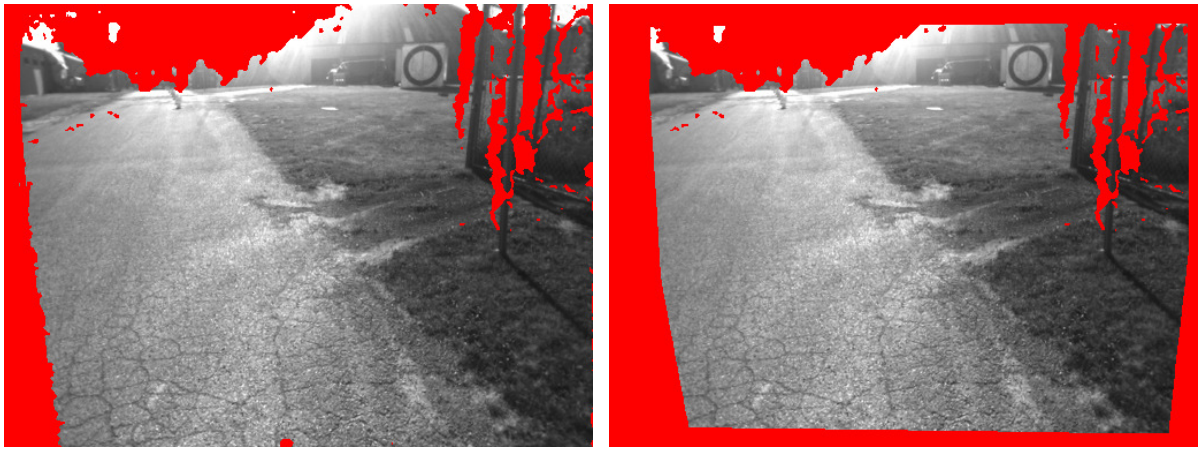


Figure 3.3: On the left we have an image that has been warped with the initial pose guess, $\mathbf{T}_{\text{op}}$. Invalid coordinates are shown using a red mask. The mask on the right is created by further perturbing the pose in individual DOF in a small area around $\mathbf{T}_{\text{op}}$. This gives us a buffer so the mask will stay the same while the pose updates during iterations.

where

$$\alpha_i = \frac{1}{u(\mathbf{T}, \mathbf{q}_i)} \left.\frac{\partial \rho}{\partial u}\right|_{u(\mathbf{T}_{\text{op}}, \mathbf{q}_i)}. \tag{3.28}$$

There are several common robust cost functions to choose from and we opt to use the Geman-McClure robust cost (Geman et al., 1992):

$$\rho(u) = \frac{1}{2}\frac{u^2}{(1+u^2)}, \tag{3.29}$$

which gives the following weight:

$$\alpha_i = \frac{1}{u(\mathbf{T}, \mathbf{q}_i)} \left.\frac{\partial \rho}{\partial u}\right|_{u(\mathbf{T}_{\text{op}}, \mathbf{q}_i)} = \frac{1}{\left(1 + \mathbf{e}(\mathbf{T}_{\text{op}}, \mathbf{q}_i)^T \mathbf{e}(\mathbf{T}_{\text{op}}, \mathbf{q}_i)\right)^2}. \tag{3.30}$$

In our experiments, we implement this robust optimization as Iteratively Reweighted Least Squares (Holland and Welsch, 1977) with the original cost function modified to include a weight for each term:

$$J''(\mathbf{T}) = \frac{1}{2}\sum_i \alpha_i \mathbf{e}(\mathbf{T}, \mathbf{q}_i)^T \mathbf{e}(\mathbf{T}, \mathbf{q}_i). \tag{3.31}$$

Our optimal pose perturbation is now given by the solution to

$$\left(\sum_i \alpha_i \mathbf{E}(\mathbf{T}_{\text{op}}, \mathbf{q}_i)^T \mathbf{E}(\mathbf{T}_{\text{op}}, \mathbf{q}_i)\right) \boldsymbol{\xi}^\star = -\sum_i \alpha_i \mathbf{E}(\mathbf{T}_{\text{op}}, \mathbf{q}_i)^T \mathbf{e}(\mathbf{T}_{\text{op}}, \mathbf{q}_i), \tag{3.32}$$

with α_i updating at each iteration. Figure 3.2 shows an example image with the pixels coloured by α_i; we see that the large shadow has been downweighted as it was not present in the reference image.

3.3.5 Implementation Considerations

For the pose optimization to succeed, we want the initial pose guess to fall inside the cost function's basin of convergence. As explained in Engel et al. (2014), a sufficiently good initialization is typically

Figure 3.4: The leftmost column shows aerial views of the two paths we are repeating, while the remaining images show example views form the two paths (by row). The bottom left image shows a view of the path in winter, though we work with data collected in spring. The first path provides challenges to localization due to changing illumination, while the second path contains a lot of vegetation, especially tall grass, that can make it difficult to extract and match sparse features.

available for VO or tracking of new camera frames, while this is not necessarily the case for loop closures. Similarly, localization with direct methods is more challenging than VO. Since VT&R solves a tracking problem we get a good initial pose estimate, $\mathbf{T}_{\text{op}}$, from propagating the previous localization forward using VO. We ended up using the initial pose guess provided by the feature-based localization method for both approaches because the direct method was somewhat more sensitive to the initial pose guess than the feature-based method and using the same $\mathbf{T}_{\text{op}}$ facilitates direct comparison of the two approaches. As described in Section 3.3.4, we use a robust cost function to reduce the effect of outliers. We also found it necessary to use a coarse-to-fine pyramid approach to widen the basin of convergence, which is common practice for most direct methods.

During warping, some image coordinates are discarded because they do not project into the new image frame. Hence, the number of image coordinates included in the photometric error computation will vary, which creates local minima in our cost function. Therefore, we warp the image coordinates to perturbed poses in individual DOF in an area around the current pose guess (2.0 cm and 2.9 radians). Then we take the union of image coordinates excluded at each of these perturbations and discard them all from the current step of the optimization. This greatly helps smooth our cost function. Figure 3.3 shows an example of the constant pixel mask we use throughout optimization.

3.4 Experiments

We conducted experiments with data gathered on two real-world outdoor paths driven by the Clearpath Grizzly robot. The camera is a forward-facing factory-calibrated FLIR Bumblebee XB3 stereo camera with 24 cm baseline. Feature-based VT&R with SURF, colour-constant images (Paton et al., 2015), and multi-experience localization (Paton et al., 2016) was used for autonomous navigation. We access images

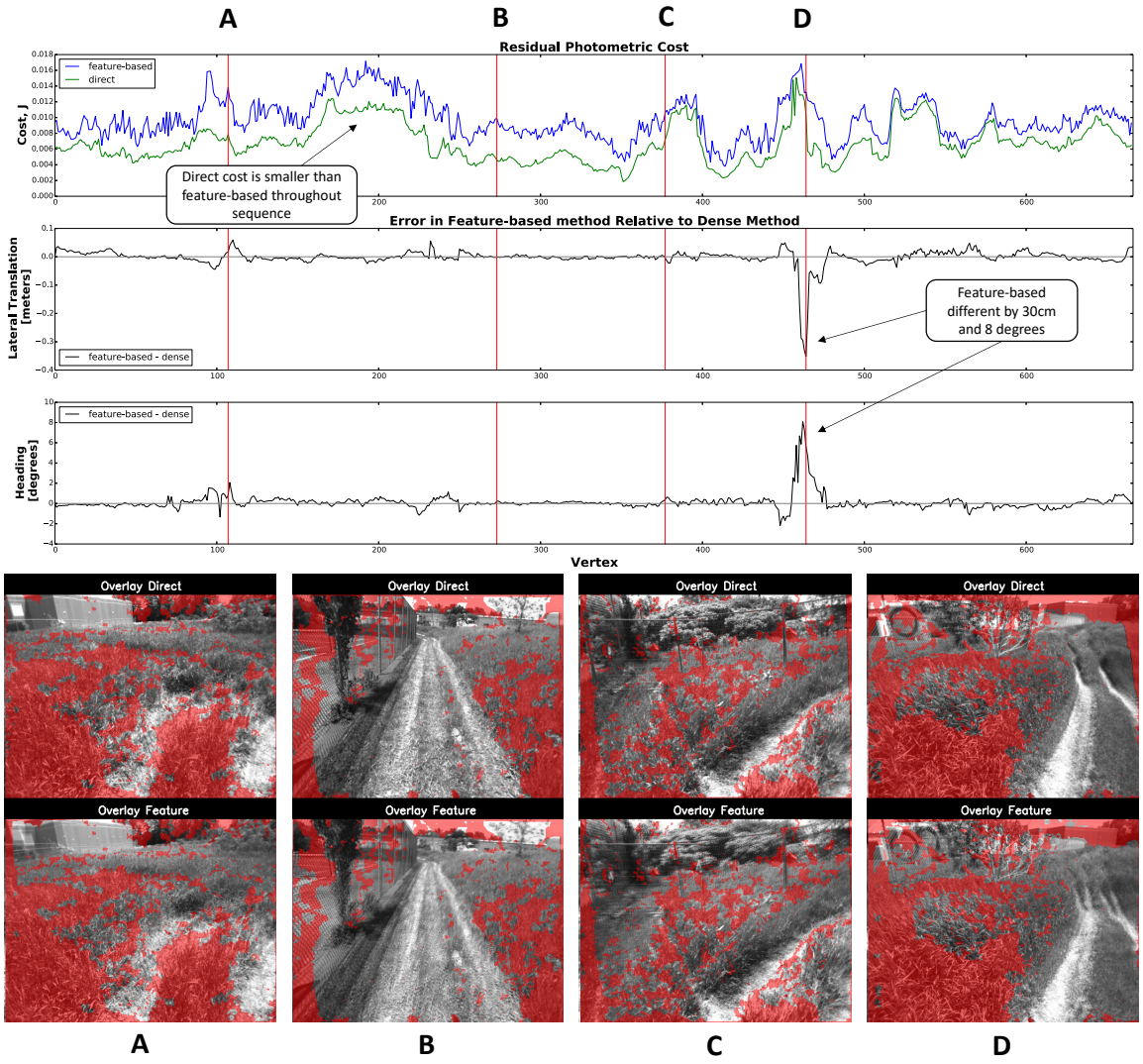


Figure 3.5: The top graph shows the photometric cost between images we are aligning. The images are warped and aligned using the feature-based (blue) and direct (green) pose solutions. The middle and bottom graphs show the difference between feature-based and direct pose for lateral translation and heading. The red lines mark four places along the path for which we display aligned and overlayed images on the bottom row. The red pixels are not used because they could not be warped into the new frame. The large amount of discarded pixels are due to the moving vegetation making stereo matching difficult. These example images show qualitatively that the direct method achieves a better alignment, which is consistent with the lower cost in the top plot.

and poses from the spatio-temporal pose graph created by VT&R during teaching and autonomous path following.

The first path from the UTIAS In-The-Dark dataset (described in detail in Section 2.4.1), is about 250 meters long, covering a paved road and grass in an area with trees, buildings and other structures. Each repeat has significant illumination change with respect to the teach, which constitutes the biggest challenge to localization for this path. The second path is about 165 metres long and runs through an area with undulating terrain and vegetation such as bushes, trees, and a lot of tall grass. The data are collected along the same path as the UTIAS Multiseason dataset (see Section 2.4.2) but was collected

at a different time and is not a part of the dataset. We use three repeat paths captured under similar lighting conditions to the mapped path. The dense vegetation, and especially the grass, poses a challenge for feature-based methods as it provides fewer distinct and stable features. Figure 3.4 displays aerial views of the two paths as well as examples views of the scenes along them.

In the experiments, we localize each vertex on the chosen repeats back to the corresponding vertex on the teach pass. We use the vertex correspondences already given in the pose graph such that we can compare the results between the feature-based method (using SURF) and our method directly. Note that the feature-based method relied on MEL.

3.5 Results

We compare the localization performance of the direct and feature-based methods for the two paths described in the previous section. The comparison is made on dataset and the code is written in Python. The pipeline in its current form does not run in real time, but implementation in C++ in addition to making use of parallelization should be sufficient to make the code run in real-time. There is no GPS pose ground truth available for the dataset we use. Therefore we compare the methods by calculating the photometric error for image alignment. The live image from the current vertex on the repeat run is warped using the localization pose estimated by each of the two approaches, i.e. the relative pose of the live vertex with respect to the corresponding map vertex. Then we take the photometric error between the pixel intensities in the warped image and the image for the corresponding map vertex. The lower the photometric cost, the better the image alignment.

Since we cannot assume ideal brightness constancy between images from different runs, using the photometric error is not a perfectly fail-safe way to compare performance. For this reason, we also provide some anecdotal image overlays between the warped and map images to qualitatively show that lower cost is indeed associated with better image alignment and pose estimation accuracy in our experiments despite the presence of some lighting-change. Furthermore, we show the difference in estimated pose for lateral translation and heading for each vertex along the path. Figure 3.5 shows the results from a repeat on the off-road path, which had little lighting change from teach to repeat, but contained a lot of tall grass and other vegetation. We see that the direct localization provides consistently lower photometric cost than the feature-based method. Across three repeats of this path the average absolute photometric cost difference between the two methods was 0.0032. The average lateral translation difference was 0.010 m and the average heading difference was 0.29 degrees.

Figure 3.6 shows the results from two repeats of the on-road path. The main challenge on this path comes from illumination change that introduces effects such as moving shadows and sun glare. The direct localization generally behaves well, except at certain locations with varying shadows, where we can see a larger discrepancy in the heading and lateral translation differences between the direct and feature-based approaches. In particular, the method tries to align the shadows in the live and mapped images. A basic direct cost function with robust M-estimation is not enough to deal with these intensity changes that violate the brightness constancy assumption (Newcombe et al., 2011) and causes the optimization to converge to local minima.

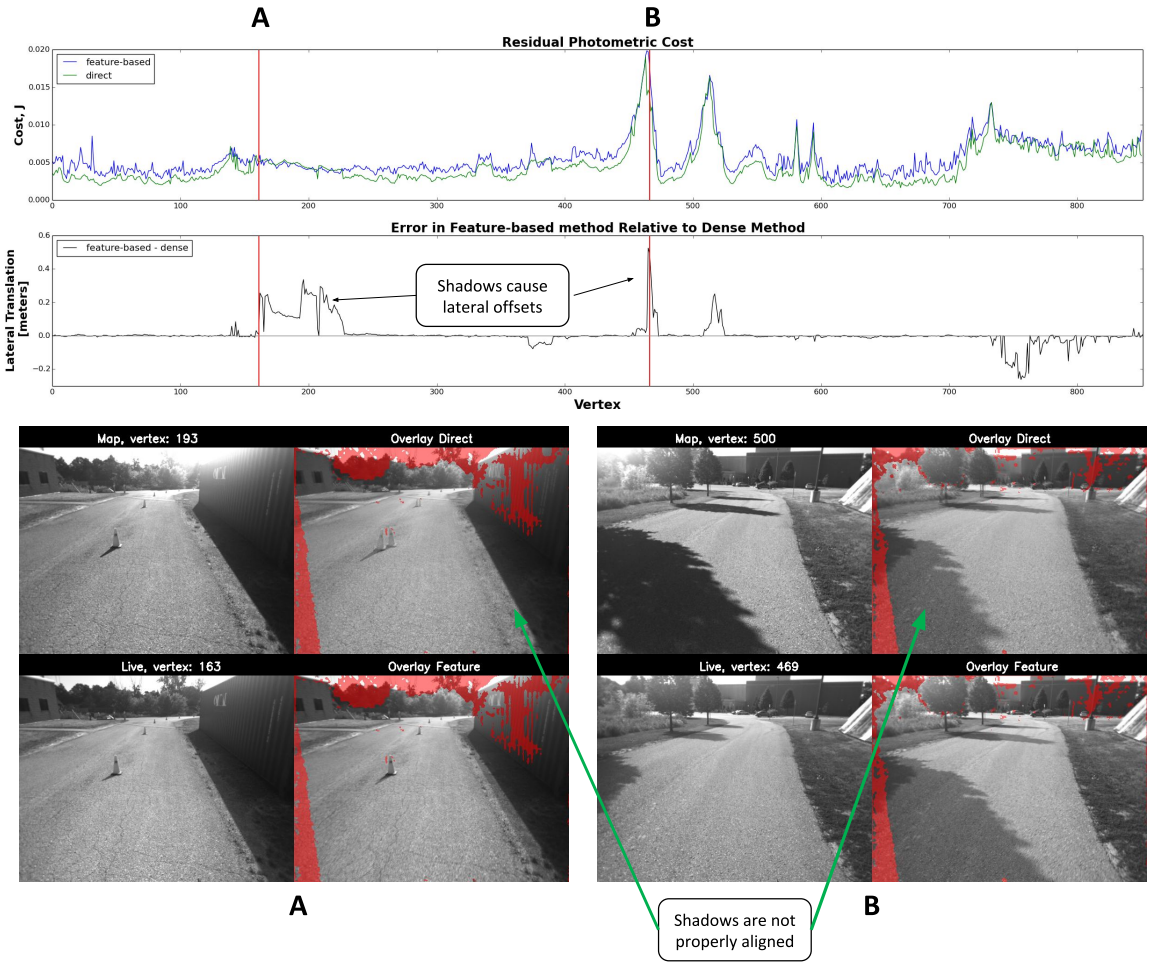


Figure 3.6: The top graph shows the photometric cost between images we are aligning. The images are warped and aligned using the feature-based (blue) and direct (green) pose solutions. The graph below shows the pose difference between the methods for lateral translation. The red lines mark the places for which we display the aligned and overlayed images below. The left column holds the original mapped and live images. These examples shows that the direct method reaches local minima, despite using a robust cost function to downweight outliers. The shaded green areas on the plots show areas, where the direct method struggles due to shadows.

3.6 Conclusions and Future Work

We have implemented a basic pipeline for direct localization in VT&R and compared its performance to feature-based localization. Direct localization provides improved localization accuracy under nominal lighting conditions in a challenging off-road scenario. This shows that we have a correct problem formulation, though some work remains to improve the optimization pipeline, in particular increased robustness to initial conditions, and to illumination changes. For areas with changing shadows, the direct localization converged to local minima. This is a limitation of using the most basic direct localization formulation since the cost function assumes brightness constancy (Newcombe et al., 2011), and adding the robust cost function was not sufficient to solve this problem.

We have seen that tackling environmental changes when using pixel intensities together with a robust cost function is difficult even when we rely on whole-image registration over sparse feature-matching.

This suggests that more work is required on the vision front-end, where we process the images before passing them to a pose optimization algorithm. In the following chapters of this thesis, we shift focus to explore whether deep learning can be used to improve robustness of visual localization to appearance change. We will investigate whether deep learning can help better match images between repeats and the original teach run in single-experience localization.

3.7 Novel Contribution

We include direct localization for stereo images into VT&R and are the first to demonstrate direct localization in a challenging off-road example with tall grass and dense vegetation, but limited appearance change.

3.8 Associated Material

Publication

Gridseth and Barfoot (2019). Towards Direct Localization for Visual Teach and Repeat. In the *2019 Conference on Computer and Robot Vision (CRV)*.

Chapter 4

Deep Learned Pose Estimation for Long-Term Localization

In this chapter we aim to use deep learning to improve robustness of single-experience long-term localization in VT&R by estimating relative poses for localization directly from images using a neural network.

4.1 Introduction

Localization for VT&R that relies on matching of hand-crafted visual features (SURF) has been extended to autonomous operation across lighting, weather, and seasonal change by adding colour-constant images (Paton et al., 2015) and MEL (Paton et al., 2016). As we explained in Section 2.3.4, MEL collects data every time the robot repeats a path, and the most relevant experiences are chosen for feature matching in an effort to bridge the appearance gap. As long as the path is repeated as the environment gradually changes, these bridging experiences will allow for long-term localization. One downside to MEL is that data needs to be gathered for each new path that is taught, even if the new path is driven in a similar environment to an existing path. Moreover, MEL requires keeping track of all the experiences and finding the relevant subset for localization. In this work we wish to compile MEL into a Deep Neural Network (DNN) such that we can perform robust single-experience pose estimation directly from images without the need for feature extraction, feature matching, and pose optimization.

Developing a robust and accurate VT&R system has taken a large research and engineering effort. As a result, we can use outdoor datasets collected with VT&R across lighting and seasonal change to train a DNN for relative pose estimation (see Figure 4.1). As explained in Section 2.3.4, VT&R stores data in a spatio-temporal pose graph that contains the relative poses of repeat vertices with respect to the mapped path for every recorded path traversal. We can sample relative poses between vertices across experiences and use them as ground truth labels to train a DNN. With the DNN, we aim to localize radically different path traversals against each other without the use of intermediate bridging experiences.

The design of the DNN is based on previous work by Melekhov et al. (2017). In particular, our DNN takes two pairs of stereo images and regresses the relative robot pose. We conduct experiments to test the ability of the relative pose regression system to generalize across changing environmental conditions.

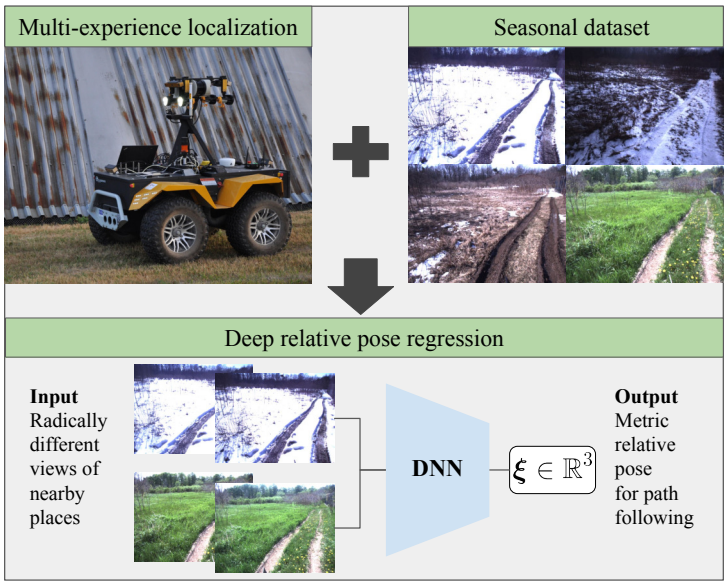


Figure 4.1: We compile MEL for path following into a DNN. We use datasets collected with VT&R across different lighting and seasons to train the DNN to perform three degrees of freedom (DOF) relative pose estimation under changing environmental conditions.

Since both VO and localization with respect to a path fall under relative pose estimation, we test the network's performance on both of these tasks. Using the exact same network architecture, we train one network with VO data and one network with localization data.

To summarize, the novel contributions of this work include estimating poses for localization across large seasonal and lighting change directly from images using a neural network. Moreover, we localize in scenes that include challenging off-road areas with fewer permanent structures, more vegetation, and larger appearance change between seasons than commonly used urban datasets such as Maddern et al. (2017); Sattler et al. (2018); Wenzel et al. (2020). The content in this chapter appeared in the IEEE International Conference on Robotics and Automation (ICRA) (Gridseth and Barfoot, 2020).

4.2 Related Work

CNNs have been used extensively for different parts of visual navigation and control. For instance, researchers have tackled environmental change by using learning as a part of the pose estimation pipeline. Examples include learning robust visual features (Sünderhauf et al., 2015; Arandjelovic et al., 2016) for place recognition, learning which visual features are stable over time and suited for long-term localization (Dymczyk et al., 2016), and transforming whole images to different appearance conditions before using them in direct localization (Clement and Kelly, 2018). Others have in turn focused on replacing the whole pose-estimation pipeline with neural networks by regressing pose directly from images in an end-to-end fashion, several examples of which are presented in a survey on deep learning and visual SLAM (Li et al., 2018a). The two main avenues for this approach is to regress absolute or relative poses.

4.2.1 Absolute Pose Regression

In absolute pose regression, the neural network outputs a global pose in the world frame for the given image input. Early work on absolute pose regression came from the development of PoseNet by Kendall et al. (2015). This system is based on a pre-trained GoogleNet architecture and regresses six DOF pose for metric localization of a monocular camera. Kendall and Cipolla (2016) extended PoseNet to use a Bayesian neural network providing localization uncertainty and an improved loss function (Kendall and Cipolla, 2017). Cai et al. (2018) were also able to generate uncertainties over the pose estimates by using Gaussian process regressors to generate the probability distribution of the camera pose. Naseer and Burgard (2017) improve on PoseNet by generating additional augmented data leading to improved accuracy, while Walch et al. (2017) perform structured dimensionality reduction on the CNN output with the help of Long Short-term Memory (LSTM) units. Clark et al. (2017) and Patel et al. (2018) were able to reduce localization error by passing sequential image data to recurrent models with LSTM units. Melekhov et al. (2018) uses an encoder-decoder together with fully connected layers to regress pose. Wu et al. (2017) reduce the problem of overfitting from sparsely available training poses by synthesizing new poses. They also introduce branching in the latter part of their network to deal with the coupling of orientation and translation. As pointed out by Laskar et al. (2017), papers on absolute pose regression often do not explore generalization to unseen places, because networks are trained separately for each place in a dataset.

4.2.2 Relative Pose Regression

Instead of learning absolute pose, regressing the relative pose between two images removes the reliance on a given scene and opens up the possibility to better tackle generalization. Melekhov et al. (2017) use a Siamese CNN architecture based on AlexNet (Krizhevsky et al., 2012) to compute relative camera pose from a pair of images. Similarly, Bateux et al. (2018) regress relative pose for use in visual servoing, while DeTone et al. (2016) estimate the relative homography between two images. VO is a special case of relative pose estimation and has been explored extensively (Mohanty et al., 2016; Wang et al., 2018; Zhan et al., 2018; Iyer et al., 2018; Costante et al., 2015). In several examples authors combine CNNs with LSTM units to incorporate a sequence of input data (Wang et al., 2018; Iyer et al., 2018). Iyer et al. (2018) use geometric consistency constraints to train their network in a self-supervised manner. Going in a different direction, Peretroukhin et. al. have combined deep learning with traditional pose estimation by learning $SE(3)$ pose corrections (Peretroukhin and Kelly, 2018) and $SO(3)$ rotation (Peretroukhin et al., 2019), which they fuse with relative $SE(3)$ pose estimates.

Relative pose estimation has also been used as a tool to regress absolute pose. In particular, Laskar et al. (2017) combine relative pose regression with image retrieval from a database. Balntas et al. (2018) retrieve nearest neighbours based on learned image features before regressing relative pose to refine the absolute pose. Saha et al. (2018) classify anchor points to which they regress relative pose. Oliveira et al. (2020) combine the outputs of two DNNs for VO and absolute pose estimation, respectively, to accomplish topometric localization. The work was further extended by using multi-task learning for localization (Valada et al., 2018) as well as learning semantics (Radwan et al., 2018).

Authors that make use of relative pose estimation have explored generalization to unseen scenes, but mostly in similar environmental conditions (Laskar et al., 2017; Melekhov et al., 2017; Wang et al., 2018; Mohanty et al., 2016). We go beyond existing work by providing an analysis of the performance of our

relative pose estimation system under drastically different environmental conditions on two challenging outdoor datasets. As explained in Section 2.4, the UTIAS In-The-Dark dataset contains a full day of lighting change, including driving after dark. The UTIAS Multiseason dataset is collected in an off-road environment from winter until spring. We provide pose regression results for VO and localization with respect to a path as well as combining the two.

4.3 Methodology

The neural network outlined in this section can be used to estimate three DOF relative poses for VO and localization in VT&R. In the following, we introduce the neural network architecture and the loss function we use to train it.

4.3.1 System Overview

VT&R stores images as vertices and relative poses between them as edges in a spatio-temporal pose graph. Temporal edges between adjacent vertices are derived from VO, and spatial edges connect vertices from autonomous repeats to vertices on the manually driven teach path. Our system estimates both relative pose for VO as well as metric localization with respect to the mapped path. We use the same neural network architecture and train two networks separately on data for VO and localization, respectively. Since VT&R achieves highly accurate path following (Clement et al., 2017), we sample ground truth relative pose labels for training the DNN from the multi-experience VT&R pose graph. The DNN takes as input RGB stereo images from a pair of vertices and regresses a three DOF relative pose given in the robot frame. For path following, the lateral offset from the path and heading are the most important DOF and so we opt to estimate $\boldsymbol{\xi} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$, where $x \in \mathbb{R}$, $y \in \mathbb{R}$, and $\theta \in [-\pi, \pi]$.

4.3.2 Network Architecture

Our DNN architecture is inspired by the one presented by Melekhov et al. (2017). Similarly to their approach, the convolutional part of our DNN is taken from the well-known AlexNet architecture (Krizhevsky et al., 2012). We do not use a Siamese configuration, but instead opt to input all four RGB images to the network at once, resulting in 12 input channels. Our images are different in size (512×384) from the standard input to AlexNet (224×224), and hence we make use of Spatial Pyramid Pooling (SPP) (He et al., 2015) as in (Melekhov et al., 2017) to reduce the size of our feature map before the fully connected layers. SPP lets us create a fixed-sized output, despite the different input image size, while maintaining spatial information by pooling the responses of each feature in spatial bins (we use max pooling). The size of the output is the number of bins times the number of features. We use four levels of pyramid pooling with the following bins: $5 \times 5, 3 \times 3, 2 \times 2$, and 1×1. Finally, we keep the same fully connected layers as in AlexNet, but add one more fully connected layer with 3 connections to regress the three DOF pose. An overview of the network can be seen in Figure 4.2.

4.3.3 Loss Function

We use a simple quadratic loss function that takes the difference in target and predicted pose coordinates. Translation and rotation are manually weighted using a diagonal matrix $\mathbf{W}$ with 1.0 on the diagonal for

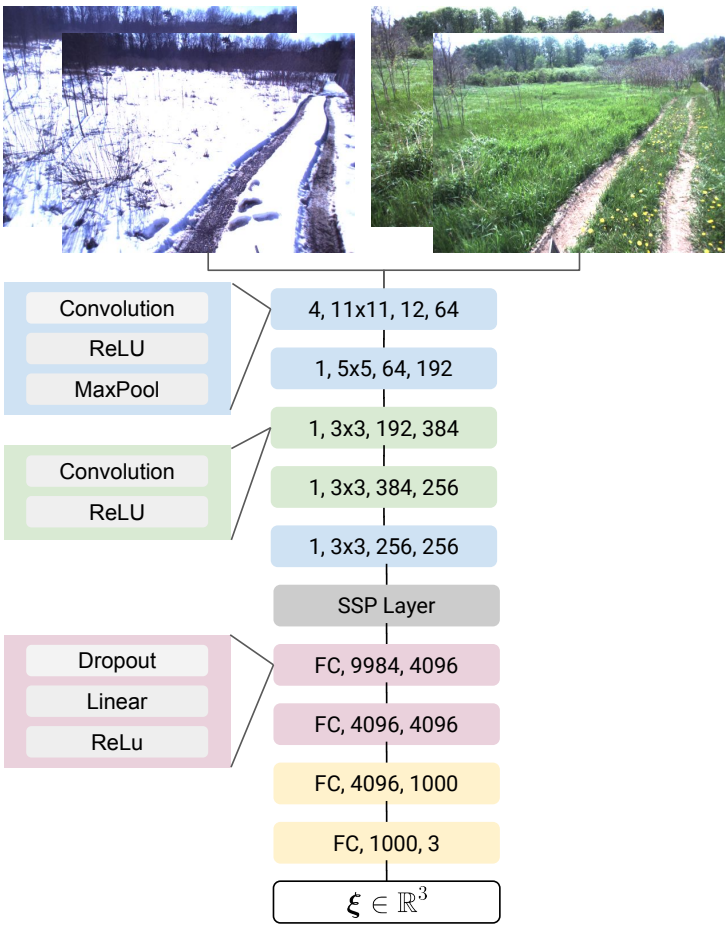


Figure 4.2: The neural network takes two sets of RGB stereo images and produces a three DOF relative pose. The architecture contains convolutional layers, spatial pyramid pooling, and fully connected layers. We list the stride, kernel size, and number of input and output channels for the convolutional layers as well as input and output sizes for the fully connected layers.

x and y and 10.0 for θ. As pointed out by Kendall and Cipolla (2017), angles may wrap around 2π, but this is not a problem we encounter as we are estimating small relative poses. The loss is

$$\mathcal{L} = \frac{1}{2}\left(\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}\right)^T \mathbf{W}\left(\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}\right), \tag{4.1}$$

where $\boldsymbol{\xi}$ represents the target pose that we have sampled from the VT&R pose graph and $\hat{\boldsymbol{\xi}}$ is the estimated pose.

4.4 Experiments

For the experiments, we will train one neural network for VO pose estimation and another for localization. We will test VO on runs under different appearance conditions and test localization across large lighting and seasonal change. The experiments make use of the UTIAS In-The-Dark and UTIAS Multiseason datasets described in detail in Sections 2.4.1 and 2.4.2, respectively. The data were previously collected with the Clearpath Grizzly robot with a factory-calibrated FLIR Bumblebee XB3 stereo camera with

24 cm baseline, see figure 4.1. VT&R with colour-constant images Paton et al. (2015) and MEL Paton et al. (2016) was used for autonomous operation and building the resulting pose graph and ground truth poses.

4.4.1 Data

We train, validate, and test our system on two separate outdoor paths. The UTIAS In-The-Dark dataset (see Section 2.4.1 for more details) covers a path that is about 250 metres long following a paved road and grass in an area with buildings. The path is repeated once per hour for over 30 hours and hence covers significant illumination change. The path has a total of 39 runs from which we choose five for testing and use the remaining for training and validation. The path in the UTIAS Multiseason dataset (see Section 2.4.2) is about 165 metres long. It covers an area with more undulating terrain and vegetation. Data is collected from winter to spring and also includes different weather. The dataset includes a total of 136 runs and we hold out eight of those for testing, while using the rest for training and validation.

As explained in Section 2.4.4, we generate ground truth labels for training and validation for VO by sampling the temporal edges between immediately adjacent vertices in the pose graph. For localization we can sample randomly from the graph in both space and time, allowing us to generate large datasets connecting vertices from both similar and radically different conditions. For this paper, we sample only in time and do not move along the graph in the spatial direction.

For the UTIAS In-The-Dark dataset our training and validation sets have $360,000$ and $40,000$ samples for localization, respectively. For VO we get $64,530$ and 7170 samples. The UTIAS Multiseason dataset has $450,000$ and $50,000$ samples for localization training and validation, respectively. For VO we have $69,659$ training samples and 7739 validation samples.

4.4.2 Training and Inference

We train and test our network once for each path and do not re-train for each individual test condition. The networks are trained separately on the UTIAS Multiseason and UTIAS In-The-Dark datasets, so in total we train four networks (two for VO and two for localization). We on an NVIDIA GTX 1080 Ti GPU with a batch size of 64 and use early stopping based on the validation loss to determine the number of epochs. We use the Adam optimizer (Kingma and Ba, 2015) with learning rate 10^{-5} and other parameters set to their default values. Network inference runs at minimum 50 fps on a ThinkPad P52 laptop with an Intel ® Core™ i7-8850H CPU, 32 GB of RAM, and an NVIDIA Quadro P2000 4 GB GPU.

4.4.3 Visual Odometry and Localization

When running the experiments on the test runs, we keep the data sequential in order to easily assess performance in a realistic scenario. We perform one experiment (per path) for VO, where we compute relative pose for each of the test runs. We test the network's localization ability across environmental change by localizing every repeat in the test set against each other. As mentioned in Section 1.2, the focus of our work is not to improve on the accuracy of VT&R pose estimation, but to tackle localization robustness in long-term operation. VT&R achieves kilometer-scale route repetition with centimeter-level accuracy (Clement et al., 2017) and so we consider the learned pose regression sufficiently accurate for VT&R operation if we stay within 20 cm error in each translational DOF and 5 degrees error in

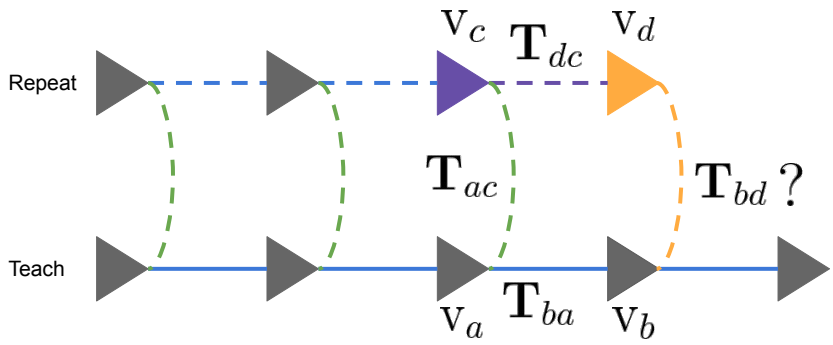


Figure 4.3: We estimate the relative pose for localization for the current repeat vertex, $\mathbf{T}_{bd}$, by using the VO estimate (purple) to propagate the previous localization pose $\mathbf{T}_{ac}$ forward. The propagated pose estimate is combined with the estimated localization pose (orange) from the localization neural network to find the final estimate of $\mathbf{T}_{bd}$.

heading compared to the VT&R ground truth data. We conduct two types of experiments (per path) for localization. In the first experiment we localize the live vertices to the closest map vertices as determined by the VT&R pose graph and compare the results to ground truth.

With the first type of experiment, we only solve one part of the localization problem since the map vertex is already chosen for us by the ground truth system. Therefore, we devise a second test, where we combine the outputs of the VO and localization networks to estimate the pose of the robot with respect to the path. We start with the localization network computing the relative pose between the initial pair of vertices localized by VT&R and then we combine VO and localization as a predictor and corrector as we move forward along the path. We illustrate how this works in Figure 4.3. Assume that we have the current repeat vertex, v_d, and we want to localize it to the closest teach vertex, v_b. We know the relative pose, $\mathbf{T}_{ac}$ from the last repeat vertex, v_c, to its closest vertex on the teach path, v_a, from the latest iteration of localization. From the map, we also know the transform between the teach vertices, $\mathbf{T}_{ba}$. Now, we use the VO network to predict the relative pose between the previous and current live frame, $\mathbf{T}_{dc}$. With the pose from VO, we can propagate the previous localized pose forward to get an initial guess $\hat{\mathbf{T}}_{bd} = \mathbf{T}_{dc}^{-1}\mathbf{T}_{ac}\mathbf{T}_{ba}$. We use the localization network to make its estimate of $\mathbf{T}_{bd}$. Finally, we make a final prediction of the relative pose for localization by taking a weighted combination of the two estimates for $\mathbf{T}_{bd}$. This is not equal to how the predictor/corrector usually works, where the VO prediction is passed as a prior to the localization pose estimation, but it allows us to combine the two neural network predictions in a reasonable way. We continue along the path localizing all the live repeat vertices. The closest map vertex at any time is chosen based on the relative pose computations and not by ground truth from VT&R.

4.5 Results

We conduct experiments with the localization and VO networks for data across large environmental change in an outdoor environment. Tables 4.1 and 4.2 list the Root Mean Squared Error (RMSE) for each repeat in the test sets. The values on the diagonal are results for VO, while the rest are for localization. The rows represent repeat runs while the columns are used as teach runs. We see that the system achieves low errors across a range of conditions. For tests across lighting change, localizing

evening and night repeats are the most challenging, but they do not perform much worse than the other combinations. For the seasonal tests, we see that the network is able to localize runs as different as winter and spring. These examples show the system's potential to localize against large environmental changes directly without relying on intermediate experiences.

Table 4.1: RMSE for each DOF for the UTIAS Multiseason dataset. Translation (x, y) is given in metres and rotation (θ) in degrees. The diagonal entries are VO results, while the off-diagonal entries are localization results. The rows are used as repeats and the columns as teach runs. The green and red cells are better and worse performing examples, respectively, picked for further qualitative analysis.

		Snow		Some snow		No snow		Green	
		Sun	Overcast	Sun	Overcast	Sun	Overcast	Sun	Overcast
Snow	Sun	x : 0.015 y : 0.0039 θ : 0.11	x : 0.073 y : 0.023 θ : 0.54	x : 0.086 y : 0.028 θ : 0.55	x : 0.086 y : 0.041 θ : 0.85	x : 0.070 y : 0.017 θ : 0.40	x : 0.075 y : 0.023 θ : 0.49	x : 0.089 y : 0.028 θ : 0.59	x : 0.082 y : 0.024 θ : 0.49
	Overcast	x : 0.073 y : 0.031 θ : 0.57	x : 0.032 y : 0.0032 θ : 0.11	x : 0.074 y : 0.031 θ : 0.60	x : 0.088 y : 0.046 θ : 0.81	x : 0.074 y : 0.027 θ : 0.55	x : 0.080 y : 0.037 θ : 0.59	x : 0.099 y : 0.040 θ : 0.79	x : 0.088 y : 0.035 θ : 0.68
Some snow	Sun	x : 0.077 y : 0.029 θ : 0.64	x : 0.075 y : 0.032 θ : 0.65	x : 0.014 y : 0.0059 θ : 0.13	x : 0.086 y : 0.037 θ : 0.92	x : 0.070 y : 0.028 θ : 0.61	x : 0.074 y : 0.029 θ : 0.59	x : 0.100 y : 0.039 θ : 0.81	x : 0.091 y : 0.035 θ : 0.70
	Overcast	x : 0.13 y : 0.071 θ : 2.1	x : 0.13 y : 0.069 θ : 2.1	x : 0.12 y : 0.066 θ : 2.4	x : 0.019 y : 0.0025 θ : 0.13	x : 0.12 y : 0.065 θ : 1.3	x : 0.13 y : 0.069 θ : 2.1	x : 0.12 y : 0.071 θ : 1.4	x : 0.13 y : 0.076 θ : 1.6
No snow	Sun	x : 0.056 y : 0.017 θ : 0.39	x : 0.061 y : 0.020 θ : 0.41	x : 0.065 y : 0.023 θ : 0.48	x : 0.079 y : 0.031 θ : 0.63	x : 0.011 y : 0.0033 θ : 0.10	x : 0.057 y : 0.021 θ : 0.42	x : 0.076 y : 0.025 θ : 0.52	x : 0.074 y : 0.019 θ : 0.46
	Overcast	x : 0.071 y : 0.025 θ : 0.49	x : 0.067 y : 0.034 θ : 0.47	x : 0.070 y : 0.028 θ : 0.57	x : 0.082 y : 0.033 θ : 0.75	x : 0.057 y : 0.024 θ : 0.47	x : 0.012 y : 0.0042 θ : 0.012	x : 0.082 y : 0.031 θ : 0.61	x : 0.074 y : 0.028 θ : 0.54
Green	Sun	x : 0.097 y : 0.034 θ : 0.63	x : 0.10 y : 0.039 θ : 0.92	x : 0.10 y : 0.042 θ : 0.81	x : 0.095 y : 0.045 θ : 0.96	x : 0.088 y : 0.031 θ : 0.66	x : 0.097 y : 0.036 θ : 0.74	x : 0.019 y : 0.0033 θ : 0.14	x : 0.070 y : 0.029 θ : 0.50
	Overcast	x : 0.090 y : 0.029 θ : 0.55	x : 0.099 y : 0.032 θ : 0.69	x : 0.10 y : 0.033 θ : 0.65	x : 0.10 y : 0.042 θ : 0.94	x : 0.10 y : 0.026 θ : 0.52	x : 0.097 y : 0.030 θ : 0.61	x : 0.089 y : 0.030 θ : 0.52	x : 0.013 y : 0.0035 θ : 0.14

To supplement our quantitative findings, we provide detailed plots for two test cases from each dataset. We pick one of the best performing test cases (marked in green in the tables) and one of the cases with the largest errors (marked in red). We integrate the results from VO to show the full paths in Figures 4.5 and 4.4. Figures 4.7 and 4.6 display localization results. For path following, the most important performance indicators are the lateral and heading errors with respect to the path. For a small segment of each path, we plot the ground truth y and θ values against those predicted by the localization network (for the ground truth vertex pairs) as well as the full localization method that combines VO and localization network outputs. We think the latter method has a smoothing effect on the pose estimates. The fact that this method chooses different map vertices for localization than the ground truth may account for some of the discrepancy between the two solutions in Figure 4.6. Additionally, in Figures 4.7 and 4.6, we plot the cumulative distribution of y and θ errors for the whole test sequence and include two example teach-and-repeat image pairs, illustrating the challenging environmental change. The path from the UTIAS In-The-Dark Dataset has less sharp turns and small lateral path offsets resulting in a smaller signal-to-noise ratio in the data. This makes the value of y harder to predict, see Figure 4.7. Given the qualitative plots from the example runs and the fact that the RMSE compared to the VT&R ground truth are less than 20 cm in translation and 5 degrees in rotation (as suggested in Section 4.4.3), we believe that the learned poses would be sufficiently accurate for successful path following in the loop.

Table 4.2: RMSE for each DOF for the UTIAS In-The-Dark dataset. Translation (x, y) is given in metres and rotation (θ) in degrees. The diagonal entries are VO results, while the off-diagonal entries are localization results. The rows are used as repeats and the columns as teach runs. The green and red cells are better and worse performing examples, respectively, picked for further qualitative analysis.

	Morning	Sun Flare	Day	Evening	Night
Morning	x : 0.0073 y : 0.0022 θ : 0.080	x : 0.012 y : 0.0053 θ : 0.17	x : 0.013 y : 0.0058 θ : 0.15	x : 0.013 y : 0.0079 θ : 0.15	x : 0.013 y : 0.0093 θ : 0.21
Sun Flare	x : 0.010 y : 0.0051 θ : 0.12	x : 0.0079 y : 0.0023 θ : 0.084	x : 0.011 y : 0.0060 θ : 0.14	x : 0.013 y : 0.0069 θ : 0.15	x : 0.013 y : 0.0086 θ : 0.20
Day	x : 0.012 y : 0.0058 θ : 0.13	x : 0.011 y : 0.0058 θ : 0.14	x : 0.0074 y : 0.0021 θ : 0.079	x : 0.013 y : 0.0069 θ : 0.15	x : 0.013 y : 0.0087 θ : 0.20
Evening	x : 0.019 y : 0.011 θ : 0.22	x : 0.019 y : 0.011 θ : 0.22	x : 0.019 y : 0.011 θ : 0.22	x : 0.0091 y : 0.0037 θ : 0.092	x : 0.020 y : 0.012 θ : 0.28
Night	x : 0.015 y : 0.013 θ : 0.29	x : 0.015 y : 0.014 θ : 0.31	x : 0.016 y : 0.013 θ : 0.29	x : 0.016 y : 0.014 θ : 0.31	x : 0.0047 y : 0.0041 θ : 0.092

4.6 Conclusions and Future Work

In this chapter, we presented a DNN that can perform relative pose regression for both VO and localization for paths in an outdoor environment across illumination and seasonal change. We sampled ground truth labels for training and testing from a spatio-temporal pose graph generated by VT&R. We conducted experiments across environmental change on data from two outdoor paths. The network carries out VO under different and challenging conditions, including after dark. Furthermore, our network can perform localization for input image pairs from different times of day or seasons without the need for intermediate bridging experiences, which are necessary for long-term operation with the original VT&R system. From the performance we achieve on these datasets we believe that the localization system is sufficiently accurate for in-the-loop path following.

An end-to-end trained neural network that regresses pose directly from images provides a very simple method for localization since visual feature extraction, data association, 3D point computation, pose optimization and other tasks in a typical pose estimation pipeline are not needed. Moreover, we see that using deep learning has helped us tackle large appearance change that has not been successful with traditional handcrafted features or the direct approach we presented in Chapter 3. With the learned approach, however, we found it to be challenging to get the neural network to generalize well to other paths not seen in the training data. Our aim in the next two chapters is therefore to introduce more structure into our deep learning approach in order to better tackle generalization to new paths

4.7 Novel Contributions

- We estimate relative poses for localization across large seasonal and lighting change directly from images using a neural network. We use the same network architecture to estimate relative poses for VO and combine VO and localization for path following.

- We are the first to tackle large seasonal appearance change in a challenging off-road environment

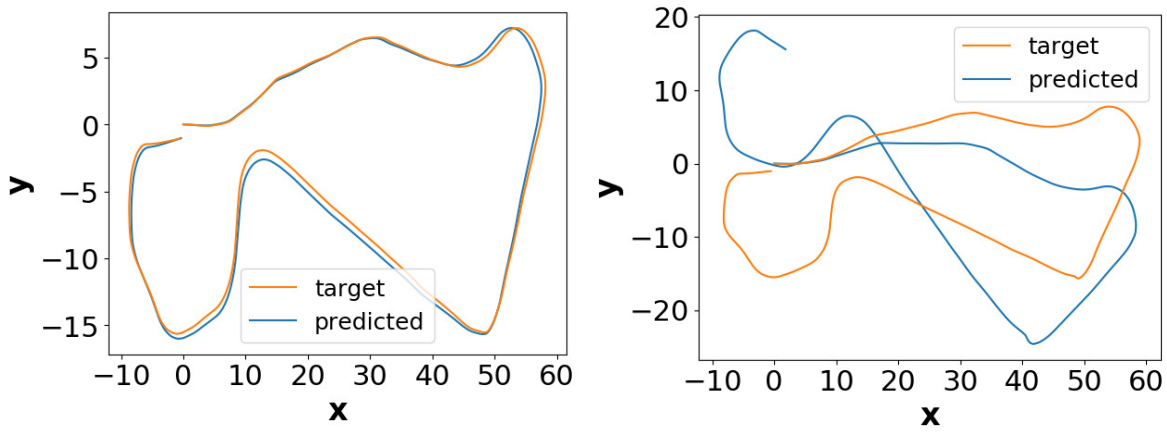


Figure 4.4: Integrated VO for sunny weather with snow on the ground and overcast weather with no snow representing one of the most and least accurate results, respectively.

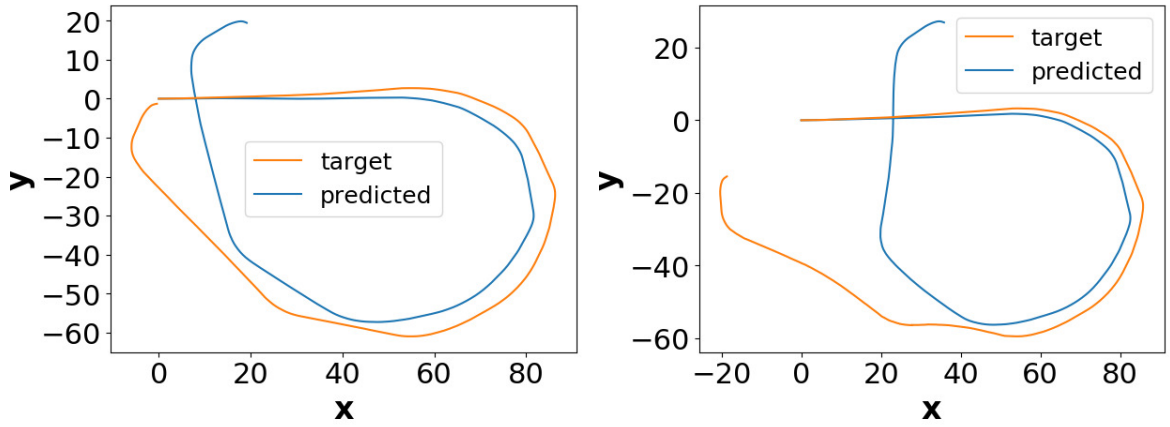


Figure 4.5: Integrated VO for day and evening with low sun and sun flares. These examples represent one of the most and least accurate results, respectively.

with deep learned pose regression. We localize in scenes with fewer permanent structures, more vegetation, and larger seasonal appearance change (such as snow and green grass) than commonly used urban datasets.

4.8 Associated Material

Publication

Gridseth and Barfoot (2020). DeepMEL: Compiling visual multi-experience localization into a deep neural network. In the *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

Videos

- Deep learned pose estimation: https://youtu.be/lXbTaAM6kSY
- Deep learned pose estimation (conference presentation): https://youtu.be/XrlLDTFiMt8

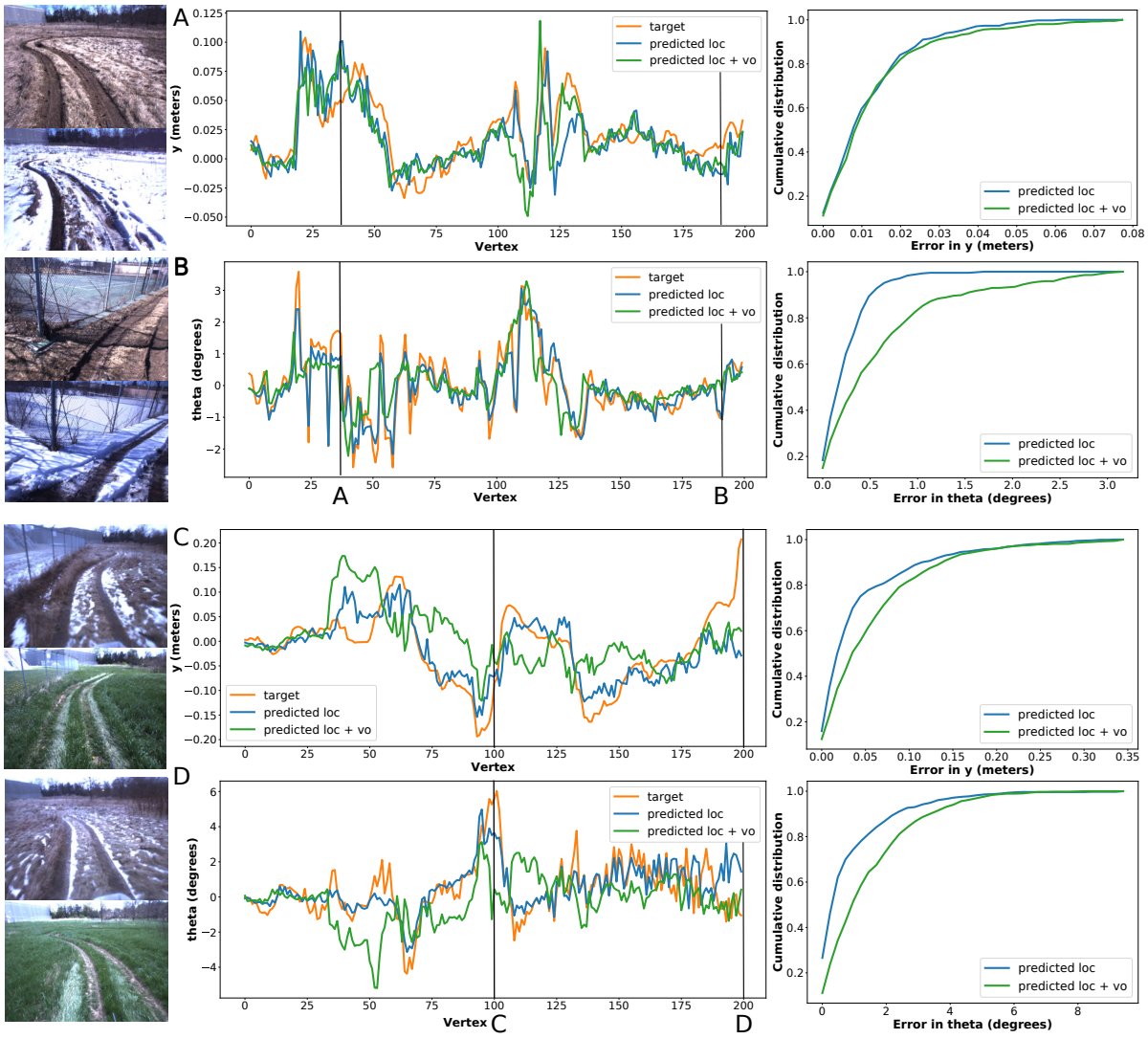


Figure 4.6: The figure shows localization results for one of the better (top) test sequences from the UTIAS Multiseason dataset and one with larger errors (bottom). We plot the relative pose estimates for y and θ from the localization network (blue) as well as pose estimates from combining VO and localization (green) together with the target values (orange) for a segment of the full path. The plots on the right show the cumulative distribution of errors for the full path. The image pairs on the left provide anecdotal examples from the test sequence and are marked in the plot.

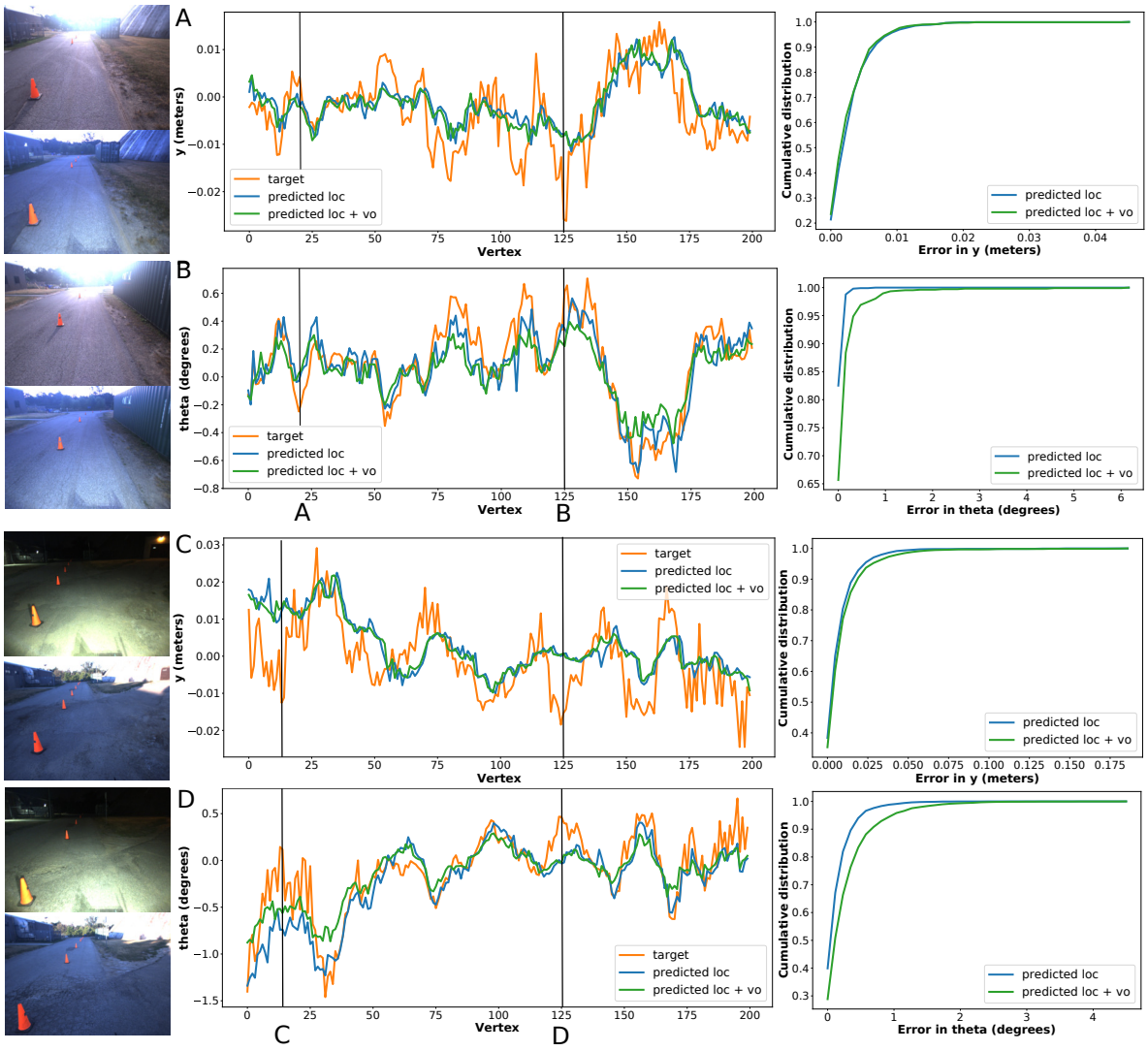


Figure 4.7: The figure shows localization results for one of the better (top) test sequences from the UTIAS In-The-Dark dataset and one with larger errors (bottom). We plot the relative pose estimates for y and θ from the localization network (blue) as well as pose estimates from combining VO and localization (green) together with the target values (orange) for a segment of the full path. The plots on the right show the cumulative distribution of errors for the full path. The image pairs on the left provide anecdotal examples from the test sequence and are marked in the plot.

Chapter 5

Deep Learned Features for Visual Odometry

5.1 Introduction

In Chapter 4, we successfully trained a network to learn relative pose for VO and localization across large appearance change directly from sensor inputs. We found, however, that the approach struggled to generalize to new paths not seen during training. Sattler et al. (2019) state that accuracy can be an issue for methods that regress global pose directly from sensor data. In their survey, Chen et al. (2020) find that hybrid methods, which combine deep learning and classical geometric models, are usually more successful for VO than models that learn pose directly, with some even outperforming state-of-the-art classical monocular VO algorithms (Yang et al., 2020).

This motivates introducing more structure to our method for learning relative poses. Instead of learning a replacement for a complete VO or localization pipeline, we focus on learning visual features, while using a traditional algorithm for pose estimation, see Figure 5.1. In this chapter, we apply this approach to VO before continuing to work with localization in Chapter 6. Due to the lack of major lighting or environmental change between image frames, VO is a simpler problem than localization to start with. Our work follows the approach of Barnes and Posner (2020), which learns features for RADAR localization. In particular, we learn keypoints with associated descriptors and scores that are passed to a differentiable point-based pose estimator. This ensures the pipeline can be trained end-to-end supervised by ground truth poses. Learning the score is supervised solely from pose error and so determines which keypoints are robust for VO, serving as outlier rejection. We only need the ground truth poses to construct our loss functions, and therefore we do not require access to additional ground truth keypoint correspondences.

In this work, we aim to use deep learning for stereo VO on challenging outdoor data with vegetation and few permanent structures. The data cover different lighting and seasonal conditions, including challenging scenarios such as snow cover, tall grass, driving after dark, and strong sun flares. Furthermore, we test the learned feature's ability to generalize to paths not seen in the training data. We compare our experimental results to a baseline using handcrafted SURF (Bay et al., 2006). We find that properly weighting keypoints during pose estimation is essential to successful VO and therefore also experiment with learned and non-learned weights for both the learned and baseline methods.

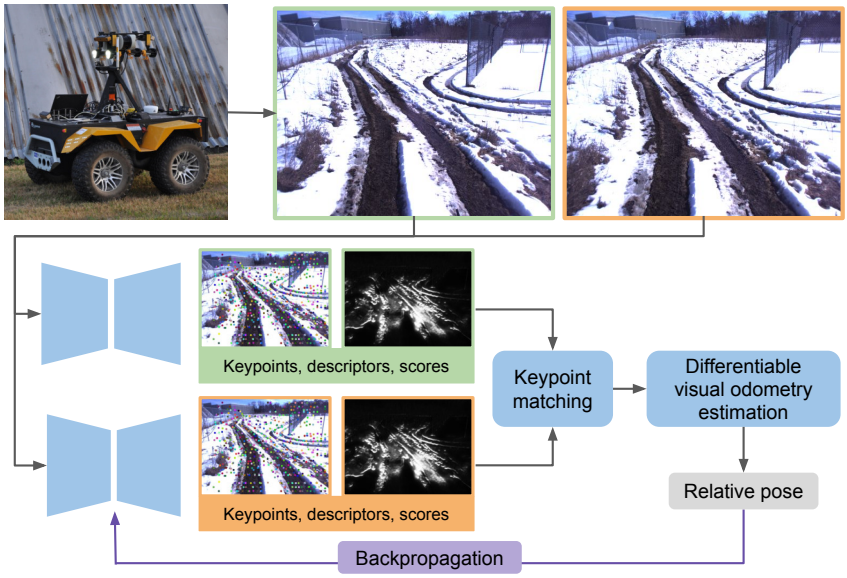


Figure 5.1: We use a neural network to predict keypoints, descriptors, and keypoint scores that can be used in classical VO pose estimation. The differentiable pipeline lets us train end-to-end from ground truth poses. We train and test on challenging outdoor data collected with a Clearpath Grizzly ground robot.

To summarize, we develop an approach to learn sparse visual features for relative pose estimation based on the earlier work on RADAR localization from Barnes and Posner (2020). We successfully apply it to VO and show that the features generalize to new paths. Other than working with off-road environments and a large variety of seasonal conditions, the successful estimation of poses for VO is in itself not a novel contribution since applications of learned features to VO already exist (DeTone et al., 2018a; Tang et al., 2020; Jau et al., 2020). The main goal of this chapter is to develop a learning pipeline that we apply to the more challenging localization problem in Chapter 6. Given the improvement in generalization over Chapter 4, we find the VO results beneficial to include in the thesis as a step on the way towards learned features for localization.

5.2 Related Work

There has been a wide range of work on deep learning for pose estimation. Chen et al. (2020) provide a survey of deep learning for mapping and localization, including VO. Both supervised and unsupervised techniques have been used to learn pose directly from sensor data for VO (Wang et al., 2018; Li et al., 2018b; Mahjourian et al., 2018). As mentioned before, Sattler et al. (2019) noted that learning pose directly from image data can struggle with accuracy, and Chen et al. (2020) found that approaches combining deep learning and classical geometric models for VO usually improve over the latter. Structure can be added to deep learning in different ways. While still learning pose directly, some researchers have used the properties of rigid motion to add loss terms that help enforce the geometric consistency of their computed trajectory (Iyer et al., 2018; Wang et al., 2019). Another approach focuses on learning corrections to poses computed by a classical VO method (Peretroukhin and Kelly, 2018; Wagstaff et al., 2020).

5.2.1 Learned Features

A further possibility for adding more structure to learned pose estimation, is to focus on learning visual features, while retaining a classical pose estimator. This tackles challenging visual appearance with learning, but avoids using learning for the part of the problem with well-known solutions. A lot of work has been done on the various components relating to visual features, from learning detectors, descriptors, weights, matching, and outlier rejection. Many papers focus on one of these parts exclusively, while others tackle several at once. Due to the amount of previous work, we choose to highlight those papers most relevant to the tasks of VO and localization, that we are interested in. Ma et al. (2021) provide an excellent and thorough survey of feature detection, description, and matching covering both handcrafted and learned features, which we recommend consulting for further detail.

We have created a table to compare a select set of methods most relevant to our work, see Table 5.1. This means that we mostly focus on methods that learn detection, description, and matching together. Moreover, we pay most attention to papers that train features targeted specifically at the tasks of either VO or metric localization. We do not consider image retrieval or place recognition. Although this chapter develops learned features for VO, we continue with localization in Chapter 6 and so set up our comparison table with both tasks in mind. For sparse feature-based localization, specifically, we concentrate on what Chen et al. (2020) refer to as 2D-to-3D localization or close variants thereof. In 2D-to-3D localization, correspondences are established between 2D points in an image and 3D points in a map, before performing outlier rejection with RANSAC and pose estimation with a Perspective-n-Point (PnP) solver. We choose to leave out localization with scene coordinate regression, which Chen et al. (2020) describe as estimating the 3D coordinates of each pixel in an image in the world coordinate system or learning a transform from an image to the global coordinates of the scene. Finally, we will also leave out feature learning for 3D-to-3D localization with LiDAR sensors. A comprehensive review of the aforementioned topics are included in Chen et al. (2020).

We start by getting an overview of a breadth of feature-learning strategies before discussing learned features applied specifically to VO (Section 5.2.2) and localization (Section 6.2.1). In particular, we look at what the methods learn and the network architectures they use, what data or ground truth labels they rely on for training, and, finally, how they test performance.

Method Design

In the second to fourth columns of Table 5.1, we have recorded which components out of keypoint detection, descriptor computation, and feature matching, that each approach learns. As documented by Chen et al. (2020) and Ma et al. (2021), many methods focus solely on a subset of these. We only include a few examples of such approaches in the first rows of Table 5.1, because we are more interested in methods that learn all three simultaneously in a differentiable pipeline (with exception of the lack of keypoint detection for direct methods).

LIFT (Yi et al., 2016) and LF-Net (Ono et al., 2018) are two of the early methods that were able to jointly train keypoint detection, description, and matching. For LF-Net, only one of the two branches of their pipeline is differentiable, which complicates training. Moreover, both methods operate on image patches instead of whole images. SuperPoint (DeTone et al., 2018b), on the other hand, computes keypoints and descriptors for whole images. The authors design a network with an encoder and two decoders, one for descriptors and the other for keypoint detection. Several other methods rely on

Table 5.1: The table compares different methods that use learned features. We compare whether they learn keypoint detection, descriptors, or matching. We distinguish between sparse and dense methods. Methods that compute densely across the image but use sparse keypoints and descriptors at test time, are listed as sparse. We record what ground truth data is used for supervision, i.e. are only poses needed or does the method need explicit point correspondence labels (an empty response indicates unsupervised). We note which task the method has been tested on such as feature matching, VO, etc. Tasks marked with * were tested on indoor data. We check whether the has been tested on trajectory data as this is most relevant for robot navigation. Finally, we indicate if the test data contains lighting or seasonal change.

Method	Detect	Describe	Match	Sparse	Supervise	Task	Trajectory	Lighting	Seasons
TILDE (Verdie et al., 2015)	✓	✗	✗	✓	Points	Keypt. detection	✗	✓	✓
(Wang et al., 2020)	✗	✓	✓	✓	Pose	Localization	✗	✓	✗
SuperGlue (Sarlin et al., 2020)	✗	✗	✓	✓	Matched points	Localization	✗	✓	✗
(Yu et al., 2020)	✗	✗	✓	✓	Matched points	Localization	✓	✓	✓
LIFT (Yi et al., 2016)	✓	✓	✓	✓	Matched patches	Feat. matching	✗	✓	✗
LF-Net (Ono et al., 2018)	✓	✓	✓	✓	Pose, depth	Feat. matching	✗	✓	✗
SuperPoint (DeTone et al., 2018b)	✓	✓	✓	✓	Synthetic points	Homography est.	✗	✓	✗
UnSuperPoint (Christiansen et al., 2019)	✓	✓	✓	✓	-	Homography est.	✗	✓	✗
(Venator et al., 2021)	✓	✓	✓	✓	-	Localization	✓	✓	✓
D2-Net (Dusmanu et al., 2019)	✓	✓	✓	✓	Matched points	Localization	✗	✓	✗
R2D2 (Revaud et al., 2019)	✓	✓	✓	✓	Dense matched points	Localization	✗	✓	✗
ASLFeat (Luo et al., 2020)	✓	✓	✓	✓	Dense matched points	Localization, VO	✓	✓	✗
S2DNet (Germain et al., 2020)	✗	✓	✓	✓	Matched points	Localization	✓	✓	✓
SAND (Spencer et al., 2019)	✗	✓	✓	✓	Matched points	SLAM	✓	✗	✗
(Spencer et al., 2020)	✗	✓	✓	✓	Matched image triplets	Feat. matching	✓	✓	✓
(Lv et al., 2019)	✗	✓	✓	✗	Pose	SLAM*, VO*	✓	✓	✗
BA-Net(Tang and Tan, 2019)	✗	✓	✓	✗	Pose, depth	Localization*, VO	✓	✗	✗
(Xu et al., 2020)	✗	✓	✓	✗	Pose	SLAM*	✓	✓	✗
(von Stumberg et al., 2020a)	✗	✓	✓	✗	Matched points	Localization	✓	✓	✓
(von Stumberg et al., 2020b)	✗	✓	✓	✗	Pose, matched points	Localization	✓	✓	✓
(Kasper et al., 2020)	✗	✓	✓	✗	VO poses	Localization	✓	✓	✓
(Germain et al., 2021)	✗	✓	✓	✓	Pose, (3D point cloud)	Localization	✗	✓	✗
(Bhowmik et al., 2020)	✓	✓	✓	✓	Pose	SfM	✗	✓	✗
(Barnes and Posner, 2020)	✓	✓	✓	✓	Pose	Odom. + localize	✓	✓	✗
(Sarlin et al., 2021)	✗	✓	✓	✓	Pose, (3D point cloud)	Localization	✓	✓	✓
(Tang et al., 2020)	✓	✓	✓	✓	-	VO	✓	✗	✗
(DeTone et al., 2018a)	✓	✓	✓	✓	-	VO*	✓	✗	✗
(Jau et al., 2020)	✓	✓	✓	✓	Synthetic points, pose	VO	✓	✗	✗
(Huang et al., 2020)	✓	✓	✓	✓	Not trained	VO*	✓	✓	✗
(Li et al., 2020)	✓	✓	✓	✓	Not trained	SLAM*	✓	✓	✗
HF-Net (Sarlin et al., 2019)	✓	✓	✓	✓	Teacher networks	Localization	✓	✓	✓
(Gladkova et al., 2021)	✓	✓	✓	✓	Not trained	VO + localization	✓	✓	✓
(Sun et al., 2021)	✓	✓	✓	✓	-	VT&R closed-loop	✓	✓	✗

the SuperPoint network design (Christiansen et al. (2019) (UnSuperPoint), Venator et al. (2021), and Bhowmik et al. (2020)). D2-Net (Dusmanu et al., 2019) chooses a different strategy of combining the weights for detection and description in one encoder by basing detection on local maxima in descriptor space. The downside is that detection happens at a lower image resolution, which limits accuracy. ASLFeat (Luo et al., 2020) relies on the design from D2-Net but remedies the resolution limitation by detecting keypoints at multiple levels of the network corresponding to multiple image resolutions. This is similar to the well-known approach of using Gaussian image pyramids for hand-crafted feature extraction. While S2DNet (Germain et al., 2020) uses an off-the-shelf keypoint detector, they employ a similar strategy to ASLFeat by matching descriptors at multiple levels of the encoder. Differently from most other methods, they match sparse descriptors in the source image to dense descriptors in the target image. R2D2 (Revaud et al., 2019) learns descriptors together with point repeatability and reliability.

They avoid the issue of detection at low resolution by using dilated convolutions to maintain the spatial resolution. Sarlin et al. (2021) compute dense descriptors at several levels, while Germain et al. (2021) compute coarse and fine descriptors with two different networks.

Barnes and Posner (2020) operate on RADAR data, but we include their approach because we build on it for our own work. The authors use an architecture with one encoder together with one decoder-branch for keypoint detection and another to compute point importance scores. They extract descriptors densely from each level of the encoder before resizing and concatenating them together. One advantage to the approach from Barnes and Posner (2020), is that they directly detect a fixed number of keypoints across the image without the use of Non-Maximal Suppression (NMS) or other strategies for picking the top keypoint candidates. SAND (Spencer et al., 2019) and Spencer et al. (2020) extract dense descriptors from the decoder in a single encoder-decoder pair. Additionally, SAND uses an off-the-shelf keypoint detector at inference for tasks that require keypoints.

Finally, we look at dense learned features for use in direct pose estimation. As explained in Section 3.3.5, direct methods do not detect and match sparse keypoints, but instead they iteratively estimate pose for different resolutions of an image pyramid. Therefore, the learned methods from Lv et al. (2019), Tang and Tan (2019) (BA-Net), Xu et al. (2020), von Stumberg et al. (2020a) (GN-Net), von Stumberg et al. (2020b) (LM-Reloc), and Kasper et al. (2020), extract descriptors densely from each level of their encoder or decoder and use these features to estimate pose at the corresponding image resolution.

Training

Next, we compare what ground truth labels the different methods use for training. This gives insight into how easy a method is to develop, especially in the case of robotics, where real-world data is time-consuming to collect and label. The column labeled "Supervise" in Table 5.1 lists the ground truth labels required for training supervision. Many approaches for learning features rely on point or point correspondences between two images (D2-Net, R2D2, ASLFeat, S2DNet, SAND, GN-Net, and LM-Reloc), which can be tedious to generate at scale. They can be computed using ground truth poses and depth, though might rely on an existing keypoint detector. Alternatively, point correspondences can be generated with classical pose estimation techniques, though they may run into challenges with appearance changes, which is one of the problems learning is trying to mitigate. Spencer et al. (2020) use image triplets with a positive sample from the same location and a negative sample for a different place. These are easier to generate than point correspondences, but still require the capacity for place recognition.

SuperPoint gets around the need for point correspondences by bootstrapping keypoint detection with synthetic data. In the second training stage, they apply the keypoint detector and use homographic adaptation to train their full network. This means that they warp images with known homographies and match descriptors between warped images. The downside is that homographies do not capture full six DOF pose changes. Moreover, they are not able to compare images taken under naturally varying appearance change. UnSuperPoint trains fully unsupervised by removing the need for initial bootstrapping but still relies on homography adaptation. Venator et al. (2021) further compensate for the appearance problem by adding learned domain adaptation, although it does not capture full real-world appearance change.

Another group of learning methods use the ground truth pose for training (LF-Net, Germain et al. (2021), and Wang et al. (2020)). For instance, Wang et al. (2020) develop a loss function that exploits

the epipolar constraint given by relative camera poses. Training networks with relative poses from VO or localization, may make the learned features better adapted to these tasks than if trained with homographies or general point correspondences. Going further than feature matching on relevant data, some approaches use a training pipeline that includes differentiable pose estimation (Lv et al. (2019), BA-Net, Xu et al. (2020), Barnes and Posner (2020), Bhowmik et al. (2020), and Sarlin et al. (2021)). This allows them to train features end-to-end explicitly for the target pose estimation task. Direct methods have the advantage that they do not rely on non-differentiable feature matching. They, however, use iterative optimization, which can be made differentiable by unrolling the optimizer a fixed number of steps as in Lv et al. (2019), BA-Net, Xu et al. (2020), and Kasper et al. (2020). Although Kasper et al. (2020) train features for localization, they are able to supervise with poses from VO because they consider the relative poses between three frames, two of which are temporally adjacent. Bhowmik et al. (2020) make their sparse feature pipeline differentiable by implementing probabilistic feature matching with the help of reinforcement learning. Sarlin et al. (2021) create a differentiable pipeline by aligning dense features and computing pose at multiple levels. Their method, however, relies on a 3D point cloud and course pose estimate as inputs. Finally, Barnes and Posner (2020) make feature matching differentiable by matching sparse descriptors in the source image to dense descriptors in the target and computing the weighted target coordinates. Moreover, they solve for the pose between two sets of 3D points with SVD (see Section 2.2.3), which avoids iteration and is also differentiable.

Testing

Finally, we look how the learned feature methods are tested and what data they use for testing. The column labeled "Task" in Table 5.1 specifies which task is used to test the features. If several tasks are used, we list the one most relevant to VO and localization, which we are interested in. In the column labeled "Trajectory", we indicated whether the features were tested on data from a trajectory. In their paper on benchmarking outdoor visual localization, Sattler et al. (2018) distinguish between datasets, where images are part of a trajectory or are taken "free viewpoints". The former is most often collected by a robot or car driving on the road, while the latter, for instance, represent images of a building taken from different viewpoints. Since our work ultimately revolves around path following, we are most interested in seeing how various learned features perform on a similar task, i.e. on datasets that contain trajectories preferably collected outdoors. Images of large buildings are less relevant as they contain large planes, have many corners and other structured features, and provide less useful viewpoints (looking up at a building instead of along a path). As the last two columns of our table indicate, we also record which features have been tested under illumination and seasonal change.

We see in Table 5.1 that some of the approaches we have discussed above are only tested on feature matching (LIFT, LF-Net, and Spencer et al. (2020)). Moreover, SuperPoint and UnSuperPoint, which were trained with homography adaptation, also test on homography estimation. Venator et al. (2021), which extended SuperPoint with domain adaptation, tests localization on trajectory data across lighting and seasonal change. Some methods test on localization and appearance change, but do not include data with trajectories (R2D2, D2-Net, Germain et al. (2021), and Bhowmik et al. (2020)). Finally, several of the methods we have discussed so far test their learned feature on tasks such as VO, SLAM, or localization on datasets containing trajectories (ASLFeat, SAND, S2DNet, Sarlin et al. (2021), GN-Net, LM-Reloc, Kasper et al. (2020), BA-Net, Lv et al. (2019), and Xu et al. (2020)). At this point, we do not go into more detail on the experiments, but leave this to the following section, where we discuss

learned features for VO and to Section 6.2.1, where we discuss localization.

5.2.2 Visual Odometry

Learned features have been applied specifically to the task of VO, either by learning new features or using existing ones for VO. DeTone et al. (2018a) use the SuperPoint network as a frontend to produce visual features that are passed to a backend that uses Bundle Adjustment and computes 3D points, which are in turn used as training signals for the frontend network. This means that they do not need any external ground truth data for training. The VO system is tested on the indoor ScanNet dataset (Dai et al., 2017), where they show improvement over classical and learned approaches they compare to. Tang et al. (2020) learn both keypoint detection and depth estimation unsupervised for VO. They introduce differentiable pose estimation to their pipeline by solving for an initial pose with a PnP algorithm and RANSAC before correcting the pose with a differentiable method. They integrate their learned features into the existing Direct Sparse Odometry framework (Engel et al., 2018) and test on outdoor VO data from the KITTI dataset (Geiger et al., 2012), where they improve over both classical and learned methods they compare with. Jau et al. (2020) use the SuperPoint architecture when learning keypoint detection and description for VO. The authors use a second network to compute weights for matched point pairs and rely on SVD for relative pose estimation, which makes their pipeline differentiable. They first train the feature network in the same way as described for SuperPoint, before training the point weighting network. At the end they put all the components together and train the full pipeline end-to-end with a pose loss. Jau et al. (2020) test their features on VO sequences from the KITTI dataset and show results on par with classical methods.

The final two papers we include, do not train features but use existing learned features in VO and SLAM pipelines, respectively. Huang et al. (2020) integrate SuperPoint features in a monocular VO pipeline and show improvement over the classical Direct Sparse Odometry (DSO) and ORB-SLAM (Mur-Artal et al., 2015) under challenging conditions from the synthetic indoor New Tsukuba (Martull et al., 2012) dataset, and the indoor EuRoC Mav dataset (Burri et al., 2016). Li et al. (2020) build a SLAM framework using both learned global descriptors and learned sparse features. They test several learned features (D2-Net, SuperPoint, and HF-Net (Sarlin et al., 2019)) and show some improved performance over ORB-SLAM on indoor data as well as outdoor data from the City Centre and New College dataset (Cummins and Newman, 2008).

5.2.3 Summary

We base our pipeline and network architecture on the design by Barnes and Posner (2020) that learns features for RADAR localization supervised solely by pose error. For camera data we found that only relying on pose error was not sufficient, and we needed to include a loss for keypoint matching errors. This loss does not require point correspondence ground truth, but is instead computed form the pose ground truth. Since we are using a stereo camera instead of a RADAR, we also include a stereo camera model that allows us to triangulate 3D points and warp 2D points for the keypoint loss. With a fully differentiable pipeline, we learn features specifically for the task of VO and, particularly, learn weights that decide which keypoints are good for VO. Next we summarize how our method differs from other learned visual features according to the criteria we have discussed at length in this section.

Method Design We use the network design for RADAR features by Barnes and Posner (2020) but make updates to suit our visual sensor. The network consists of an encoder together with one decoder for keypoint detection and another to learn importance scores. An advantage to this network, is that a fixed number of keypoints are detected spread across the image avoiding NMS or other strategies to pick the top keypoints as in, for instance, SuperPoint, R2D2, and ASLFeat. The descriptors are derived directly from encoder layers and resized and concatenated together.

Training The learning pipeline we use is fully differentiable, which allows us to train end-to-end for the task of VO. While Bhowmik et al. (2020) uses a probabilistic approach to make feature matching differentiable, our method matches sparse descriptor vectors in the source image with dense descriptors in the target and finds the resulting target coordinates by weighting. Furthermore, differentiability is preserved by using SVD for point-cloud alignment to estimate the final pose. Several of the direct methods (Lv et al. (2019), BA-Net, Xu et al. (2020), and Kasper et al. (2020)) are also fully differentiable, but they do not detect sparse keypoints, and their pose optimization requires iteration. Finally, Sarlin et al. (2021) also presents a fully differentiable pipeline for vision data. However, they rely on a 3D point cloud and initial pose estimate as input to their method.

Testing The main difference in testing between ours and other methods, is that we use a dataset that includes off-road (UTIAS Multiseason dataset) data, while other approaches that test pose estimation outdoors do so either on trajectories driven by cars in urban environments (ASLFeat, SAND, S2DNet, Sarlin et al. (2021), GN-Net, LM-Reloc, and Kasper et al. (2020)) or on images of buildings or places from different viewpoints (R2D2, D2-Net, Germain et al. (2021), and Bhowmik et al. (2020)). Off-road data provides challenges with vegetation, as well as snow cover, that is not prominent in urban datasets that contain roads and other permanent structures. Although VO does not compute the pose between different appearance conditions, the variety of illumination conditions (dark, low sun) from the UTIAS In-The-Dark and UTIAS Multiseason datasets provide challenging scenarios.

Visual Odometry Of the VO approaches we summarized in Section 5.2.2, only Jau et al. (2020) and Tang et al. (2020) use a fully differentiable learning pipeline. As opposed to them, we did not include additional losses for descriptors or scores. Instead of learning scores for individual keypoints, Jau et al. (2020) use a second network to learn weights for matched point pairs. Their training process is also more complicated that ours, because they first train their SuperPoint network, then proceed with training the point weighting network, before they finally train both networks together end-to-end. While our approach computes pose in one step with SVD, Tang et al. (2020) first solves for an initial pose with a RANSAC and PnP before correcting the pose with a differentiable method. The different VO methods test performance either on indoor or urban datasets, while we include off-road data and challenging lighting conditions.

5.3 Methodology

The neural network for feature extraction and following differentiable pose estimation outlined in this section can be used to estimate relative poses for VO and is end-to-end trainable. We introduce the

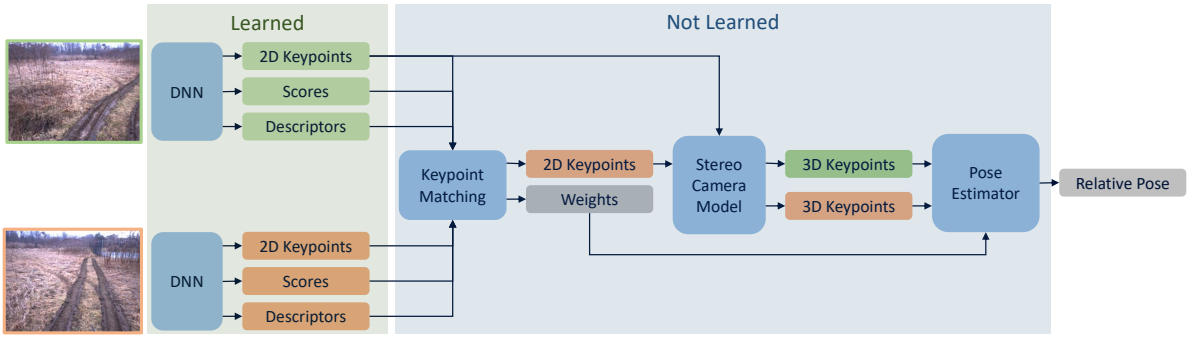


Figure 5.2: In our learning pipeline we pass the images from a source frame (green) and a target frame (orange) to a neural network that predicts keypoints, descriptors, and scores. We match keypoints between the image frames, use the stereo camera model to find their corresponding 3D coordinates, and use the matched point pairs to estimate the relative pose.

neural network architecture, the rest of the components of the pose estimation pipeline as well as the loss functions we use for training.

5.3.1 System Overview

Our fully differentiable training pipeline takes a pair of source and target stereo images and estimates the relative pose, $\mathbf{T}_{ts} \in SE(3)$, between their associated frames. We build on the approach for RADAR localization in Barnes and Posner (2020) with some modifications to allow for the use of a vision sensor. In summary, we use a neural network to detect keypoints and compute their descriptors and scores. Then we match keypoints from the source and target before computing their 3D coordinates with a stereo camera model. Finally, the point correspondences are used in a differentiable pose estimator. For an overview of the pipeline, see Figure 5.2.

5.3.2 Visual Features

The keypoints, descriptors, and scores are extracted from a neural network and matched in a way that preserves the differentiability of the learning pipeline.

Keypoint Detection and Description

We start by detecting sparse 2D keypoints, $\mathbf{q} = \begin{bmatrix} u_\ell & v_\ell \end{bmatrix}^T$, at sub-pixel locations in the left stereo image and computing descriptor vectors, $\mathbf{d} \in \mathbb{R}^{248}$, as well as scores, $s \in [0, 1]$, for all pixels in the image. The descriptor and score for a given keypoint is found using bilinear interpolation. The score is used to determine how important a point is for VO pose estimation. We pass an image to an encoder-decoder neural network following the same design as in Barnes and Posner (2020), illustrated in Figure 5.3. After the bottleneck, the network separates into two decoder branches, one with values needed to compute keypoint coordinates and one for the scores. We divide the image into size 16×16 windows and detect a keypoint for each window by taking the weighted average of the pixel coordinates. We get the weights by computing the softmax over the network output for each window. Applying a sigmoid to the output of the second decoder branch gives us the scores. Finally, the descriptors are found by resizing

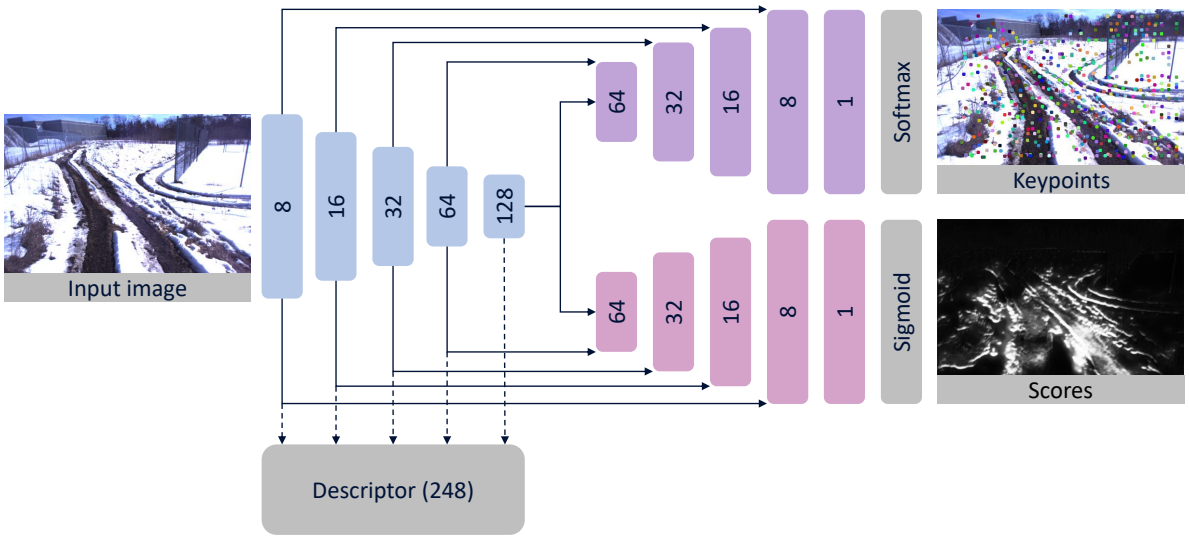


Figure 5.3: Neural network architecture with one decoder to detect keypoints and one to learn scores for all pixels. Descriptors for all pixels are computed by resizing and concatenating the output feature maps of each encoder layer.

and concatenating the output feature maps of each of the encoder blocks, leaving us with a length 248 descriptor vector for each image pixel coordinate.

Keypoint Matching

We have a set of N keypoints for the source image and we need to perform data association between these and points in the target image. Descriptors are compared using zero-normalized cross correlation (ZNCC), which means that the resulting value will be in the range $[-1, 1]$. For each detected keypoint in the source image, $\mathbf{q}_s^i, i \in [1, N]$, we compute a corresponding matched point, $\hat{\mathbf{q}}_t^i$, in the target image. This point is the weighted sum of all image coordinates in the target image, where the weight is based on how well descriptors match:

$$\hat{\mathbf{q}}_t^i = \sum_{j=1}^{M} \sigma(\tau f_{\text{zncc}}(\mathbf{d}_s^i, \mathbf{d}_t^j))\mathbf{q}_t^j. \tag{5.1}$$

M is the total number of pixels in the target image, $f_{\text{zncc}}(\cdot)$ computes the ZNCC between the descriptors, and $\sigma(\cdot)$ takes the temperature-weighted softmax with τ as the temperature, which is determined empirically by trying a range of values. The keypoint matching is differentiable. We found that using all target pixels worked better in practice than only including keypoints detected in the target image. Finally, we find the descriptor, $\hat{\mathbf{d}}_t^i$, and score, $\hat{s}_t^i$, for each computed target point using bilinear interpolation.

5.3.3 Stereo Camera Model

In order to estimate the pose based on our matched 2D keypoints, we need to get their corresponding 3D coordinates, which is straight-forward with a stereo camera. The stereo camera model is described in more detail in Section 2.2.2. The camera model, $\mathbf{g}(\cdot)$, for a pre-calibrated stereo rig maps a 3D point, $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, in the camera frame to left stereo image coordinates, $\mathbf{q}$, together with disparity,

$d = u_\ell - u_r$, as follows:

$$\mathbf{y} = \begin{bmatrix} u_\ell \\ v_\ell \\ d \end{bmatrix} = \begin{bmatrix} \mathbf{q} \\ d \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 0 & f_u b \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{5.2}$$

where f_u and f_v are the horizontal and vertical focal lengths in pixels, c_u and c_v are the the camera's horizontal and vertical optical centre coordinates in pixels, and b is the baseline in metres. The stereo camera model is invertible and its inverse is given by

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{g}^{-1}(\mathbf{y}) = \frac{b}{d} \begin{bmatrix} u_\ell - c_u \\ \frac{f_u}{f_v}(v_\ell - c_v) \\ f_u \end{bmatrix}. \tag{5.3}$$

We obtain disparity, d, using the stereo matching algorithm implemented in OpenCV, which is based on Hirschmuller (2008). We use the inverse stereo camera model to get each keypoint's 3D coordinates.

We have made two adjustments to the stereo camera model compared to how it is defined in Section 2.2.2, and how it is used in Section 3.3.2, to better fit our problem setup in this chapter. In particular, we represent the 2D image coordinates, $\mathbf{q}$, with left camera coordinates only, instead of the full four left and right coordinates as in 2.15. We add disparity together with $\mathbf{q}$ to form the vector $\mathbf{y}$, which then represents the same information as the stacked left and right coordinates. Furthermore, we represent the 3D points, $\mathbf{p}$, in $\mathbb{R}^3$ instead of using homogeneous coordinates as in 2.16.

5.3.4 Pose Estimation

Given the correspondences between the source keypoints, $\mathbf{q}_s^i$, and matched target keypoints, $\hat{\mathbf{q}}_t^i$, we can compute the relative pose from the source to the target, $\mathbf{T}_{ts} = \begin{bmatrix} \mathbf{C}_{ts} & \mathbf{r}_t^{st} \\ \mathbf{0} & 1 \end{bmatrix}$, where $\mathbf{r}_t^{st}$ is the translation from the target frame to the source frame given in the target frame. As described in Section 5.3.3, we use the inverse stereo camera model (5.3) to compute 3D coordinates, $\mathbf{p}_s^i$ and $\hat{\mathbf{p}}_t^i$, from the corresponding 2D keypoints. This means we have two sets of 3D point measurements and need to solve a point-cloud alignment problem. In Section 2.2.3, we describe in detail how to solve this problem and only include the main points here.

To find the relative pose that best aligns the two sets of points, we minimize the following cost function:

$$J = \sum_{i=1}^{N} w^i ||(\mathbf{C}_{ts}\mathbf{p}_s^i + \mathbf{r}_t^{st}) - \hat{\mathbf{p}}_t^i||_2^2. \tag{5.4}$$

The minimization is implemented using SVD. This method does not require iteration and hence does not break the differentiability of our pose estimation pipeline. An iterative method would break differentiability by checking a stopping condition, unless adjustments are made such as stopping after a fixed number of iterations. The weight, $w^i \in [0, 1]$, for a matched point pair is a combination of the learned point scores and how well the descriptors match:

$$w^i = \frac{1}{2}(f_{\text{zncc}}(\mathbf{d}_s^i, \hat{\mathbf{d}}_t^i) + 1) s_s^i \hat{s}_t^i. \tag{5.5}$$

We additionally remove large outliers at training time based on ground truth keypoint coordinate error and using RANSAC at inference.

5.3.5 Loss Functions

Barnes and Posner (2020) supervise training using only a loss on the estimated pose. We found this was insufficient for our approach and add a loss on the image coordinates of the matched keypoints. We generate the keypoint ground truth from the poses and do not require additional keypoint correspondence data. These losses are sufficient and we do not need to add further losses for descriptors or scores. Our goal is to keep the training of the model as simple as possible.

Using the stereo camera model (5.2) and its inverse (5.3), we can form a *warp* operator that transforms the left stereo image coordinates and disparity, $\mathbf{y} = \begin{bmatrix}\mathbf{q}^T & d\end{bmatrix}^T$, from one frame to the corresponding, $\mathbf{y}' = \begin{bmatrix}\mathbf{q}'^T & d'\end{bmatrix}^T$, in another frame given the relative pose, $\mathbf{T} = \begin{bmatrix}\mathbf{C} & \mathbf{r} \\ \mathbf{0} & 1\end{bmatrix}$, between the two camera frames:

$$\mathbf{y}' = \mathbf{w}(\mathbf{T}, \mathbf{y}) = \mathbf{g}\left(\mathbf{C}\,\mathbf{g}^{-1}(\mathbf{y}) + \mathbf{r}\right). \tag{5.6}$$

This is the same warp function as presented in (2.17), but written slightly differently to account for the point representation adjustments discussed in Section 5.3.3. Given the ground truth pose, $\mathbf{T}_{ts}$, we can therefore transform a keypoint in the source image, $\mathbf{q}_s^i$, to its ground truth coordinate in the target image, $\bar{\mathbf{q}}_t^i$, and find its error with respect to the matched target keypoint, $\hat{\mathbf{q}}_t^i$:

$$\mathcal{L}_{\text{keypoint}} = \sum_{i=1}^{N} ||\bar{\mathbf{q}}_t^i - \hat{\mathbf{q}}_t^i||_2^2. \tag{5.7}$$

Taking the ground truth pose, $\mathbf{T}_{ts}$, and estimated pose, $\hat{\mathbf{T}}_{ts}$, we get the following pose loss:

$$\mathcal{L}_{\text{pose}} = ||\hat{\mathbf{r}}_t^{st} - \mathbf{r}_t^{st}||_2^2 + \beta||\hat{\mathbf{C}}_{ts}\mathbf{C}_{ts}^T - \mathbf{1}||_2^2, \tag{5.8}$$

where β determined empirically to balance rotation and translation. The total loss is a weighted sum of $\mathcal{L}_{\text{keypoint}}$ and $\mathcal{L}_{\text{pose}}$, where the weight is determined empirically to balance the two loss terms.

5.4 Experiments

We devise an experiment to test our learned features for VO outdoors under different conditions and on both familiar and new paths.

5.4.1 Data

The datasets, which are presented in detail in Section 2.4, were previously collected using a Clearpath Grizzly robot with a FLIR Bumblebee XB3 camera, see Figure 5.1. The robot autonomously repeats previously taught paths using multi-experience VT&R (Paton et al., 2016). We can use the resulting data as ground truth for supervised learning. In VT&R, stereo image frames are stored as vertices in a spatio-temporal pose graph. Edges contain the relative pose between temporally adjacent vertices and

(a) UTIAS Multiseason (b) UTIAS Multiseason (c) UTIAS Multiseason (d) UTIAS In-The-Dark

(e) UTIAS In-The-Dark (f) UTIAS In-The-Dark (g) West (h) West

(i) East (j) East-Road (k) Lawn

Figure 5.4: Images taken by the robot's camera from the different test paths labeled with their path name. Images from the same path are taken in the same location along the path during different repeats. The data used for testing covers several challenging conditions such as tall grass, driving at after dark, and sun flares.

between a repeat vertex and the teach vertex it has been localized to. We sample image pairs and poses for VO from the pose graph to build a training dataset, as described in Section 2.4.4.

We use data collected from two different paths for training. The UTIAS In-The-Dark dataset contains 39 runs of a path collected at along a road and on grass. The path is repeated over 31 hours and systematically captures lighting change. The UTIAS Multiseason dataset contains 136 runs of a path in an area with more vegetation and undulating terrain. It was repeated from January until May capturing varying seasons and weather.

For testing we use some repeats from these two paths that were not included in the training dataset covering different challenging conditions (snow, night-time, sun flares etc.). Additionally, we test generalizability to paths not seen in the training datasets with data from four more paths, called West, East, East-Road and Lawn. These are described in detail in Section 2.4.3. We note that these test paths were collected in the spring and summer of 2019, which is three years after the UTIAS In-The-Dark training data was gathered and over two years after the UTIAS Multiseason training data was gathered. The first two paths were collected in areas with a lot of vegetation and East is particularly challenging with tall moving grass. East-Road was collected along a road and Lawn on a grass lawn. Example images from the test set in Figure 5.4 show that the data covers many challenging scenarios not present in urban self-driving datasets. In total we have a training set with 115,800 samples and a validation set with 45,171 samples. The validation set is fairly large relative to the training set because it contains

samples from all different paths, not just the UTIAS In-The-Dark and UTIAS Multiseason datasets.

5.4.2 Training and Inference

We train the network by giving it pairs of images and the ground truth relative pose, which we extract from the VT&R pose graph. In addition to the weights used for pose estimation, we remove large outliers based on keypoint error using the ground truth pose during training and with RANSAC during inference. The network is trained using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-5} and other parameters set to default values. We determine the number of training epochs with early stopping based on the validation loss. The network is trained on an NVIDIA Tesla V100 DGXS GPU with batch size 5. Network inference runs on average at 21 ms on a ThinkPad P52 laptop with an Intel ® Core™ i7-8850H CPU, 32 GB of RAM, and an NVIDIA Quadro P2000 4 GB GPU. The full pipeline implemented in Python takes 370 ms, but can easily be made faster by transitioning to C++. Alternatively, the learned features could be inserted into an existing optimized classical VO framework instead of using the learning pipeline for inference.

5.4.3 Visual Odometry

In order to evaluate our approach, we use the pipeline described in Section 5.3 to perform frame-to-frame VO. We compute the relative pose between sequential images from the repeat of a path in the test dataset. The performance of the learned method is compared to using a classical point detector and descriptor with the same pose estimator. We use the SURF detector, descriptor, and descriptor matching implemented in OpenCV. We found that properly weighting matched keypoint pairs is essential to successful VO pose estimation. Since we do not learn scores for the classical method, we instead compute weights based on the method proposed in Maimone et al. (2007), which starts by computing a covariance for each 3D point:

$$\boldsymbol{\Sigma}_{\mathbf{p}} = \frac{\partial \mathbf{g}^{-1}}{\partial \mathbf{y}} \begin{bmatrix} \boldsymbol{\Sigma}_{\ell} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{r} \end{bmatrix} \frac{\partial \mathbf{g}^{-1}}{\partial \mathbf{y}}^{T}, \tag{5.9}$$

where $\boldsymbol{\Sigma}_{\ell} = \boldsymbol{\Sigma}_{r} = \begin{bmatrix} 2^k & 0 \\ 0 & 2^k \end{bmatrix}$ are the covariances for the image coordinates in the left and right frames and k is the octave of the image pyramid at which the keypoint is detected. The inverse camera model, $\mathbf{g}^{-1}$, is defined in (5.3) and:

$$\frac{\partial \mathbf{g}^{-1}}{\partial \mathbf{y}} = \frac{b}{d^2} \begin{bmatrix} c_u - u_{\ell} + d & 0 & u_{\ell} - c_u \\ c_v - v_{\ell} & d & v_{\ell} - c_v \\ -f & 0 & f \end{bmatrix}, \tag{5.10}$$

where we have used the fact that $d = u_{\ell} - u_r$ in our derivation. The scalar weight is then obtained with:

$$w^i = \left(\det(\boldsymbol{\Sigma}_{\mathbf{p}_s^i}) + \det(\boldsymbol{\Sigma}_{\hat{\mathbf{p}}_t^i}) \right)^{-1}. \tag{5.11}$$

Computing $\boldsymbol{\Sigma}_{\mathbf{p}}$ and then taking the determinant means that the disparity term in the denominator of the derivative becomes dominant. Points close to the robot get a very large weight and points far away get a very small weight. On paths with tall grass and vegetation computing disparity can be difficult close to the robot due to challenging left-right matching between the stereo images. Hence, we are left

path	method	RMSE (m)	RMSE (°)	RPE (m)	ATE (m)	inlier
Multiseason	Learned detector, descriptor, score	0.052	**0.52**	**4.36**	**4.56**	483
165 m	Learned detector, descriptor	0.035	0.54	8.08	7.32	**489**
	SURF baseline	**0.034**	**0.52**	5.71	5.92	454
In-The-Dark	Learned detector, descriptor, score	**0.012**	0.24	12.83	15.40	556
250 m	Learned detector, descriptor	0.014	0.29	14.21	14.74	**566**
	SURF baseline	**0.012**	**0.23**	**8.94**	**11.76**	465
West	Learned detector, descriptor, score	0.035	0.78	**7.74**	15.04	**428**
470 m	Learned detector, descriptor	**0.031**	**0.76**	15.25	23.08	**428**
	SURF baseline	0.040	**0.76**	7.86	**12.65**	355
East	Learned detector, descriptor, score	0.045	0.74	9.30	9.60	**354**
190 m	Learned detector, descriptor	0.033	**0.67**	5.97	6.62	348
	SURF baseline	**0.032**	0.68	**3.98**	**3.61**	309
East-Road	Learned detector, descriptor, score	**0.025**	**0.21**	10.81	16.25	572
410 m	Learned detector, descriptor	**0.025**	**0.21**	10.06	7.72	**582**
	SURF baseline	0.049	0.24	**7.12**	**6.24**	390
Lawn	Learned detector, descriptor, score	0.025	0.43	3.89	4.67	528
120 m	Learned detector, descriptor	**0.020**	**0.42**	3.91	3.96	**545**
	SURF baseline	0.035	0.45	**2.98**	**3.16**	431

Table 5.2: RMSE in metres and degrees for translation and rotation for relative poses, Relative Pose Error (RPE) averaged over all possible relative poses, Absolute Trajectory Error (ATE), and average number of point inliers for each path. We compare the two learned methods to the SURF baseline. The approximate length of each path is listed in the first column.

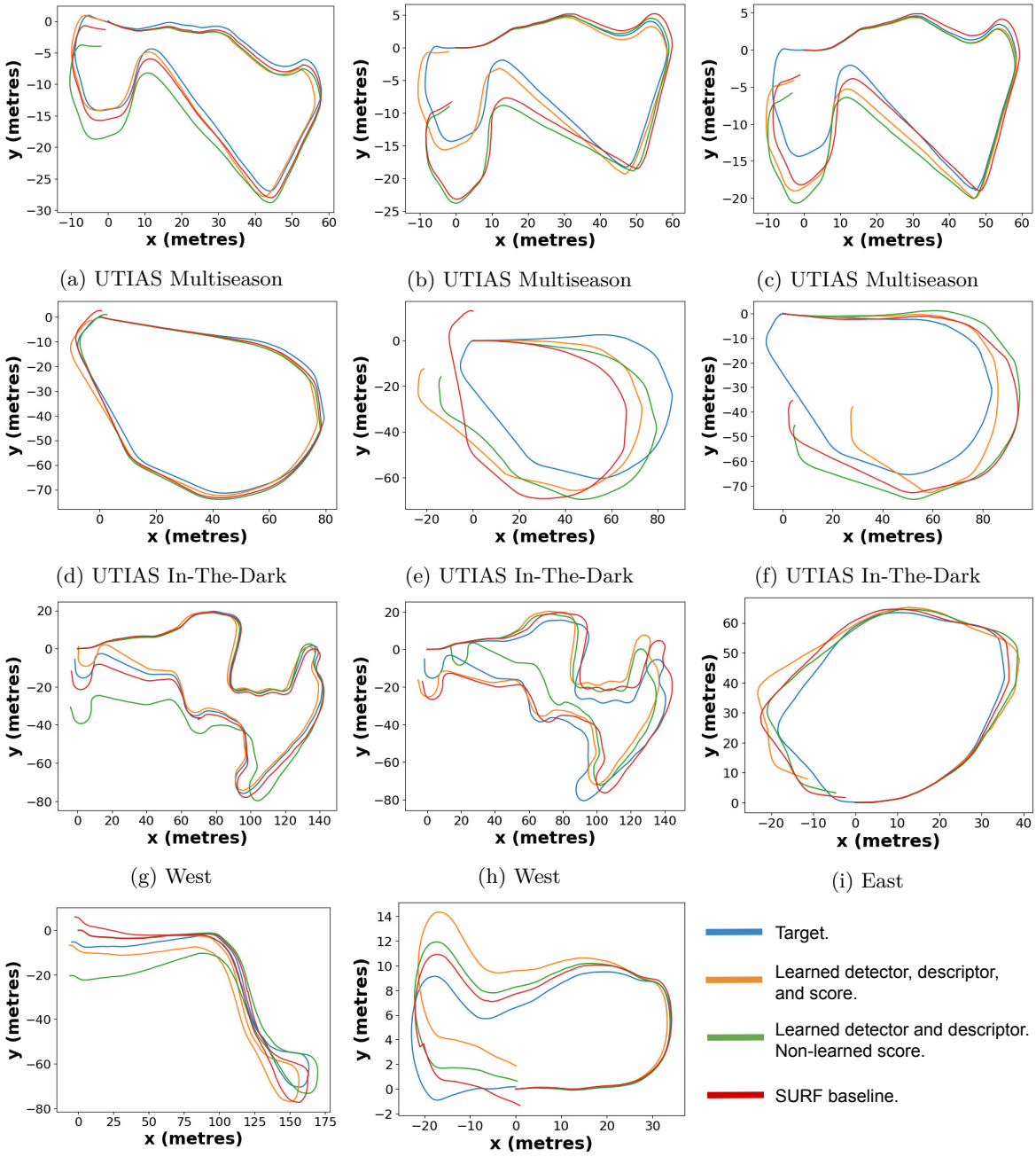
with too many points that have very low weights, causing poor VO performance on some paths. Instead we adjusted the denominator to be d instead of d^2 and found that this resulted in good performance across all paths.

Since we found that weighting keypoint pairs was essential for good VO, we also experiment with the effect of learned and non-learned weights (as defined in (5.11)). All together we run four different experiments. First, we use the learned detector and descriptor with the weights computed from the learned scores and descriptors defined in (5.5). Second, we train the network to learn the detector and descriptors, but use the non-learned weights from (5.11). Third, we use the base-line SURF method as described above, and, finally, we use the SURF detector and descriptor, but try to learn scores. Then the SURF descriptors and learned scores are combined to compute the keypoint pair weights following (5.5).

5.5 Results

In our experiments, we found that using a learned detector and learned descriptors worked both with the learned and non-learned weights. However, learning scores to use with the SURF detector and descriptor was unsuccessful with our learning pipeline. We found that the method very quickly overfit to the training data and gave poor results on the test data. It may be possible to learn such scores, but we believe it would require larger changes to our system design. Hence, we report results on the two versions of the learned method and compare to the baseline SURF approach.

Quantitative results are listed in Table 5.2, which reports the RMSE for translation and rotation in



(a) UTIAS Multiseason (b) UTIAS Multiseason (c) UTIAS Multiseason

(d) UTIAS In-The-Dark (e) UTIAS In-The-Dark (f) UTIAS In-The-Dark

(g) West (h) West (i) East

(j) East-Road (k) Lawn

Figure 5.5: Plots of the integrated paths for different repeats of the test paths. The plots correspond to the images in Figure 5.4 that show the conditions under which the data was collected. The blue line shows the ground truth target path we get from multi-experience VT&R, orange corresponds to the learned method with learned weights, green to the learned method with non-learned weights, and red to the baseline SURF method.

metres and degrees for the relative pose for each test path. We compute relative poses between each pair of adjacent vertices in the pose graph, hence the frames are father apart than the camera frame rate. The vertices are usually spaced about 0.3 metres apart, but can be as far apart as 1.4 metres. In

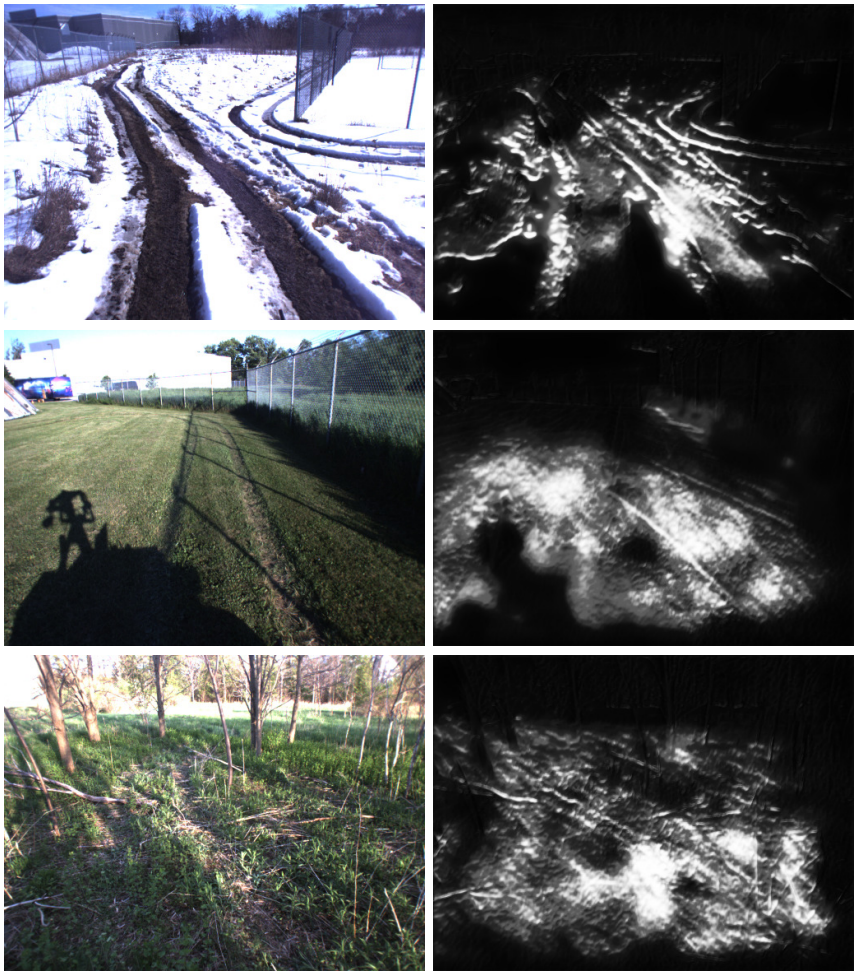


Figure 5.6: Input images and their corresponding learned scores that emphasize which pixels are useful for VO. The brighter the pixel, the higher the score.

addition to this forward distance, the DOF that matter most for path following with a ground robot are the lateral direction and heading. Across the test dataset, the largest lateral offset is approximately 0.4 metres but most values fall between +/- 0.1 metres. While the largest heading offset is about 13°, most values fall in the +/- 6° range. In order to evaluate performance over the whole path, we compute Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) as defined by Sturm et al. (2012). In particular, we average RPE over all possible intervals of relative poses. For frame-to-frame VO, the RPE can be smaller than ATE in cases where errors are larger at the beginning of the path. For the UTIAS Multiseason data, the results cover 9 different repeats with different seasonal conditions, UTIAS In-The-Dark provides 5 test repeats, West has 2 repeats, East has 1, East-Road has 2, and Lawn has 4 repeats. We see that both learned methods perform similarly to the baseline SURF method, but do not improve over it. Furthermore, the learned methods generalize well to the new paths (West, East, East-Road, and Lawn) and provide a higher number of inliers as determined by RANSAC.

For a qualitative visualization of the results, we plot the integrated paths in Figure 5.5. The plots correspond to the images from the paths illustrated in Figure 5.4. For the UTIAS Multiseason data, we show an example where all methods perform similarly well (a), one where the fully learned method

outperforms the two others (b), and the repeat where the fully learned method performs the poorest while the baseline does better (c). For the UTIAS In-The-Dark dataset, the first example (d) is handled well by all approaches, while they all struggle in the more difficult scenarios (e, f) with strong sun flare and night-time driving. For the West path, we show the best (g) and poorest (h) repeats, the first collected during the day and the latter during early evening with low sun. For the remaining paths, we plot the results of one repeat each. East provides challenging data with taller grass than was seen in the training dataset.

In our method we use the learned keypoint score to determine which points are useful for VO. The results indicate that both the weights based on the learned scores and descriptors (5.5) and the non-learned weights based on the camera model (5.11) work well. It is important to note that for the latter case, the network had to be trained from scratch using these weights. Simply inserting them at test time with an already trained detector and descriptors, leads to inferior results. As opposed to the non-learned weights that only rely on the camera parameters and disparity (i.e., distance from the camera), the learned weights take appearance into account. Although this does not play as important a role for VO, where source and target images look similar, this may become important for localization between different conditions. In Figure 5.6, we show examples of the learned score. We see that scores are higher closer to the camera, which is consistent with (5.11). Additionally, the scores emphasize the ground below the horizon and certain parts of the image based on appearance. In particular, in the first example edges and areas of contrast are highlighted. In the second image the moving shadow of the robot is downweighted, while the still shadows from the fence are highlighted. In the final example, the ground is highlighted with edges from sticks getting higher weights. Although difficult to make out at this image size, the darker areas are not entirely black, but do pick out trees, fence posts etc., with a gray colour.

5.6 Conclusions and Future Work

We adapted the method from Barnes and Posner (2020) that learns a keypoint detector, descriptors, and scores for RADAR localization for use in VO. The learning pipeline is fully differentiable, allowing for end-to-end training specifically for the VO task. We show that the learned features perform similarly to a SURF baseline for frame-to-frame VO on challenging outdoor data. Moreover, they generalize to paths not seen during training, which was not possible in our earlier work with a network that estimated pose directly from input images. Although the learned features do not provide a performance benefit over SURF for VO, apart from increasing the number of inliers, we will see that they are useful for the more challenging localization task, where handcrafted features fail. The next step of this work, presented in the following chapter, is to adapt this work to localization and include the learned features in the existing VT&R system for closed-loop path following.

Chapter 6

Deep Learned Features for Long-Term Localization

6.1 Introduction

After setting up a pipeline for relative pose estimation with learned features, and showing that they work and generalize well for VO, we apply the work from the previous chapter to localization. While handcrafted visual features such as SURF work well for VO, they struggle under large appearance change. As we have explained before, VT&R relies on multi-experience localization to handle long-term localization. Intermediate experiences are used to bridge the appearance gap when localizing to the initial map becomes difficult. With the help of learned features, we aim to remove the need for such intermediate experiences for localization, which is the motivation for and ultimate goal of the work in this thesis. We have arrived at a learned approach to pose estimation that is both simple to train from ground truth poses and easy to integrate with classical pose estimation.

In this chapter, we use learning for the perception front-end of long-term localization, while the remaining components of pose estimation are implemented with classical tools, see Figure 6.1. Furthermore, we insert the learned features into the existing VT&R pipeline and perform autonomous path following outdoors. In particular, we teach a path in the day time and repeat it for a full range of lighting conditions, including after dark. In another experiment, we show that the learned features generalize to new areas not present in the feature training dataset. Finally, we complete a seasonal experiment, where we teach a path in August and keep repeating the path for three months until November. The experiments tackling lighting change all achieved a 100% autonomy rate. 21% of the runs in the seasonal experiment experienced path-following failures caused by the compounding effect of seasonal and large illumination change.

For training, we use data collected across lighting and seasonal change by a robot using multi-experience VT&R. The two datasets were collected four and five years prior to our closed-loop experiments. We train a network to provide keypoints with associated descriptors and scores. Using a differentiable pose estimator allows us to backpropagate losses based on poses from the training data. The network is small enough to fit on a laptop GPU, fast enough to run in real time, and simple to train with only two training losses generated directly from pose ground truth.

Our method differs from others that use learned features for localization across environmental change,

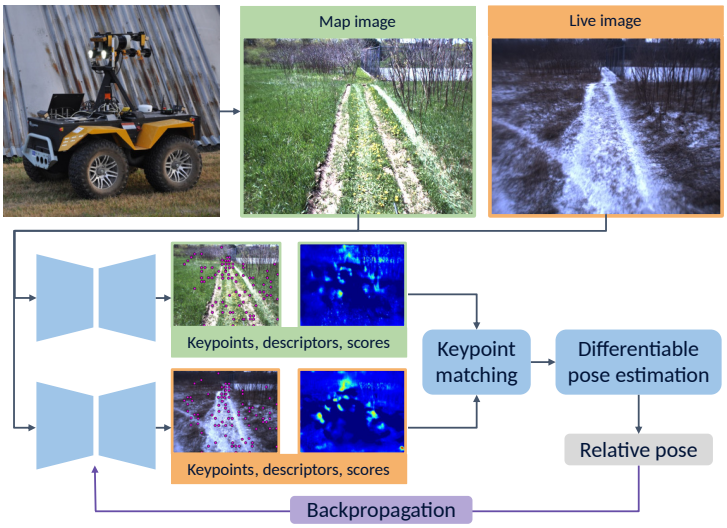


Figure 6.1: We train a neural network to predict keypoints, descriptors, and scores that can be used in classical pose estimation. We train on outdoor data collected with a Grizzly ground robot and later drive the same robot autonomously with the learned features despite lighting and seasonal change.

such as (Piasco et al., 2019; von Stumberg et al., 2020a,b; Kasper et al., 2020; Spencer et al., 2020; Sarlin et al., 2021), since they test localization standalone, while we include our features in the full VT&R system. Gladkova et al. (2021) combine learned features for localization with VO, but only test on datasets. Since our goal is to learn features for path following, we focus on closed-loop performance. Good localization performance on datasets does not guarantee successful real-life autonomous driving, which involves interaction of localization with VO and path-tracking control. Sun et al. (2021) published closed-loop path following with learned features around the same time our work was submitted for publication, but their lighting-change experiments are less extensive than ours and they do not test during seasonal change.

To summarize, the novel contributions of this work are learning features for localization, including them in the VT&R system, and showing successful autonomous path following on a robot across lighting and seasonal change. Moreover, we operate in scenes that include challenging off-road areas with few permanent structures and abundant vegetation, which makes localization challenging under large appearance change. The content in this chapter including the lighting-change experiments appeared in the IEEE Robotics and Automation Letters (RAL) (Gridseth and Barfoot, 2022), and will be presented at ICRA (2022). The seasonal experiments were not included in the publication.

6.2 Related Work

There has been a wide range of work on deep learning for robotic navigation. Our work focuses on learning for pose estimation in visual path following, though others have tackled visual path following, either in simulation or a limited indoor environment, by learning discrete navigation directions (Swedish and Raskar, 2018), learning a visual MPC-policy that can avoid obstacles (Hirose et al., 2019), and learning how to retrace a path under noisy actuation and a changing environment (Kumar et al., 2018). Chen et al. (2020) provide a thorough survey of deep learning for mapping and localization. Some

research has focused on learning pose for localization directly from images via absolute pose regression (Kendall et al., 2015), relative pose regression (Laskar et al., 2017), or combining learning for localization and VO (Valada et al., 2018). Sattler et al. (2019) note that learning pose directly from image data can struggle with accuracy. More structure can be imposed on the learning problem by using features to tackle front-end visual matching, while retaining a classical method for pose estimation. In this section we will discuss relevant work on learned features with focus on localization.

In Section 5.2.3, we reviewed several methods that learn features for pose estimation and created Table 5.1 for comparison. We will not repeat the discussion here, but summarize some of the high-level points. We looked at the design of the different learning pipelines, where one of the main differentiating factors was whether the methods learned only feature matching or provided a fully differentiable pipeline that included pose estimation. We also considered what data was needed to train the network such as point correspondences, poses, or no labels at all. Finally, the data used for testing also matters. Are the features tested on the tasks that we care about? In this chapter we focus on long-term localization and so we want to compare our method to others that tackle localization outdoors under appearance change.

6.2.1 Learned Features for Localization

Several of the methods that tackle localization in outdoor environments have already been described in Section 5.2.1, namely the work from D2-Net (Dusmanu et al., 2019), R2D2 (Revaud et al., 2019), ASLFeat (Luo et al., 2020), S2DNet (Germain et al., 2020), GN-Net (von Stumberg et al., 2020a), LM-Reloc (von Stumberg et al., 2020b), Kasper et al. (2020), Germain et al. (2021), Spencer et al. (2020), Venator et al. (2021), Bhowmik et al. (2020), and Sarlin et al. (2021). Sarlin et al. (2019) present HF-Net, which learns features for localization. They extract local features using the SuperPoint architecture and global descriptors using a NetVLAD layer (Arandjelovic et al., 2016). The local features and global descriptors are combined in a hierarchical approach to localization. HF-Net trains using multi-task distillation, which means that they learn from off-the-shelf trained teacher models for local and global features, respectively. DOAP (He et al., 2018) is used as the teacher network for local features, while NetVLAD is used for the global descriptors.

Gladkova et al. (2021) integrate existing learned features (SuperPoint, R2D2, and ASLFeat) for localization in a classical VO pipeline. The localization poses are used as a prior for frontend tracking and integrated into backend bundle adjustment. Global localization poses are fused with VO estimates. The method does not need training as they use existing learned features as is. Zhao et al. (2021) run VT&R in closed loop by performing visual servoing between topological nodes in a map. Their deep steering network combines learned feature extraction (using the SuperPoint architecture) and matching with learned scale estimation to infer a sequence of linear and angular velocities for servoing to the next node. They do not compute poses based on the matched features but instead rely on histogram voting over the horizontal offsets between matched features to estimate the robot's angular velocity. The authors use odometry in combination with scale estimation to select the goal node and then determine the linear velocity. Sun et al. (2021) drive a robot in closed loop using learned features in VT&R. They use SuperPoint features, but update the original training method to include illumination adaptation using a Gamma Transform. In particular, they start by using the existing bootstrapped point detector, before continuing with homographic adaptation in addition to their own illumination adaptation. For this process they do not need any ground truth poses.

The various methods in Table 5.1 that tackle localization employ different strategies for testing. Most

papers test localization standalone on datasets. ASLFeat, D2-Net, R2D2, Germain et al. (2021), and Bhowmik et al. (2020) test localization on datasets that contain large illumination change, but not data from trajectories. ASLFeat does test on trajectories, but only for VO. As discussed in Section 5.2.3, data from free-standing view points often contain planar scenes with images of buildings. They are less relevant to path following than datasets collected by cars or robots driving along a road or a path. Spencer et al. (2020) test feature matching over illumination and seasonal data. Technically, they also run localization experiments under the same conditions, but pass the learned features to a pre-trained pose regression network and do not rely on classical pose estimation. BA-Net tests performance for localization along a trajectory but only for indoor data. S2DNet, GN-Net, LM-Reloc, Kasper et al. (2020), Sarlin et al. (2019), Sarlin et al. (2021), and Venator et al. (2021) all test localization on the Oxford Robotcar dataset (Maddern et al., 2017), which contains a route through Oxford repeated across different lighting, weather, and seasonal conditions. Additionally, Sarlin et al. (2019) and Sarlin et al. (2021) also test on the CMU Seasons dataset (Sattler et al., 2018) gathered with a car in urban, suburban and park areas across one year. Sarlin et al. (2021) also show good generalization between very different domains with their learned features. For instance, they are able to use features trained on outdoor data for indoor localization.

The downside to the above approaches, is that they all test localization standalone. As mentioned earlier, the work from Gladkova et al. (2021) integrates learned features for localization in a VO pipeline. They test on the Oxford Robotcar dataset and the 4Seasons dataset (Wenzel et al., 2020), which is another dataset collected while driving in an urban environment. Zhao et al. (2021) perform autonomous visual path following with a robot based on their learned velocities, but only operate indoors without any appearance change. Sun et al. (2021) moves away from datasets and drive a robot in closed loop using learned features in VT&R. They test day-to-night localization, though their experiments are over a short time range. Furthermore, they test lighting change in on-road areas such as a parking lot and by a church.

6.2.2 Summary

We base our learning pipeline and network architecture on the design presented by Barnes and Posner (2020), which learns keypoints, descriptors, and scores for RADAR localization. We chose this method because it required only a pose loss for supervision and provided a simple network design with a fully differentiable learning pipeline. For camera data, we found that only relying on pose error was not sufficient and needed to include a loss for keypoint matching errors. This loss does not require point correspondence ground truth, but is instead computed form the pose ground truth. Since we are using a stereo camera instead of a RADAR, we include a stereo camera model that allows us to triangulate 3D points and warp 2D points for the keypoint loss. With a fully differentiable pipeline, we learn features specifically for the task of localization and apply them to real-time closed-loop path following. Next, we summarize how our method differs from the other learned visual features according to the criteria we have discussed at length in this section.

Method Design We use the network design devised by Barnes and Posner (2020) for RADAR localization. It consists of an encoder together with one decoder for keypoint detection and another to learn importance scores. The network detects a fixed number of keypoints spread evenly across the image and, therefore, avoids pruning with NMS or other strategies as required by, for instance, SuperPoint, R2D2,

and ASLFeat. The descriptors are derived directly from encoder layers and resized and concatenated together. Overall, network size and run time is important for our real-time application. Since they use deep learned feature for real-time closed-loop operation, both Zhao et al. (2021) and Sun et al. (2021) use the SuperPoint architecture because of its runtime performance. When compared to SuperPoint, R2D2, and D2-Net (see Section 6.7.1), our network has the fastest feature extraction time and second smallest memory utilization. In particular, our network is faster than SuperPoint and uses less memory.

Training Our learning pipeline is fully differentiable, which allows us to train end-to-end for localization. While Bhowmik et al. (2020) use a probabilistic approach to make feature matching differentiable, our method matches sparse descriptor vectors from the source image with dense descriptors in the target and finds the resulting target coordinates by weighting. Furthermore, differentiability is preserved by using SVD for point-cloud alignment to estimate the final pose. Several of the direct methods (BA-Net, Lv et al. (2019), Xu et al. (2020), and Kasper et al. (2020)) are also fully differentiable, but need to unroll iterative pose optimization with a fixed number of iterations. Finally, Sarlin et al. (2021) also presents a fully differentiable pipeline for vision data, however, they rely on a 3D point cloud and initial pose estimate as input to their method.

Testing While most learned feature approaches test localization standalone, Gladkova et al. (2021) integrate learned features for localization in a classical VO pipeline and test on urban datasets with illumination and seasonal change. Zhao et al. (2021) test their learned velocities for path following in closed-loop on a robot but only in an indoor environment. Similarly to Sun et al. (2021), we integrate our learned features into VT&R and perform closed-loop path following on a robot. While Sun et al. (2021) perform day-to-night localization over a limited time range in an urban environment, we show successful localization across a full range of lighting change, including driving during the dark, sunrise and sunset. Moreover, we include driving in an off-road area with tall grass, bushes and trees, which leads to additional challenges for localization. Finally, we also include a three month seasonal experiment from mid-August to mid-November.

6.3 Methodology

The method outlined in this Section lets us extract learned visual features that we can train end-to-end and use with a classical pose estimator. The learning pipeline remains the same as in Chapter 5, apart from some adjustments to the loss functions. For this reason, we provide an overview of the system, but do not repeat the detailed explanations from the previous chapter.

6.3.1 System Overview

Our fully differentiable training pipeline, see Figure 6.2, takes a pair of source and target stereo images and estimates the relative pose, $\mathbf{T}_{ts} \in SE(3)$, between their associated frames. We build on the approach for RADAR localization in Barnes and Posner (2020) with some modifications to use a vision sensor. In short, the neural network (illustrated in Figure 6.3) detects keypoints and computes their descriptors and scores. We match keypoints from the source and target before computing their 3D coordinates with a stereo camera model. Finally, the point correspondences are used in a differentiable pose estimator based on SVD for point-cloud alignment. See Figure 6.4 for example matched keypoints and the learned

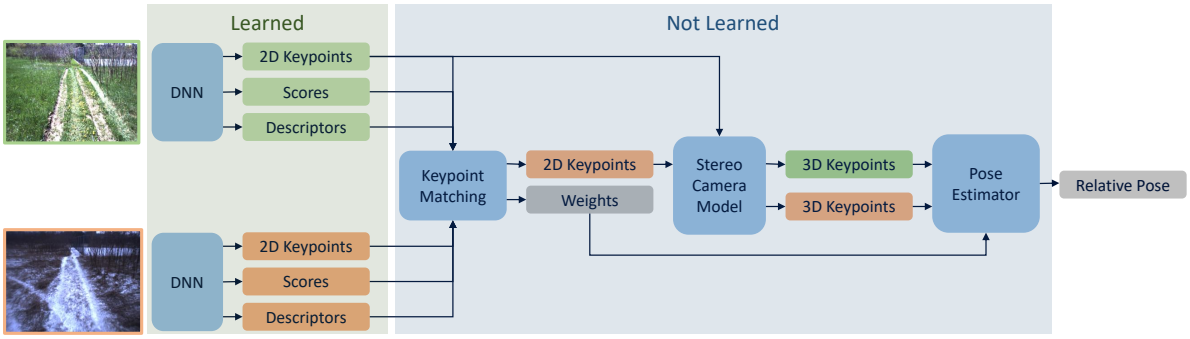


Figure 6.2: In the learning pipeline, we pass images from a source frame (green) and a target frame (orange) to a neural network that predicts keypoints, descriptors, and scores. We match keypoints between the image frames, use the stereo camera model to find their corresponding 3D coordinates, and use the matched point pairs to estimate the relative pose.

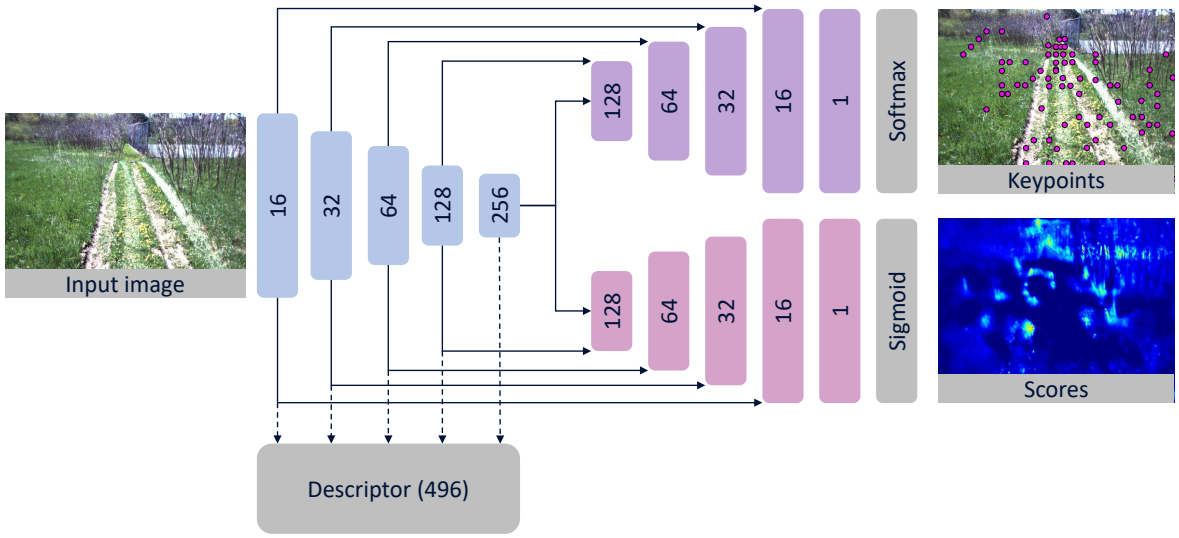


Figure 6.3: Neural network architecture with one decoder to detect keypoints and one to learn scores for all pixels. Descriptors for all pixels are computed by resizing and concatenating the output feature maps of each encoder layer.

scores. When tackling localization, instead of VO, we add more capacity to network relative to the architecture in Section 5.3.2. This leaves us with a descriptor vector of size 496.

6.3.2 Loss Functions

Since the learning pipeline is fully differentiable, we can train the network end-to-end from ground truth poses. Barnes and Posner (2020) supervise training using only a loss on the estimated pose. As in the VO case, we found this was insufficient for our approach and therefore include a loss on the 3D coordinates of the matched keypoints, similar to Christiansen et al. (2019). We generate the keypoint ground truth from the poses and do not require additional keypoint correspondence data. These losses are sufficient and we do not include further losses for descriptors or scores.

For our training datasets, we only use a subset of the pose DOF for supervision due to accuracy

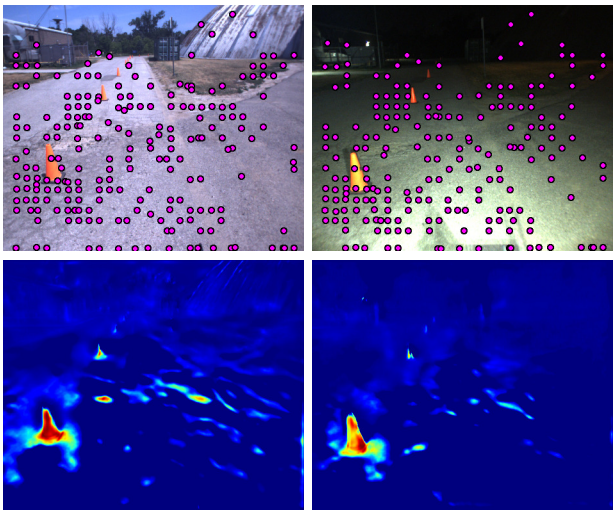


Figure 6.4: Example detected and sparse-to-densely matched keypoints with associated scores for a source and target pair of images. The shown keypoints are inliers after feature matching. The lowest scores are dark blue and the highest scores are red.

variability in the remaining DOF for some sequences. Specifically, we use the robot longitudinal direction, α, lateral offset, β, and heading, γ. When training the learned features for VO, we used the full six DOF poses. For this reason, we modify the keypoint loss from (5.7) to use fewer DOF. Using the inverse stereo camera model from (5.3), we can compute the 3D coordinates of the keypoints in the source and target camera frames. Given the ground truth pose, $\mathbf{T}_{ts}$, we form a pose

$$\mathbf{T}'_{ts} = \begin{bmatrix} \mathbf{C}'_{ts} & \mathbf{r}_t^{st\prime} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 1 & \alpha \\ \sin(\gamma) & \cos(\gamma) & 1 & \beta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

that we use to transform the source keypoints. Because we transform the source points in the plane, we only compare the x and y point coordinates:

$$\mathcal{L}_{\text{keypoint}} = \sum_{i=1}^{N} ||\mathbf{T}'_{ts}\ \mathbf{p}_s^i\big|_{z=0} - \hat{\mathbf{p}}_t^i\big|_{z=0}||_2^2. \tag{6.2}$$

Forming $\mathbf{T}'_{ts}$ from the ground truth pose and $\hat{\mathbf{T}}'_{ts}$ from the estimated pose, we get the following updated pose loss:

$$\mathcal{L}_{\text{pose}} = ||\hat{\mathbf{r}}_t^{st\prime} - \mathbf{r}_t^{st\prime}||_2^2 + \lambda||\hat{\mathbf{C}}'_{ts}{\mathbf{C}'_{ts}}^T - \mathbf{1}||_2^2, \tag{6.3}$$

where λ is used to balance rotation and translation. The total loss is a weighted sum of $\mathcal{L}_{\text{keypoint}}$ and $\mathcal{L}_{\text{pose}}$, where the weight is determined empirically to balance the influence of the two terms.

6.4 Implementation

In this section we will discuss some of the practical considerations that went into using the learned features in a closed-loop system. We describe the data used to train our networks and how the training

was done. We also explain how we added the learned features to the existing VT&R system and how we made it run in real time for autonomous path following.

6.4.1 Data

In order to train the network that will compute the keypoints, descriptors, and scores, we need training, validation, and testing data. We use the UTIAS In-The-Dark and UTIAS Multiseason datasets that we have described in detail in Sections 2.4.1 and 2.4.2, respectively. These datasets are particularly suited for localization because they contain many runs of the same path across large appearance change. In particular, the UTIAS In-The-dark dataset has 39 runs collected in an mostly on-road area over 31 hours in summer 2016, providing a full range of lighting change. The UTIAS Multiseason dataset has 136 runs of the same path collected in an off-road area from January until May in 2017, including everything from full snow cover to green grass.

As explained in Section 2.4.4, we generate training, validation, and testing datasets by sampling images and relative poses from the VT&R spatio-temporal pose graph. Because the UTIAS In-The-Dark and UTIAS Multiseason datasets have a large number of runs, we get a large set of potential samples. We create one set of training, validation, and testing datasets from the UTIAS the In-The-Dark data and another set from the UTIAS Multiseason data. For the feature training and validation data, we randomly draw samples from the pose graphs and pick vertices that are topologically up to three steps away from each other. Both datasets that we generated, have 100,000 training samples and 20,000 validation samples. For testing, we do not extract random samples, but pick pairs of held-out runs (using one as the teach and one as the repeat) and extract images and relative poses for the entire sequence to emulate localization in path following.

6.4.2 Network Training and Inference

We train the network by giving it pairs of images and the corresponding ground truth relative poses from the training dataset. Since our learning pipeline involves pose estimation, we remove large outliers after feature matching. During training, we can easily remove these based on the residual error between matched keypoints in the target frame and their ground truth coordinates. When running inference with the pipeline, large outliers are removed with RANSAC as we do not need to worry about differentiability.

The network is trained using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-5} and other parameters set to default values. We determine the length of training with early stopping based on the validation loss. Because we have such a large training dataset, we do not wait for a complete parse of the 100,000 samples before we compute the validation loss. Instead, we randomly pick and train on 10,000 samples from the training data and compute the validation loss over 2,500 random samples from the validation data. The network is trained on an NVIDIA Tesla V100 DGXS GPU with batch size 3 or 4 depending on available space. Feature extraction on this server using PyTorch takes on average 14.8 ms, while it takes 7.3 ms using C++ in the VT&R system running on a ThinkPad P52 laptop with an Intel ® Core™ i7-8850H CPU, 32 GB of RAM, and an NVIDIA Quadro P2000 4 GB GPU.

For our experiments, that we describe in more detail in Section 6.5.6, we train two different networks. We train a network with data only from the UTIAS In-The-Dark dataset, which we use to test robustness to lighting-change in a closed-loop experiment in a mostly on-road area. This network has a higher number of network weights than the network illustrated in Figure 6.3, with the first layer being of size

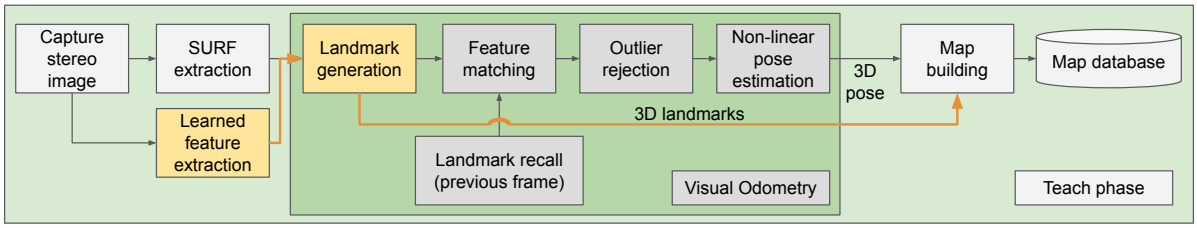


Figure 6.5: An overview of the main blocks of the VT&R teach phase with our updates to include learned features marked with yellow boxes and orange lines. Image capture and feature extraction is followed by VO for pose estimation. We introduce extraction of learned features, and update the landmark triangulation to include the computation of disparity images. The learned features are passed to map building to be stored in the map but are not used for VO, which relies on SURF.

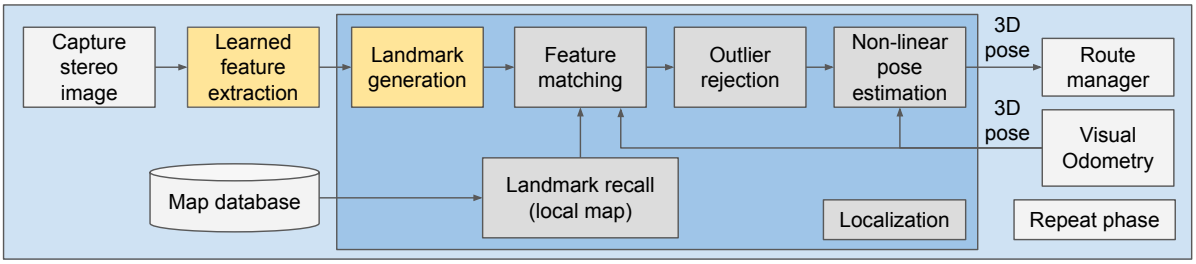


Figure 6.6: An overview of the main blocks of the VT&R repeat phase with our updates to include learned features marked in yellow. For localization, SURF are replaced by learned features and landmark triangulation is updated to include computation of disparity images. SURF are still used in VO, which acts together with localization in a predictor/corrector fashion. The relative pose provides the offset to the path needed for path tracking.

32 instead of 16. This was the first network we used in closed-loop path-following with VT&R and with more experimentation, we later found that using fewer weights was sufficient. Naturally, it was beneficial to reduce the network size when trying to reduce processing time for real-time operation. This network was trained on 165 randomly generated subsets of the training data, meaning it has trained on a total of $165 \times 10{,}000 = 1.65$ million samples, which took approximately 8.5 days.

The final network was trained with the aim to drive both in an on-road and off-road area, and so we used both the UTIAS In-The-Dark and UTIAS Multiseason datasets. We started by training on the UTIAS Multiseason dataset before fine-tuning with the UTIAS In-The-Dark data. We trained on $391 \times 10{,}000 = 3.91$ million samples from the UTIAS Multiseason dataset, which took approximately 14.5 days, and another $194 \times 10{,}000 = 1.94$ million samples from UTIAS In-The-Dark, which took a bit over 6.5 days. Due to time constraints, we did not try to train with samples from both datasets at the same time given that our approach worked well for the experiment we were conducting. In Appendix A, we elaborate on the training process and include plots of losses and errors over time. We also discuss how we determined when the features performed well enough to transition to closed-loop testing.

6.4.3 Learned Features in Visual Teach and Repeat

In order to test the performance of the learned features in closed-loop on a robot, we add them to the VT&R system, which normally relies on sparse SURF for localization. We have explained VT&R in detail in Section 2.3.4, and here we will focus on the parts we adapted to use learned features. Since

(a) Manual teach at 12:56 to build a visual map.

(b) Autonomous repeat after dark at 17:22.

Figure 6.7: In VT&R, the user teaches a path by driving the robot manually (a), after which the path is repeated autonomously (b). For each figure the UI (left) shows the estimated path and the robot's position. On the right we see an external view of the Clearpath Grizzly robot (top) together with the live camera frame and matched features (bottom) for VO during the teach (SURF) and localization during the repeat (learned features).

VT&R is modular with different components for tasks such as feature extraction, feature matching, point triangulation, VO, and localization, the code we needed to implement was limited to creating a new feature extractor, computing disparity images, and enforcing single-experience localization. Some additional adjustment were necessary, but these involved minor tasks such as creating new variables and parameters etc. so that the new features could be used in subsequent parts of the pipeline.

For the teach phase, we update VT&R by extracting learned features, triangulating landmarks using disparity images, and storing them in them map. See Figure 6.5 for an illustration of the updates to the teach phase. For the repeat phase, we extract learned features, triangulate landmarks using disparity images, and use the learned features for localization, see Figure 6.6. In the rest of the section we will explain how these changes were made.

In VT&R, a new type of visual features can be introduced by writing a class for feature extraction. Extractors for SURF and Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011) already exist. In multi-experience VT&R, SURF are used for both VO and localization. We use learned features instead of SURF for localization but keep the VO implementation with SURF as is. We need to find the disparity for each detected keypoint so that they can be triangulated to find the corresponding 3D coordinates. For SURF this is done for each keypoint individually by matching the keypoints detected in the left and right image using their SURF descriptors. Since our descriptors are only trained for localization, we do not use them for left-right stereo matching but instead compute disparity images. We use the StereoSGM class in OpenCV, which is based on Hirschmuller (2008) and allows us to add the computation on the GPU. Computing disparity images on the CPU was too slow for real-time operation.

In order to extract keypoints, descriptors, and scores from the neural network, we convert our PyTorch model to TorchScript such that we can use it with libtorch in C++. We run VT&R on a ThinkPad P52 laptop with an Intel® Core™ i7-8850H CPU, 32 GB of RAM, and an NVIDIA Quadro P2000 4 GB GPU. In addition to the forward pass of the neural network, the GPU is used to compute SURF and disparity images. It turned out that our original version of the full-size network with the first layer of size 32 was too big to fit on the GPU. We solved this problem by cropping away the left edge of the image such that it is of size 464×384 instead of 512×384. While training the network in PyTorch, we used OpenCV's StereoSGBM algorithm to compute disparity images because it has a Python implementation. With

this algorithm, also based on Hirschmuller (2008), a fixed-width left edge of the left image does not get disparity values and, hence, the loss was not backpropagated through the keypoints detected in this part of the image. Therefore, cropping away this part of the image during operation with VT&R did not result in loss of information as the keypoints would have been masked out regardless. After our initial closed-loop experiment, we found that using a network with fewer features (first layer of size 16 instead of 32) was feasible and allowed us to further improve computation time. As mentioned in the previous section, one forward pass of the network takes on average 7.3 ms on the laptop.

After the learned features have been extracted for a live image during the repeat phase, they are compared to learned features from an image in the map. We normalize the descriptors and use the same method to compute the distance between descriptors as already used for SURF. As explained in Section 5.3.2, in the training pipeline, we match a sparse descriptor from the source image densely to all descriptors in the target image and compute target coordinates. Doing this type of matching would require a bigger implementation change in VT&R, would likely use up too much space on the GPU, and possibly slow processing. Therefore, we opted to use the sparse nearest neighbour matching of descriptors already implemented in VT&R. Some additional checks are performed, such as comparing against a depth threshold, comparing depth for matched points, checking the distance in pixels between matched keypoints etc. Relying on sparse feature matching worked well in the closed-loop experiments.

In VT&R, we rely on RANSAC to remove outliers from the set of matched points. The inliers are then passed on to pose estimation. We did not make any adjustments to the pose estimation in VT&R for these experiments. The scores output by the network for each keypoint can be included to weight matched points in future work. The scores helped pose estimation during training, which relied on a simple single-frame method with SVD. In VT&R, which uses a more sophisticated approach with a prior from VO and a window of frames, we got pose estimates that were good enough for path following without using the scores. The final change we made to VT&R was to force the pipeline to only run single-experience localization. Originally, the system uses multiple experiences if they are available in the pose graph.

The main challenge of adding the learned features into the VT&R system was ensuring real-time operation. Because we could not remove the detection and matching of SURF needed for VO, we added additional computation to the system without freeing up processing. We made some adjustments such as cropping the image, using a network with fewer features, and disabling visualizations. Moreover, we reduce the speed of the robot from 1.0 m/s to 0.8 m/s to avoid delays in localization. Meanwhile, there are further improvements that can be pursued in order to improve computation time, and we discuss these under future work in Section 7.3.3. We also tested on a newer version of the same laptop from 2020 that has a larger, 12 GB GPU. In this case, we were able to raise the speed of the robot to 1.5 m/s without localization slowing down. The repeats were only run during the daytime one day after the path was taught (also during daytime) and so the appearance change was not very challenging. This indicates that processing would keep up with driving at higher speeds, but does not guarantee that the learned features' performance remains robust if the appearance change was larger.

To summarize, we have converted our learned model to C++, created a new feature extractor in VT&R for learned features, and updated stereo-matching. We are able to run the code in real time on a Lenovo laptop. Figure 6.7, shows a teach and repeat we did with learned features for a live demo.

Experiment	Environment	Conditions	Dates	Distance
Lighting Change	On road	Lighting change	02.08.2021 - 09.08.2021	10.9 km
Generalization I	On road and off road	Lighting change	14.08.2021 - 20.08.2021	20.6 km
Generalization II	Off road	Lighting change	11.11.2021 - 12.11.2021	2.2 km
Generalization III	On road	Lighting change	11.11.2021 - 12.11.2021	1.8 km
Seasonal Change	On road and off road	Seasonal change	14.08.2021 - 19.11.2021	52.3 km

Table 6.1: Overview of the closed-loop experiments to test the robustness of learned features to lighting change and seasonal change, as well as their ability to generalize to new areas not seen in training data. We have driven 87.8 km across all the experiments.

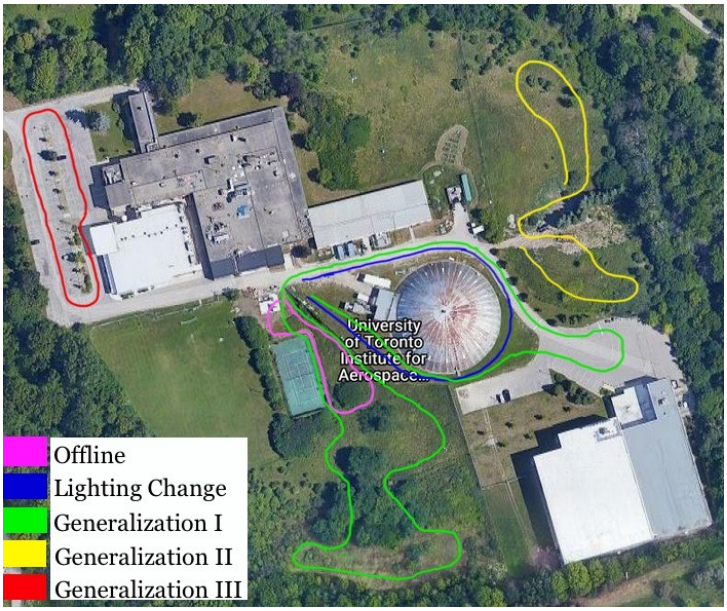


Figure 6.8: Aerial view of UTIAS, where all experiments were run. The approximate paths for the different are overlayed. The green path is used for both the Generalization I experiment and the Seasonal Change experiment.

6.5 Experiments

We have completed four experiments to test the ability of the learned features to localize as appearance changes. We start with an offline experiment to compare our learned method against other state-of-the art learned features. For this, we test localization-only on held-out repeats from the UTIAS Multiseason dataset. The rest of the experiments are performed in closed loop with the Clearpath Grizzly robot. We aim to asses the features' robustness to a full range of lighting change, including driving after dark with headlights. We also investigate the learned features' ability to generalize to new areas that were not seen in the training data. Finally, we complete a long-term experiment that encompasses seasonal change from August to November. During the three closed-loop experiments, which are summarized in Table 6.1, we complete 87.8 km of driving across four different paths.

Figure 6.9: Images of the robot driving in the areas, where we conduct the experiments summarized in Table 6.1. The first row of images are taken on the green path in the large off-road area below the tennis court in Figure 6.8, the second row on the green and blue paths in the area around the Mars Dome, the third on the yellow path in the off-road field, and the fourth row on the red path in the parking lot. The images are taken at different times during closed-loop experiments from August through November.

6.5.1 Environment

The experiments were either run on data previously collected at UTIAS or run in closed loop at UTIAS. The offline comparison to state-of-the-art methods was run on previously collected data from the UTIAS Multiseason dataset, whereas all other experiments were run in closed loop. See figure 6.8 for an overhead view of the institute and explanation of which areas were used for testing. The data used for training, explained in detail in Section 2.4 and Section 6.4.1, was also collected at UTIAS. In order to give an impression of what the test environments look like, we show pictures in Figure 6.9 taken of the robot during autonomous operation from each part of UTIAS described in Figure 6.8. We conduct testing in two off-road areas that contain abundant vegetation such as tall grass, bushes, and trees. The two on-road areas contain views of permanent structures such as buildings, storage units, fences, and street lights.

The offline experiment spans appearance change from January to May in an off-road area with conditions as different as complete snow cover and green grass. The closed-loop experiments are completed either across short times spans (a few days) to capture lighting changes or a long time span to capture seasonal changes. The experiments were run during August until November, 2021. The seasonal changes are naturally strongest in the off-road areas. At the start, in August, trees and bushes had green leaves and there was tall, but partially dry grass. As we went into autumn, some leaves changed colour, while trees and bushes slowly lost more and more leaves. The grass also changed, initially growing and be-

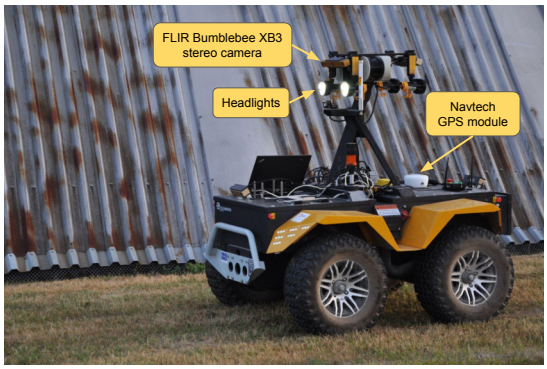


Figure 6.10: The Clearpath Grizzly robot with a forward-facing FLIR Bumblebee XB3 stereo camera, LED headlights, and a Navtech GPS.

coming greener as the weather cooled down, but turning brown and less dense later into autumn. The images in Figure 6.9 illustrate some of these seasonal changes. The day-to-night lighting changes were most drastic off road. While the on-road areas received some light from buildings or street lights, the robot's headlights provided the only light in the dark off-road areas.

6.5.2 Hardware Configuration

Closed-loop experiments are conducted with the Clearparth Grizzly robot, pictured in Figure 6.10. We use a factory-calibrated FLIR Bumbleebee XB3 stereo camera with 24 cm baseline and 16 Hz frame rate as the only sensor for VT&R. The camera is mounted on a mast and the extrinsic transformation between the robot base and the camera is hand-measured. The robot has two forward-facing and two backward-facing LED lights that can be used while driving during the dark. There is also a GPS unit, which we used together with a base station to get RTK GPS ground truth pose measurements for one of our experiments. VT&R was run exclusively on a ThinkPad P52 from early 2019 with an Intel® Core™ i7-8850H CPU, an NVIDIA Quadro P2000 4 GB GPU, and 32 GB of RAM.

6.5.3 Experiment 1: Offline Comparison

We start the experiments by comparing our deep-learned features to other methods. We pick SuperPoint (DeTone et al., 2018b), D2-Net (Dusmanu et al., 2019), and R2D2 (Revaud et al., 2019) because they are state-of-the-art methods with code available online that represent different strategies to learning visual features. This comparison is done offline in PyTorch (not in VT&R) for localization only on held-out repeats from the UTIAS Multiseason dataset. We compare feature matching performance, memory usage, and run time for feature extraction. The test runs cover conditions from winter with snow or clear ground to green grass in spring. Images from the different runs can be seen in Figure 6.12 and an overhead view of the path is depicted in Figure 6.11. We test localization on all combinations of these conditions, meaning that each run is used as a teach and localized against each of the remaining runs.

The SuperPoint method uses one encoder, one decoder for keypoint detection, and another decoder to compute descriptors of size 256. Unlike our decoder, which has several layers of upsampling, SuperPoint only relies on one level of non-learned up-sampling. SuperPoint also uses NMS to pick a set of keypoints that are evenly distributed across the image. D2-Net proposes a describe-and-detect methodology, where detection is not done on low-level image structures. Rather it is done at a higher level jointly with image

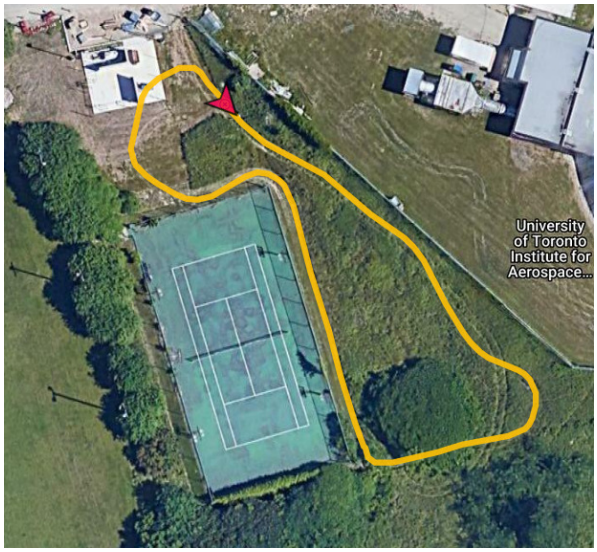


Figure 6.11: An overhead view of the path for the UTIAS Multiseason dataset, which we use for the offline comparison experiment, overlaid on a Google Earth image.

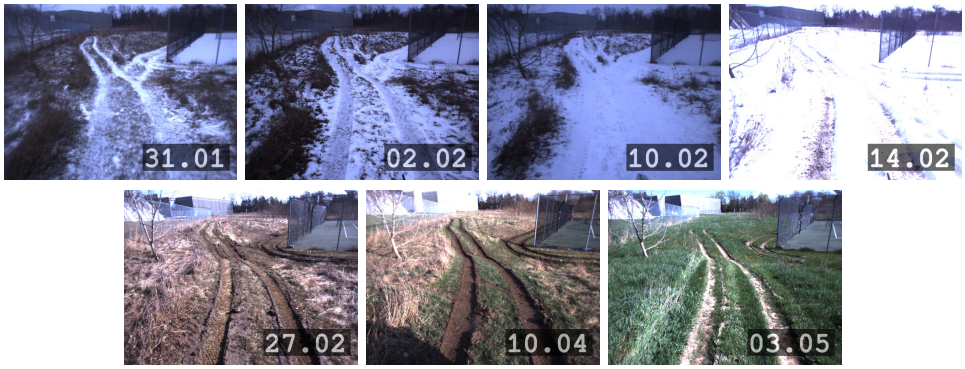


Figure 6.12: Images taken by the robot from the held-out runs from the UTIAS Multiseason dataset that we use for the offline comparison of different methods. Each image is labeled with the collection date (dd.mm) of the corresponding run.

description. They use a truncated, pre-trained VGG-net and fine-tune the last layer during training which gives them descriptors of size 512. Finally, R2D2 uses a fully connected network that outputs dense descriptors, a heat map of repeatability for detecting sparse keypoints, as well as a reliability score for all pixels. During test time they run detection on different scales of the image and choose the best keypoints to keep based on repeatability and reliability.

We ran this comparison on just the localization task in PyTorch instead of full path following in VT&R. The code for the other methods is provided in PyTorch, and it turned out to be difficult to convert all of these networks into C++ without making changes that could potentially introduce errors and compromise their performance. Our method uses the network trained on the UTIAS Multiseason and UTIAS In-The-Dark datasets (described in Section 6.4.2), while we use weights provided online for the other methods. Since we test on off-road data that is quite different from the more commonly used urban datasets that other methods use for training, this limits the usefulness of our experiment to compare feature matching performance. However, it does serve as a sanity check to make sure that our

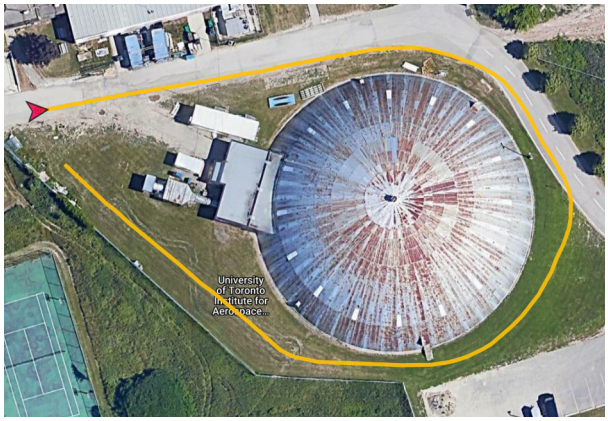


Figure 6.13: The path we taught on August 2nd for the lighting-change experiments is shown overlaid on a Google Earth image. It is driven in the same area as the UTIAS In-The-Dark path.

Figure 6.14: The robot repeating the path for the lighting-change experiment under different appearance conditions.

method trained on relevant data is not outperformed by the others. Moreover, the design of the different feature regression networks or the potentially limited difference in feature matching performance between them is not very important to us. The training method, which allows us to train features tailored for the task of localization in a fully differentiable pipeline, is the main reason we chose the approach from Barnes and Posner (2020) over other state-of-the-art methods. Finally, given the importance of run-time performance for closed-loop operation, the comparison of network memory usage and run time is very useful for our application.

6.5.4 Experiment 2: Lighting Change

The ultimate goal of our work is to develop learned features for long-term localization during autonomous operation. The remaining experiments of this chapter all run in closed loop with the Clearpath Grizzly robot. First, we evaluate the features' robustness to drastic lighting change. We use the network trained on the UTIAS In-The-Dark dataset only. The training of this specific network was described in detail in Section 6.4.2.

We drive the robot manually to teach a new path in the same area as the UTIAS In-The-Dark dataset was collected, see Figure 6.13. This means we do not use the path from the dataset, but teach a new, similar path. Hence, this experiment assesses the learned features' robustness to drastic appearance change, but does not attempt to generalize to new environments not seen in the training dataset. The UTIAS In-The-Dark data were gathered across two consecutive, sunny days (with cloud cover for a small number of runs) at the end of July, 2016, and our experiment was conducted on mostly sunny days in August, 2021. Thus the weather and appearance conditions are mostly similar between the training

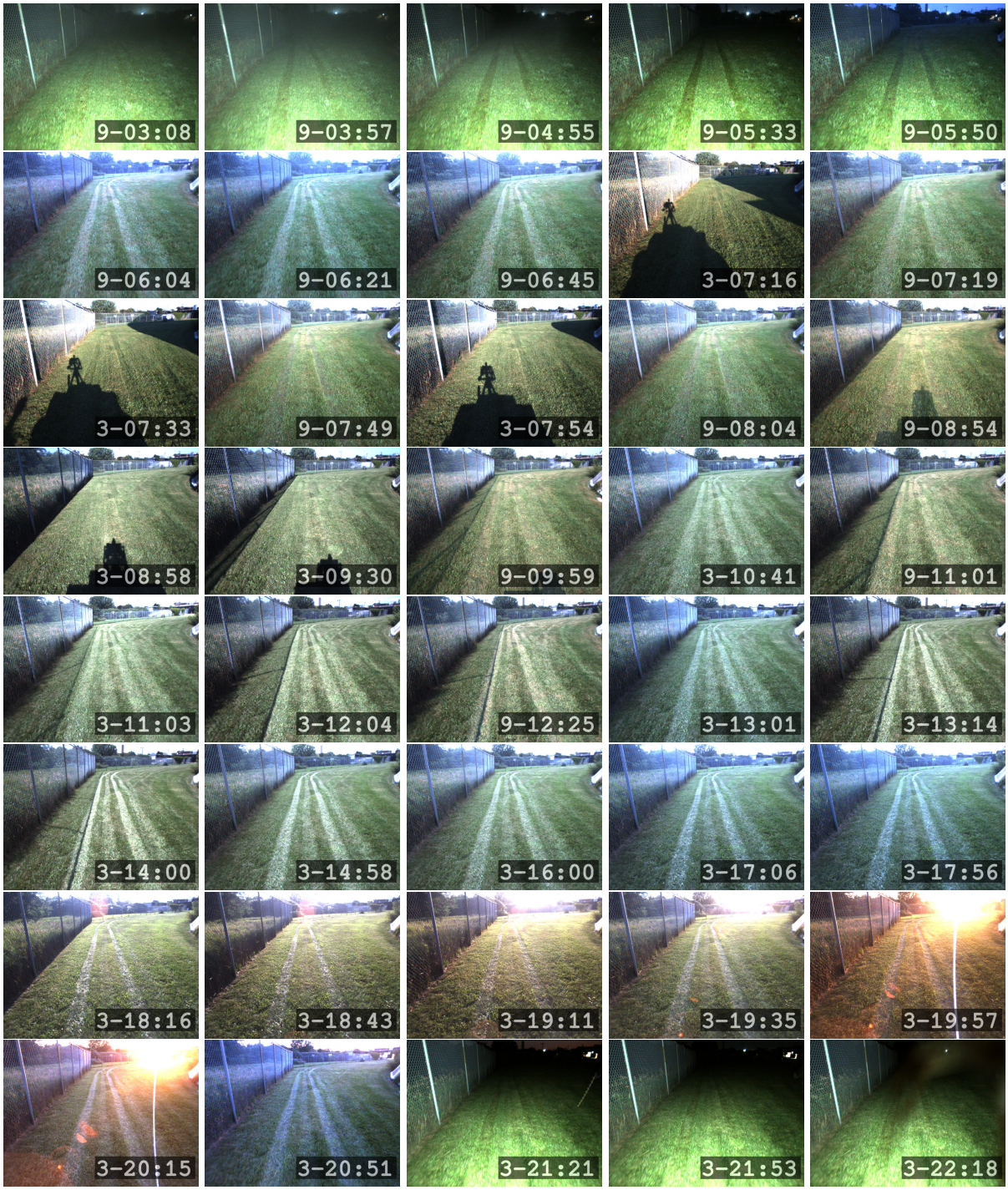


Figure 6.15: Images taken during closed-loop testing for the lighting-change experiment labeled with collection day and time (dd-hh:mm). The images are ordered according to the time of day when they were collected as we are most interested in the lighting change, and the weather did not vary significantly.

data and the experiment. The training data, however, was collected five years prior to our experiment leading to some physical changes in the environment.

The new path is approximately 265 metres long and was taught at 12:14 on August 2nd, 2021. We repeated the path on August 3rd and 9th covering lighting conditions between 03:00 and 22.30. The

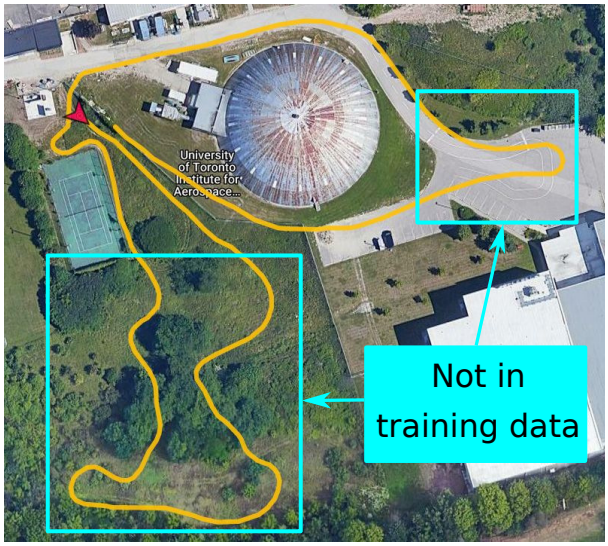


Figure 6.16: Overhead view of the path for the first generalization experiment overlaid on a Google Earth image. The path was taught on August 14th. The top left area of the path is close to the UTIAS Multiseason path, which can be seen with the tennis court as reference in Figure 6.11. Most of the path around the Mars Dome is close to the path from the UTIAS In-The-Dark dataset. The rectangles indicate the parts of the path that are outside of the training data.

experiment was conducted over two days to be able to cover the large range of times, while keeping the robot battery charged and handling transfer of the collected data to a server. Images from the robot camera from each repeat in Figure 6.15 show the variety in appearance. We also provide some images of the robot that show the different appearance conditions along the path in Figure 6.14.

6.5.5 Experiment 3: Generalization

In the second closed-loop experiment, we aim to investigate whether the learned features will generalize to areas that were not seen in the training data. That is, we want to make sure that the network is not just remembering locations along the paths from the training data. If the features generalize, this would provide an advantage over MEL beyond removing the need for intermediate bridging experiences. In MEL, the experiences collected over time can only be used for that specific path. If we wish to teach and repeat a new path, in the same area or elsewhere, we need collect a new set of experiences.

For the generalization experiment, we teach three different paths and will refer to the different tests as generalization I, II, and III. For all three tests. we use the network trained using both the UTIAS Multiseason and UTIAS In-The-Dark datasets so that we can follow paths in both off-road and on-road areas. The network training is explained in detail in Section 6.4.2.

Generalization I

For the first test we taught an approximately 760 metre long path on August 14th at 13:26, which includes both on-road and off-road driving. As shown in Figure 6.16, we add two new areas that are not in the training data. The larger of the two new sections is driven in an off-road area with more vegetation and taller grass than what was seen in the most similar UTIAS Multiseason training dataset. On three occasions, the robot traverses through areas with trees and bushes close on either side. Moreover, the

Figure 6.17: The top row shows images from the last run in the UTIAS Multiseason dataset (used for training our model) collected on May 11th, 2017. The middle row contain images from the teach path collected on August 14th, 2021, used for our generalization and seasonal closed-loop experiments, while the bottom row is from a repeat on October 28th, 2021. The leftmost and rightmost images from all rows are collected in approximately the same two locations. The rest of the images are from the same locations on the last two rows, but differ from the training data as we do not have spatial overlap. We can see that, although all runs are collected off road, both the seasonal difference and variation in area contributes to a significant visual gap between the training and experimental data.

experiment is conducted in August, while the UTIAS Multiseason dataset only contains data until May. This means that none of our training data exactly resembles this off-road summer condition. Figure 6.17 shows example images along the new off-road section to illustrate the challenging environment and how it is different from the training data. It contains spring image examples from the UTIAS Multiseason dataset for comparison.

The second part of the path that runs through a new and unknown area, is smaller and contained in a parking lot. This area more closely resembles the environment in the UTIAS In-The-Dark training dataset. It is also more closely related in seasonal appearance, since the UTIAS In-The-Dark data was collected in summer. Finally, the training datasets were collected in summer 2016 and winter to spring 2017, while we conduct our experiment several years later in August 2021.

As stated above, the first generalization path was taught on August 14th. It was repeated on August 15th, 16th, and 20th covering lighting change between 04:00 and 21:00. The weather was mostly sunny on all days, with some intermittent clouds. We used several days to be able to cover a complete range of lighting changes while keeping the robot battery charged and handling data transfer after each repeat. Images from the robot for each repeat are displayed in Figure 6.18, while we include pictures of the robot during autonomous traversal under different appearance conditions in Figure 6.19.

Generalization II, III

For the last two test of the generalization experiments, we teach off-road in a field (generalization II) and on-road in a parking lot (generalization III). These new paths have no spatial overlap at all with the UTIAS Multiseason or UTIAS In-The-Dark training data. We used the same network for feature

extraction as in the first generalization experiment. We taught an approximately 275 metre path in the field at 09:52. on November 11th, 2021. An overhead view of the path can be seen in Figure 6.20. The second path is approximately 220 m and was taught at 11:34 on November 11th, 2021, in a parking lot, see Figure 6.20.

6.5.6 Experiment 4: Seasonal Change

After completing the experiments to test robustness to lighting change, we also want to see how the learned features perform as the weather and environment changes over a longer time period. Hence, we keep repeating the 760 metre path from the first generalization experiment, taught on August 14th, into autumn. See Figure 6.16 for the overhead view. After a break at the end of August, we start repeating the path on September 10th and continue until November 19th. The repeats are collected at different

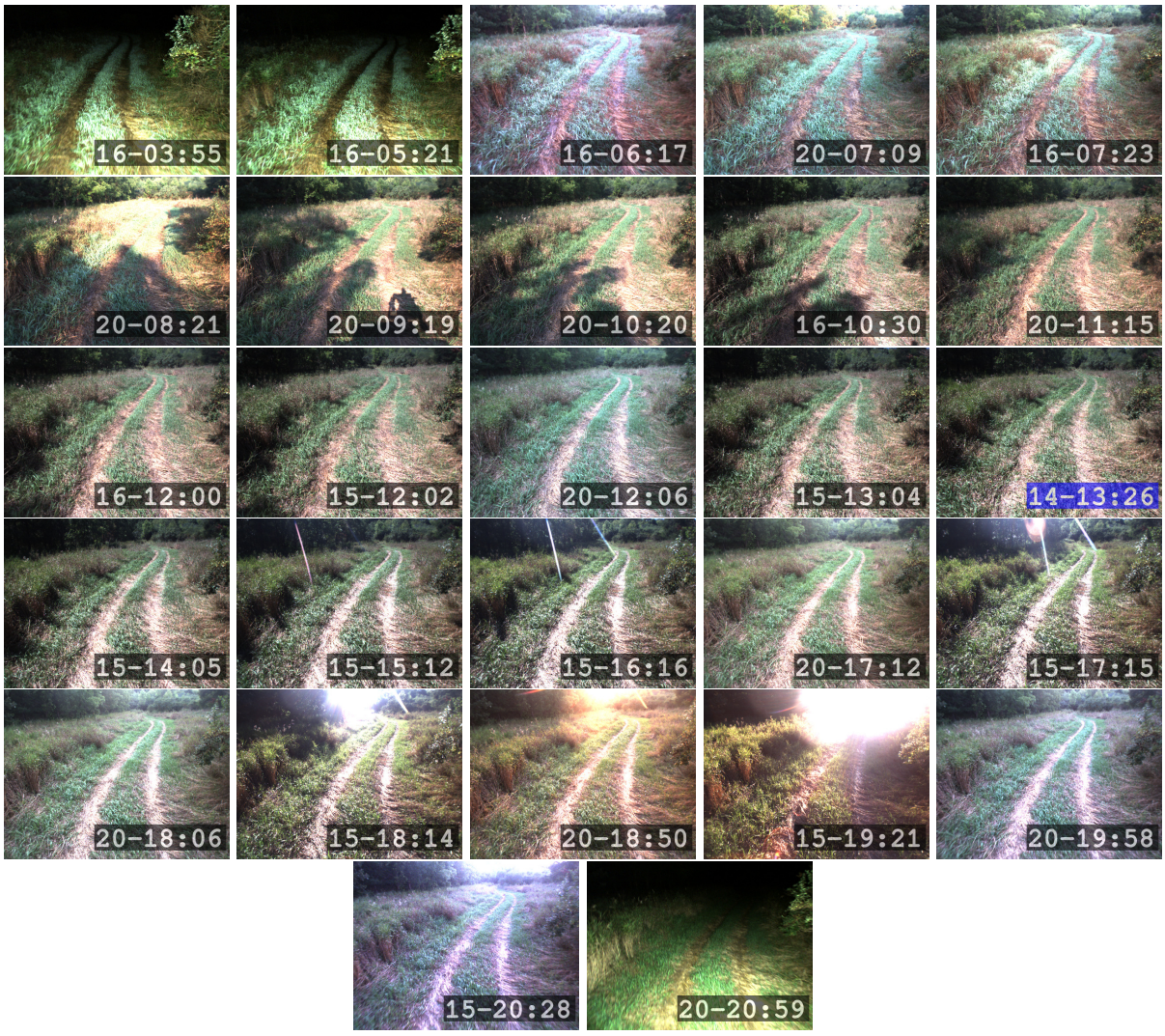


Figure 6.18: Images taken during closed-loop testing for the first generalization experiment labeled with collection day and time (dd-hh:mm). The image labeled in blue is from the teach pass for reference. The images are taken in the new off-road area not contained in the feature training dataset. The images are ordered according to the time of day when they were collected as we are most interested in the lighting change, and the weather did not vary significantly.

Figure 6.19: The robot autonomously repeating the path for the first generalization experiment under different appearance conditions.

Figure 6.20: Overhead view of the paths for the second (left) and third (right) generalization experiments overlaid on a Google Earth image. The path on the left is taught in a field, while the other is driven in a parking lot. Neither path has any spatial overlap with the training data.

Figure 6.21: The robot autonomously repeating the paths for the second (top row) and third (bottom row) generalization experiments under different appearance conditions.

times of day, hence, lighting and seasonal change will both influence visual appearance .

We can note that since the UTIAS Multiseason dataset was collected from January to May, we do not have any specific off-road training data for summer or autumn. Some examples of the difference between the training data and the images seen during autonomous traversal are illustrated in Figure 6.17. The same is true for the on-road area, as the UTIAS In-The-Dark training data was only gathered over two days in summer. The road and surrounding buildings, however, make for an easier matching task, while the off-road area is subject to larger structural and visual change between seasons. We include pictures of the robot during autonomous repeat at different locations along the path under different seasonal conditions, see Figure 6.24. Images from the robot camera for the different repeats of the path during September, October, and November can bee seen in Figures 6.25, 6.26, and 6.27, respectively.

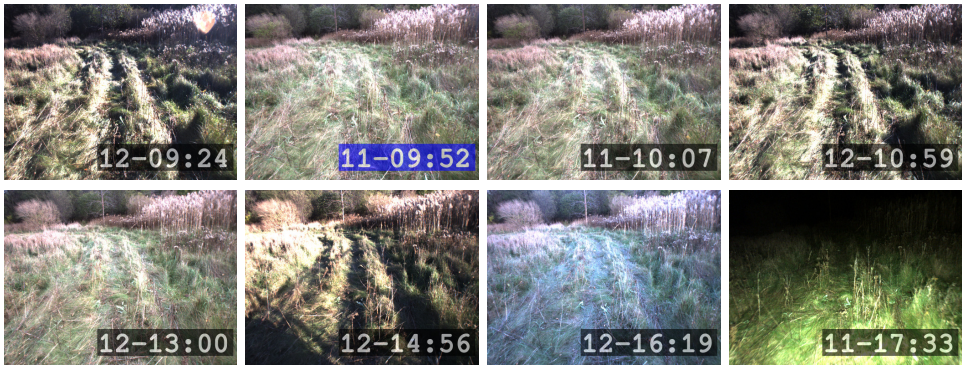


Figure 6.22: Images taken during closed-loop testing for the second generalization test labeled with collection day and time (dd-hh:mm). The image labeled in blue is from the teach pass for reference, and the images are ordered by the time of day when they were collected.

Figure 6.23: Images taken during closed-loop testing for the third generalization test labeled with collection day and time (dd-hh:mm). The image labeled in blue is from the teach pass for reference, and the images are ordered by the time of day when they were collected.

Figure 6.24: The robot repeating the path for the seasonal experiment under different appearance conditions. The coloured frames indicate pictures taken in the same location at different times.

6.6 Evaluation Metrics

In order to evaluate the various experiments we have outlined, we use a set of metrics that measure quantities such as feature matches, the distance we rely on dead-reckoning in case of localization failures, as well as the path-following autonomy rate and path-tracking error. For each experiment, we will provide a subset of the metrics depending on the data available.

Figure 6.25: Images taken in September during closed-loop testing for the seasonal change experiment labeled with collection date and time (dd.mm-hh:mm). The image from the teach sequence is included for reference and marked with a blue label.

6.6.1 Feature Matches

A feature-based approach to localization relies on matching keypoints between two images using their associated descriptors. After the initial feature matching, RANSAC is used to find the inliers, which are subsequently passed on to pose estimation. Analysis of the feature matching inliers are provided for all experiments in sections 6.7.1, 6.7.2, 6.7.3, and 6.7.4.

Inlier Count

Having a high number of inliers is beneficial for accurate and robust localization. Hence, we consider the number of inliers as a useful indicator of the learned features' performance in long-term localization. In our experiments, six inliers are set as the threshold for successful localization. If the number of inliers is lower, we rely on dead-reckoning from VO until localization is reestablished. We aggregate the number of inliers for all keyframes in each autonomous traverse of a path. This allows us to compare the number of inliers for different repeats, i.e., different appearance conditions. Since some areas might be more challenging than others, we will also look at the inlier count along different parts of the path.

A high number of inliers is not a perfect predictor for pose estimation success. For instance, many matches can be concentrated in one part of the image or we could get a high number of matches far away from the robot and fewer matches with small depth. Localization produces an uncertain pose estimate, which is passed on to other components such as path-tracking control. Path-following performance can therefore also be affected by issues down stream from localization. Inlier count is therefore a useful proxy for localization performance, but not a perfect gauge for overall path-following performance on its own. We include more metrics in the following to help assess performance in different ways.

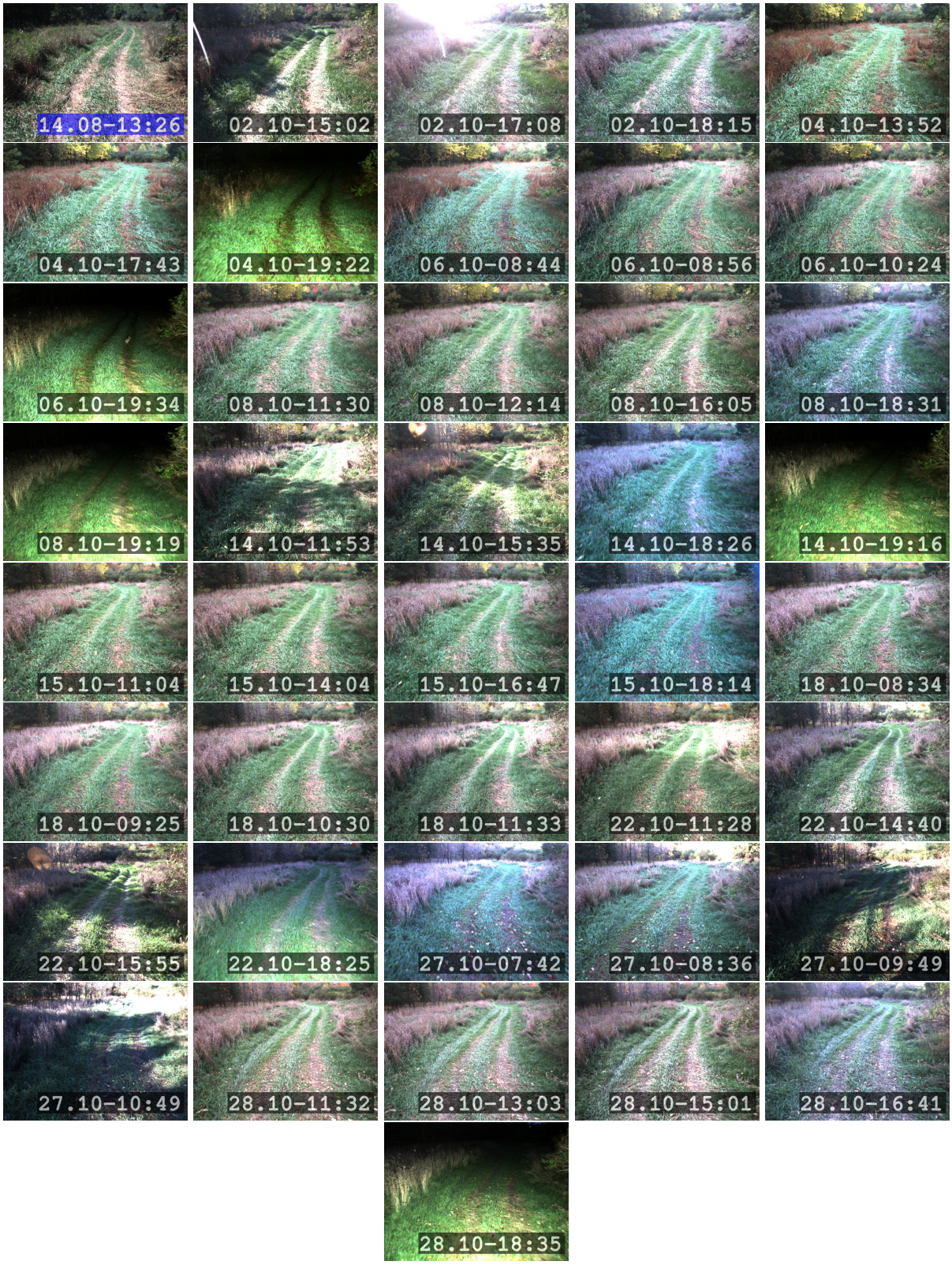


Figure 6.26: Images taken in October during closed-loop testing for the seasonal change experiment labeled with collection date and time (dd.mm-hh:mm). The image from the teach sequence is included for reference and marked with a blue label.

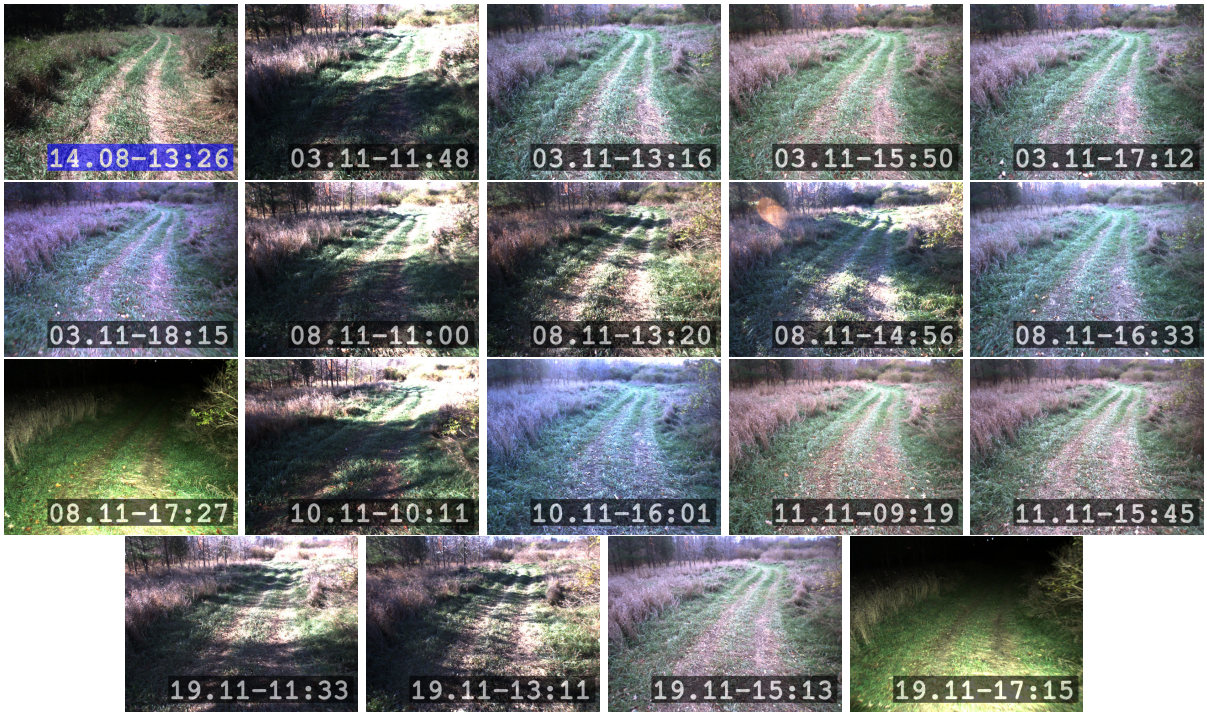


Figure 6.27: Images taken in November during closed-loop testing for the seasonal change experiment labeled with collection date and time (dd.mm-hh:mm). The image from the teach sequence is included for reference and marked with a blue label.

Inlier Spatial Distribution

The number of inliers alone, is not necessarily a perfect indicator for localization performance. If we successfully match a lot of features, but all matches are concentrated in one part of the image, this could lead to poor pose estimates. It is beneficial to have feature matching inliers distributed across images. In particular, having feature matches close to the robot (smaller depth) is important for a good translation estimate, while features farther away around the horizon (larger depth) are useful for constraining heading. We analyze the distribution of the features for different locations along a path and over time as appearance changes.

6.6.2 Distance on Dead-Reckoning

When the number of matched feature inliers drops below six, we consider localization to have failed for the given frame. In this case, VT&R will rely on dead-reckoning with VO until localization is reestablished. If the robot drives on dead-reckoning for more than 20 metres, the robot will stop as continuing is not considered safe. Measuring the distance the robot drives using VO for a given repeat gives us an indication of the robustness of localization with learned features and the resulting path-following performance. We analyze the distance driven on VO for all of the closed-loop experiments in Sections 6.7.2, 6.7.3, and 6.7.4.

Table 6.2: Comparison of the median number of inliers for SuperPoint, D2-Net, R2D2, and our method for localization of the runs shown in Figure 6.12. The rows list the condition used to teach by date (dd.mm), while the columns correspond to the repeats. As localization becomes more challenging, our method keeps a high number of inliers, while the other methods deteriorate.

		Repeat																													
		31/01				02.02				10.02				14.02				27.02				10.04				03.05			
		SP	D2	R2D2	Our	SP	D2	R2D2	Our	SP	D2	R2D2	Our	SP	D2	R2D2	Our	SP	D2	R2D2	Our	SP	D2	R2D2	Our	SP	D2	R2D2	Our
Teach	31.01					70	272	**613**	311	35	76	**284**	230	4	2	4	**112**	4	46	17	**201**	4	37	7	**186**	4	39	9	**154**
	02.02	69	263	**606**	310					77	154	**319**	248	16	17	10	**124**	11	47	21	**223**	6	36	5	**206**	8	34	5	**166**
	10.02	31	76	**283**	230	79	155	**313**	248					24	22	30	**117**	3	10	3	**173**	3	8	2	**161**	2	8	2	**131**
	14.02	5	2	3	**114**	19	31	15	**134**	30	31	45	**128**					2	2	3	**121**	1	0	0	**111**	1	1	0	**102**
	27.02	4	45	15	**201**	12	47	22	**223**	4	13	5	**171**	2	2	2	**114**					15	136	104	**257**	13	75	25	**196**
	10.04	3	34	7	**183**	7	31	4	**206**	3	7	1	**158**	1	0	0	**106**	13	136	119	**257**					15	56	33	**208**
	03.05	4	36	9	**153**	8	38	4	**166**	3	10	2	**131**	1	2	0	**98**	14	74	25	**197**	15	55	33	**207**				

6.6.3 Autonomy Rate

In the pipeline, from feature matching, through localization, and to path tracking, the final indication of closed-loop performance is whether the robot can successfully follow the path. For all the closed-loop experiments in Sections 6.7.2, 6.7.3, and 6.7.4, we record any path-following failures during autonomous operation, where the robot has to be stopped and path repetition has to be restarted.

6.6.4 Path-Tracking Error

In order to assess the autonomous path-following accuracy, we use a Navtech GPS antenna on the Clearpath Grizzly robot together with a separate base-station that provide accurate RTK GPS pose estimates. Since, ultimately, we care about the robot's ability to follow a path, we measure the path-tracking error and do not compute localization error. We record the GPS poses during the teach run as 3D positions without orientation. For each GPS pose in a repeat, we find the closest point on the teach run and compute the distance between them. We compute the RMSE for each repeat individually and for all repeats as a whole. We collected RTK GPS poses for the lighting-change experiment in Section 6.7.2, but not for the remaining generalization and seasonal experiments. For the first generalization experiment and seasonal experiment, we drive in an area where tree cover blocks the GPS signal and, moreover, we often end up too far from the RTK base station, resulting in inaccurate GPS pose estimates. Fore the second and third generalization experiment we did not have the time to set up the RTK GPS.

6.7 Results

In this section we present and analyze the results of using learned features for long-term localization in the experiments from Section 6.5.6. The learned features let us autonomously repeat paths across all lighting conditions, including when driving in areas outside the training data, with a 100% autonomy rate. We are able to repeat paths across seasonal change from August to November, though we get some path-following failures, especially when seasonal change is compounded with large lighting change.

6.7.1 Experiment 1: Offline Comparison

We compared our method to state-of-the-art learned features on localization in PyTorch. Keypoints, descriptors, and scores were extracted using the code provided by the authors, and the same sparse nearest neighbour feature matching was used for all methods for fairness. For D2-Net and R2D2, we

used multi-scale detection to get the best possible results. All conditions from Figure 6.12 were localized against each other. We take one run and use it as a teach while using the rest of the runs as repeats. This is repeated for all conditions.

Since our target application runs on a laptop in real-time, memory consumption and speed matter in addition to feature matching performance. R2D2 (1531 MB) and our method (2443 MB) used the lowest memory, whereas SuperPoint (5305MB) and D2-Net (9521MB) would be too large to fit on the 4 GB laptop GPU. Our method was the fastest with 14.8 ms for feature extraction, while SuperPoint used 37.3 ms and R2D2 used 38.7 ms. D2-Net was the slowest with 477.4 ms, though it would likely be faster without multi-scale detection. Our method detects a fixed number of keypoints in one shot, while both SuperPoint and R2D2 rely on additional NMS to pick a subset of the keypoints.

Feature Inlier Count

We evaluate the median number of feature matching inliers and provide the results in Table 6.2. R2D2 gets the best result localizing the first three conditions to each other. Our method, however, is the only one with a high number of inliers across all experiments, while the others deteriorate as seasonal change increases. Table 6.2 shows that the run from 14.02 with snow and high brightness is the most difficult. Since we test on off-road data from the UTIAS Multiseason dataset, the experiment has limited usefulness when comparing matching performance. We did this test as a sanity check as our method trained on relevant data should outperform the others.

Conclusion

Our method gets a high number of feature matching inliers across all seasonal conditions, while other methods struggle as appearance change increases. This is as expected given that our method has more relevant training data. Moreover, our method has the fasted inference time, while R2D2 and our method both have a low enough memory consumption to fit on the GPU of the laptop we use to run VT&R.

6.7.2 Experiment 2: Lighting Change

For this experiment, we ran VT&R with learned features for localization in closed loop. We taught a new path similar to the one from the UTIAS In-The-Dark dataset on August 2nd at 12:14 and repeated it autonomously on August 3rd and 9th spanning lighting change between 03:00 and 22:30. An overview of the teach and all repeat runs with time of day, appearance conditions, length of the repeats, and more, is provided in Table 6.3. In summary, we trained learned features with data from five years prior and were able to localize the repeat frames directly to the teach without the need for any intermediate experiences. The robot repeated the path accurately with a 100% autonomy rate for all conditions including driving after dark with headlights and during low sun causing long shadows and challenging sun flares. In the following, we study the results in more detail using the metrics from Section 6.6.

Feature Inlier Count

We start by looking at the number of learned feature matching inliers, which is determined by RANSAC. Having a high number of feature matching inliers means that we have a high number of points to pass to the pose estimation. We visualize the inliers in a few different ways. The repeats were conducted over two days, on which the weather was similar with sun and some occasional clouds. In such a short

Table 6.3: Overview of the teach and each repeat run for the lighting-change experiment. The teach run has ID 0.

	ID	Start Time	Duration [hh:mm]	Distance [m]	Dist. VO [m]	Failures [count]	RMSE [m]	Conditions
Day	0	02.08.21 12:14	-	265.5	-	-	-	Sun, shadows
	1	03.08.21 07:16	00:08	266.8	0.00	0	0.043	Sun flare, long shadows
Sunrise	2	03.08.21 07:33	00:08	266.5	0.00	0	0.045	Sun flare, long shadows
	3	03.08.21 07:54	00:09	266.1	0.00	0	0.042	Sun flare, long shadows
	4	03.08.21 08:58	00:08	265.1	0.00	0	0.041	Light sun flare, long shadows
	5	03.08.21 09:30	00:09	265.1	0.00	0	0.042	Sun, long shadows
	6	03.08.21 10:41	00:08	267.0	0.00	0	-	Mostly cloudy
	7	03.08.21 11:03	00:08	266.4	0.00	0	-	Sun, cloud, shadows
	8	03.08.21 12:04	00:08	266.5	0.00	0	-	Sun, cloud, shadows
	9	03.08.21 13:01	00:08	267.0	0.00	0	-	Sun, cloud, shadows
Day	10	03.08.21 13:14	00:08	265.9	0.00	0	0.043	Sun, shadows
	11	03.08.21 14:00	00:08	266.2	0.00	0	0.044	Sun, shadows
	12	03.08.21 14:58	00:08	267.1	0.00	0	0.050	Sun, cloud, shadows
	13	03.08.21 16:00	00:08	267.6	0.00	0	0.043	Mostly cloudy
	14	03.08.21 17:06	00:09	268.6	0.00	0	0.070	Sun, cloud, shadows
	15	03.08.21 17:56	00:08	268.0	0.00	0	0.048	Sun, long shadows
	16	03.08.21 18:16	00:08	268.9	0.00	0	0.052	Sun flare, long shadows
	17	03.08.21 18:43	00:08	269.2	0.00	0	0.043	Sun flare, long shadows
	18	03.08.21 19:11	00:08	269.0	0.00	0	0.048	Sun flare, long shadows
	19	03.08.21 19:35	00:08	268.0	0.00	0	0.044	Sun flare, long shadows
Sunset	20	03.08.21 19:57	00:08	268.1	0.00	0	0.046	Sun flare, long shadows
	21	03.08.21 20:15	00:08	267.7	0.00	0	0.068	Sun flare
	22	03.08.21 20:51	00:09	267.5	0.00	0	0.045	Dusk
	23	03.08.21 21:21	00:09	263.4	0.00	0	0.056	Dark
	24	03.08.21 21:53	00:08	263.2	0.00	0	0.042	Dark
	25	03.08.21 22:18	00:08	263.0	0.37	0	0.057	Dark
Night	26	09.08.21 03:08	00:08	263.1	0.00	0	0.043	Dark, foggy
	27	09.08.21 03:57	00:08	263.5	0.00	0	0.061	Dark, foggy
	28	09.08.21 04:55	00:08	263.1	0.00	0	0.039	Dark, foggy
	29	09.08.21 05:33	00:08	263.4	0.00	0	0.044	Dark
	30	09.08.21 05:50	00:08	263.6	0.00	0	0.042	Dawn
	31	09.08.21 06:04	00:09	266.4	0.00	0	0.068	Dawn
Sunrise	32	09.08.21 06:21	00:09	266.5	0.00	0	0.052	Cloudy
	33	09.08.21 06:45	00:08	266.4	0.00	0	0.048	Cloudy
	34	09.08.21 07:19	00:09	266.5	0.00	0	0.047	Cloudy
	35	09.08.21 07:49	00:09	266.6	0.00	0	0.056	Sun, cloud
	36	09.08.21 08:04	00:08	266.7	0.00	0	0.051	Sun, cloud
	37	09.08.21 08:54	00:08	266.7	0.00	0	0.062	Sun, long shadows
Day	38	09.08.21 09:59	00:08	266.5	0.00	0	0.061	Sun, long shadows
	39	09.08.21 11:01	00:09	266.7	0.00	0	0.068	Sun, shadows
	40	09.08.21 12:25	00:08	266.0	0.00	0	0.052	Sun, shadows
Total	41		05:30	10.9 km	0.37	0		

time, the physical environment did not change much (as it would in a longer seasonal experiment), and so the dominant change between runs is the lighting variation. For this reason, we plot our data for the

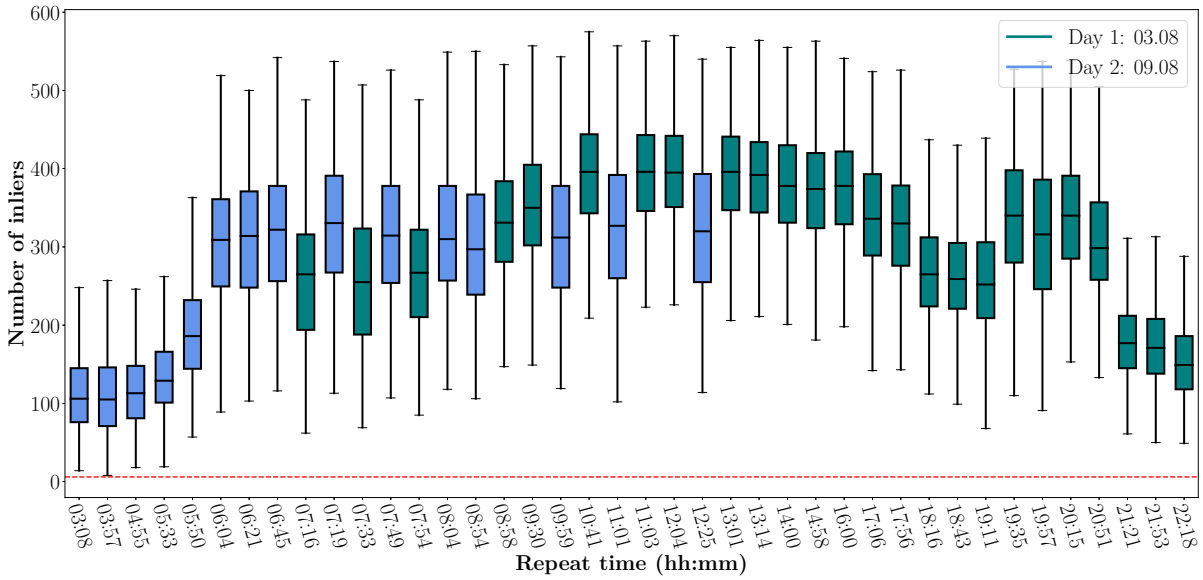


Figure 6.28: A box plot of the number of inlier feature matches for localization for every run of the lighting-change experiment listed in Table 6.3. The path was taught at 12:14. The horizontal axis shows the time of day when the runs were traversed, and we colour the boxes according to the day the run was completed. The red line shows the localization failure threshold of six inliers. The highest number of inliers occur during the day and the lowest when it is dark. Dips in the number of inliers also occur at sunrise and sunset.

experiment according to the time of day when it was collected, instead of the date and time. First, we provide a box plot of the number of inliers for each run of the experiment in Figure 6.28. We also include Figure 6.29, which displays the median number of inliers together with the 0.25 and 0.75 quantiles as shaded regions. As opposed to the box plot, this visualization allows us to distribute the runs along the horizontal axis according to the time of day when they were collected.

From the two plots, we see that the highest number of inliers occurs during the day, when the appearance is most similar to the teach run, which was collected at 12:14. The number of inliers are the lowest when driving during the dark, either at night (21:21, 21:53, and 22:18) or during the early morning (03:08, 03:57, 04:55, and 05:33). Due to fog, the number of inliers is a bit lower for the early morning runs compared to the night runs. This is a new condition not seen in the UTIAS In-The-Dark training data. Finally, we also see that the number of inliers dip for some runs in the morning and evening. This is due to low sun during sunrise and sunset, which causes long shadows and sun flares in the images seen by the robot. Overall, we retain a high number of inliers across all runs providing robust localization despite large appearance change.

The previous plots showed the median and quantile results for each repeat of the path. In VT&R only a subset of the live images frame from the camera are used for localization. Going forward we refer them as keyframes. We can get more detail on the feature inlier distribution within each run by looking at the Cumulative Distribution Function (CDF) of live keyframes over the range of number of inliers. In Figure 6.30, we plot the CDF for every repeat of the experiment. Since there are many repeats, we colour each line by the difference in hours between the time of day of the autonomous traverse (not considering the date) and the initial teach run. Consistently with the previous plots, we see that the runs from early morning (dark blue) and night (strong yellow) have the lowest number of inliers. We

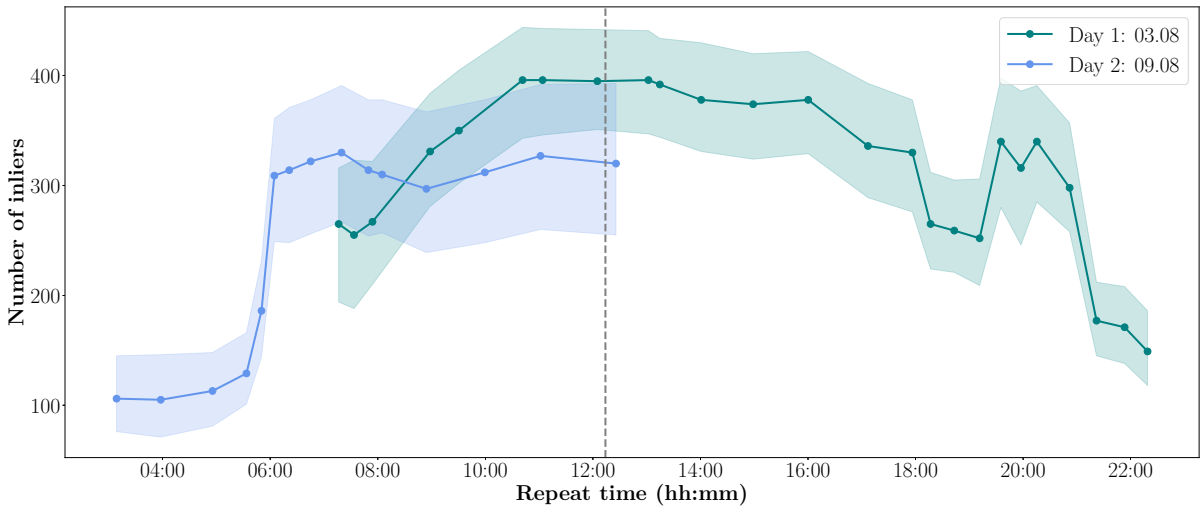


Figure 6.29: A plot of the median number of inlier feature matches for all runs of the lighting-change experiment listed in Table 6.3. The shaded regions represent the 0.25 and 0.75 quantiles. The horizontal axis lists the time of day when the runs were traversed, and we colour the lines according to the day the run was completed. The gray line marks when the path was taught (12:14). The highest number of inliers occur during the day and the lowest when it is dark. Dips in the number of inliers also occur at sunrise and sunset.

also see that the number of inliers is consistently high across all runs and that all runs have 100 inliers or more for approximately at least 90% of their keyframes.

Finally, we are also interested in seeing how the number of matched feature inliers may vary along different parts of the path. Are there any parts of the path where localizing may prove more difficult than others? To this end, we plot the median number of inliers across all runs of the experiment for each vertex, or keyframe, in the map. We also include the 0.25 and 0.75 quantiles. Figure 6.31 shows the plot, where the path is coloured according to the inlier count. The number of inliers is high across the whole path, but consistently a little lower in the bottom right quarter of the path. This coincides with an area, where the robot drives on lawn grass instead of asphalt, and at the same time the camera views an area with somewhat fewer structures.

To summarize, we see that the learned features match well with a high number of inliers despite drastic lighting change. The features are proving robust to challenging conditions such as driving during the dark (including fog) with headlights and low sun at sunrise and sunset causing large shadows and sun flares.

Feature Inlier Distribution

A high number of matched feature inliers alone is not necessarily sufficient for robust pose estimation. We can imagine getting many matches is one part of the image. Because our approach divides the image into size 16×16 windows and detects one feature per window, a high number of inliers is a fairly good indication that features are spread across a substantial part of the image. Even so, it is useful to understand how the distribution of matched features behaves for different locations along the path and for different lighting conditions.

To this end, we provide qualitative results that visualize feature matches for selected map vertices

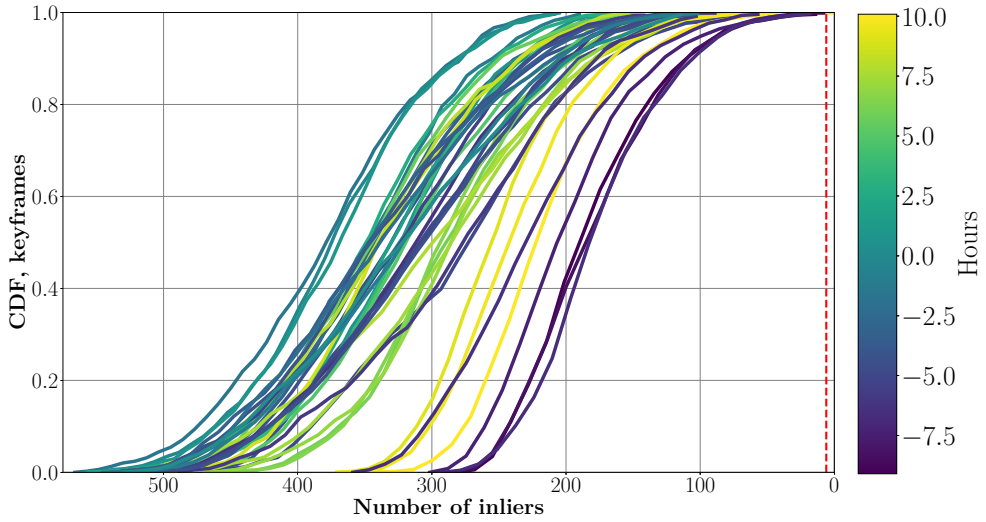


Figure 6.30: The CDF of keyframes with a given number of inliers for each repeat in the lighting change experiment. The red vertical line shows the localization failure threshold of six inliers. The lines are coloured according to the time of day that they were collected, given by the number of hours before or after the teach was collected (12:14 pm). As expected, the runs collected at the beginning of the day and at night (in dark blue and strong yellow) have the lowest number of inliers.

along the path. We aggregate the data for four different times of day - sunrise, day, sunset, and night - as indicated in Table 6.3. The results can be seen in Figure 6.32. For a vertex in the map, we find all the runs that have localized to this vertex for a particular time of day (sunrise, day, sunset, or night). Every map vertex may not have every live run localized to it, but we pick vertices, where at least five live keyframes have localized for each of sunrise, day, sunset, and night. We divide each image into size 16 $\times$ 16 bins and record the image coordinates of all matched feature inliers. We count how many keypoints fall into each bin and divide by the number of repeat runs that have localized to this keyframe for each time of day group. If the bin has final value 1.0, that means a keypoint was matched in this part of the image for each run that was considered for that condition, for example each night run.

From Figure 6.32, we observe the highest number of feature matches during the day and the lowest during the night, which is consistent with the inlier count in Section 6.7.2. When the number of matched features is low, for the more challenging lighting conditions, we tend to loose more keypoints close to the robot in the lower part of the image, while keypoints detected closer to the horizon seem more robust. This could result in the translation estimates becoming more uncertain. However, we retain a high number of inliers throughout this experiment and only run into localization failure for three frames during one night-time run (see Table 6.3). We can also see some qualitative difference in feature distribution between locations along the path. In particular, it looks like matching features close to the robot when driving on lawn grass is more difficult than when driving on asphalt.

Distance on Dead-reckoning

When the number of feature inliers fall below six, VT&R relies on dead-reckoning from VO to keep driving until localization is reestablished. As can be seen in Table 6.3, for this experiment, we only had localization failure after dark during run 25 starting at 22:18 at night. Localization failed for three frames and we relied on VO for a total of 0.37 m. Figure 6.33 shows the CDF of live keyframes over the

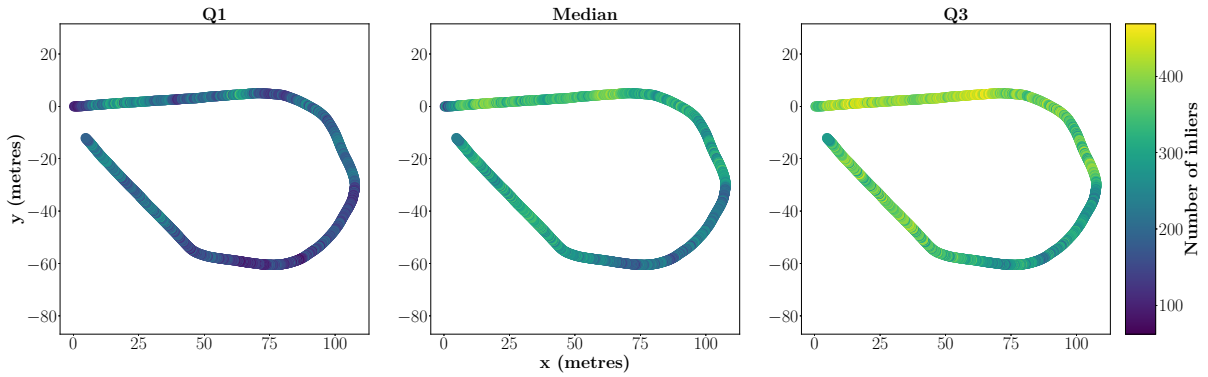


Figure 6.31: The plot shows the median number of inliers across all runs for each map vertex along the path, as well as the 0.25 (Q1) and 0.75 quantiles (Q3). The area in the lower right quadrant, where the robot drives on lawn grass while facing few structures, proves the most challenging for feature matching.

distance driven on VO. The fact that we only rely on VO for a very small number of frames is another indication that the learned features provide robust localization across challenging lighting change.

Path-tracking Error

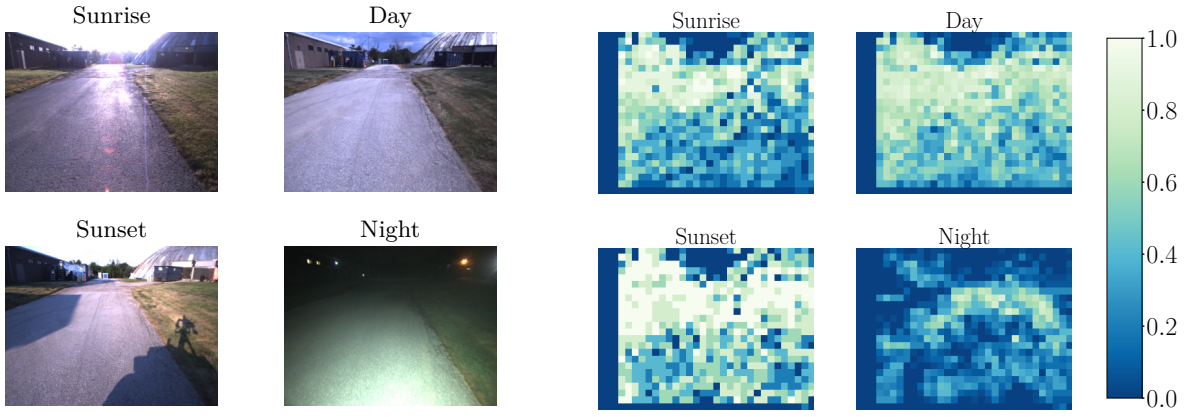
We collected the ground truth pose using RTK GPS for most runs, except a few, where we were unable to establish a GPS fix with sufficiently low pose covariance in time to complete the autonomous traverse of the path. For a few runs, the pose covariance of the ground truth deteriorated for a short stretch along the path. This happened sometimes when the robot drove between the Mars Dome building and a parked trailer temporarily blocking the signal either to the base station or losing connection with one or more satellites. We report RMSE for each individual run, where ground truth was collected. On runs, where pose covariance was temporarily high, we exclude these measurement points from the RMSE. The RMSE for each run are listed in Table 6.3. The RMSE computed from all runs taken together is 0.049 m and the highest RMSE for a single run is 0.070 m. We get low path tracking RMSE across all runs, which shows that we are able to follow the path accurately across a full range of lighting conditions.

Autonomy Rate

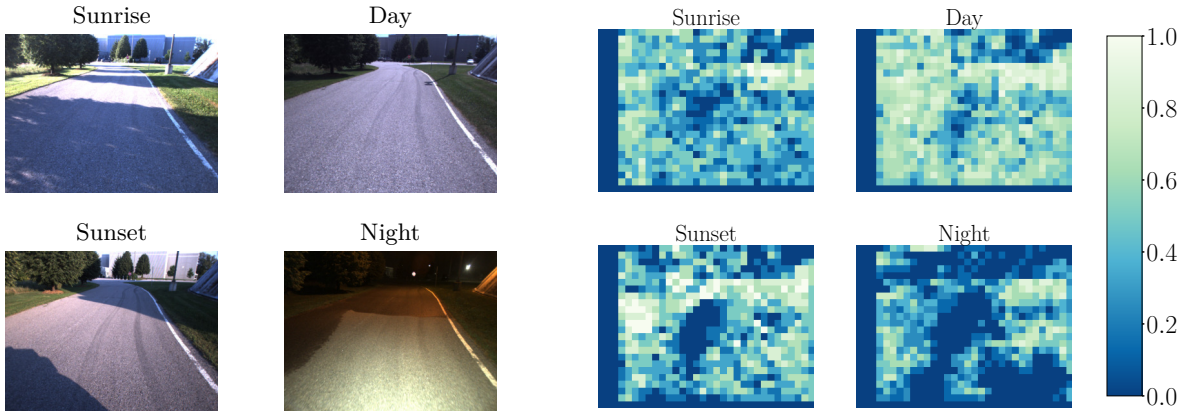
For this experiment autonomous path following succeeded for all runs without any manual intervention and the experiment was completed with 100% autonomy rate, see Table 6.3.

Conclusion

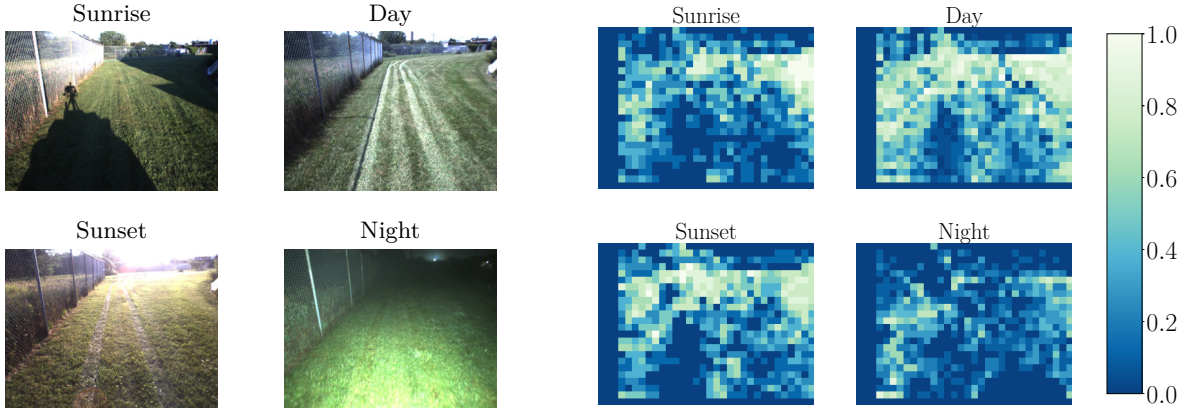
In this closed-loop experiment, we saw that the learned features provided a high number of feature matching inliers for localization across a full range of lighting change. We trained a network with data from 31 hours of lighting change from summer 2016 and were able to teach and repeat a path in the same area five years later. The robot followed the path accurately with a low path-tracking error and only encountered a very small number of localization failures, where it had to rely on dead-reckoning for navigation. We have shown that the learned features work well in a closed-loop system when the spatial area and appearance conditions are similar to those of the training data. In our next experiment, we teach a more challenging path to investigate the features' ability to generalize beyond the feature training dataset.



(a) Driving on the road in and area with buildings.



(b) Driving on the road in and area with buildings.



(c) Driving in on grass in an area with buildings.

Figure 6.32: Distribution of feature matching inliers for one map vertex along the path. The heat map counts the number of inliers that fall into the 16×16 windows for all live keyframes localized to the given map vertex. We normalize by the number of localized live keyframes. On the left we show one example live image localized to the map vertex for each time of day. When comparing (a) and (b) to (c), we see that we get somewhat fewer features on the ground close to the robot when driving on grass.

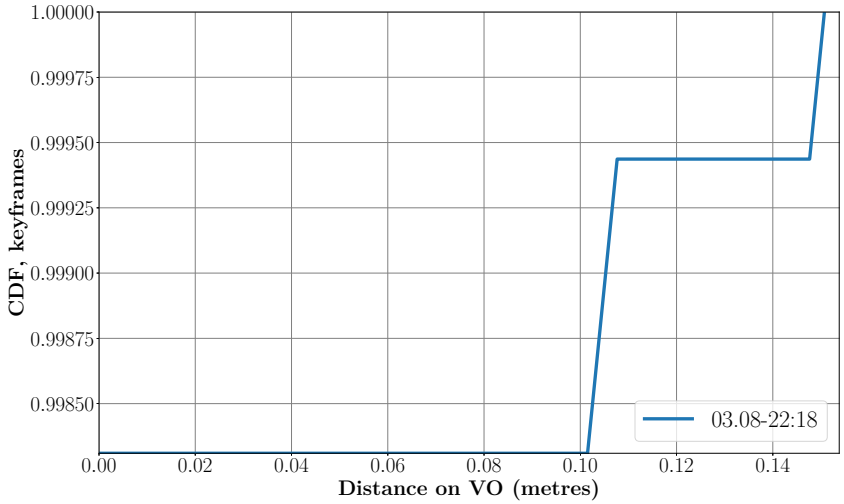


Figure 6.33: A CDF over live keyframes for the distance VT&R relies on VO to compensate for localization failures. The horizontal axis lists the distance VT&R has relied on dead-reckoning for a given live keyframe.

6.7.3 Experiment 3: Generalization

So far, we have shown that the learned features successfully facilitate long-term localization directly to the taught map across a full range of lighting change. In this experiment we investigate whether the features can generalize outside the training data both spatially and in appearance. Ideally, the network should not just memorize locations along the path or the exact appearance conditions given in the training data. To this end, we completed three different experiments that test autonomous path following in new areas, as described in detail in Section 6.5.5.

We taught a longer, approximately 760 metre, path on August 14th at 01:26 p.m, that covers some of the on-road area from the UTIAS In-The-Dark training dataset and some of the off-road area from the UTIAS Multiseason training dataset. See Figure 6.16 for an aerial view of the path. In addition, it covers a new off-road area with more tall grass, trees, and other vegetation than seen in the training data. Moreover, this experiment is conducted in August, whereas the UTIAS Multiseason dataset was collected from January until May. Given the difference in vegetation and season, this means that the new environment's appearance is also somewhat different from training data. Finally, a shorter, new section of the path was added in the parking lot, which is outside of path from the UTIAS In-The-Dark training dataset. We repeated the path over three days across a full range of lighting change with a 100% autonomy rate. Detailed information on all runs of the experiment is listed in Table 6.4.

Given that some parts of the path in the first generalization experiment overlap with the training data, we also taught two additional paths in November that have no spatial overlap with the training datasets. The first path was taught on November 11th at 09:52 in a field with tall grass and with trees and bushes along the side, see Figure 6.20 for an overhead view. Given that the path was taught in November, there is also difference in appearance compared with the most relevant UTIAS Multiseason data. The second path was taught the same day at 11:34 in a parking lot. An aerial view of the path is given in Figure 6.20. Both paths were repeated seven times with a 100% autonomy rate under different lighting conditions, including driving after dark. Details on the second generalization experiment conducted in the field are listed in Table 6.5. The third generalization experiment from the parking lot is summarized

Table 6.4: Overview of the teach and each repeat run for the first generalization experiment. The teach run is the run with ID 0.

	ID	Start Time	Duration [hh:mm]	Distance [m]	Dist. VO [m]	Failures [count]	Conditions
	0	14.08.21 13:26	-	763.3	-	-	Sun, shadows
	1	15.08.21 12:02	00:25	761.6	0.00	0	Sun, shadows, light sun flare
	2	15.08.21 13:04	00:24	762.2	0.00	0	Sun, shadows, light sun flare
Day	3	15.08.21 14:05	00:24	761.9	0.00	0	Sun, shadows, light sun flare
	4	15.08.21 15:12	00:26	763.3	0.00	0	Sun, shadows, light sun flare
	5	15.08.21 16:16	00:24	763.3	0.00	0	Sun, shadows, light sun flare
	6	15.08.21 17:15	00:24	763.8	0.00	0	Sun, long shadows, sun flare
	7	15.08.21 18:14	00:24	766.3	0.00	0	Sun, long shadows, sun flare
Sunset	8	15.08.21 19:21	00:24	769.1	0.00	0	Sun, long shadows, sun flare
	9	15.08.21 20:28	00:24	765.0	0.00	0	Sun, long shadows, sun flare
Night	10	16.08.21 03:55	00:26	759.6	9.37	0	Dark
	11	16.08.21 05:21	00:28	760.1	10.17	0	Dark
Sunrise	12	16.08.21 06:17	00:26	765.5	0.00	0	Bright, sun not visible
	13	16.08.21 07:23	00:27	763.9	0.00	0	Sun, cloud, long shadows, sun flare
Day	14	16.08.21 10:30	00:28	762.6	0.00	0	Sun, shadows, light sun flare
	15	16.08.21 12:00	00:26	761.8	0.00	0	Sun, shadows, light sun flare
Sunrise	16	20.08.21 07:09	00:28	764.7	0.00	0	Sun, long shadows, sun flare
	17	20.08.21 08:21	00:23	766.8	0.00	0	Sun, long shadows, sun flare
	18	20.08.21 09:19	00:27	764.9	0.00	0	Sun, long shadows, sun flare
	19	20.08.21 10:20	00:25	765.1	0.00	0	Sun, shadows, light sun flare
Day	20	20.08.21 11:15	00:24	764.6	0.00	0	Sun, shadows, light sun flare
	21	20.08.21 12:06	00:24	764.8	0.00	0	Sun, cloud, shadows
	22	20.08.21 17:12	00:25	765.9	0.00	0	Cloudy
	23	20.08.21 18:06	00:24	766.1	0.00	0	Sun, cloud, long shadows, sun flare
Sunset	24	20.08.21 18:50	00:25	766.7	0.00	0	Sun, long shadows, sun flare
	25	20.08.21 19:58	00:25	763.9	0.00	0	Bright, sun not visible
Night	26	20.08.21 20:59	00:25	760.9	9.63	0	Dark
Total	27		10:55	20.6 km	29.17	0	

Table 6.5: Overview of the teach and each repeat run for the second generalization experiment (driven in a field). The teach run is the run with ID 0.

ID	Start Time	Duration [hh:mm]	Distance [m]	Dist. VO [m]	Failures [count]	Conditions
0	11.11.21 09:52	-	276.5	-	-	Cloudy, light sun, windy
1	11.11.21 10:07	00:09	275.0	0.00	0	Cloudy, light sun, windy
2	11.11.21 17:33	00:09	274.2	1.99	0	Dark, windy
3	12.11.21 09:24	00:09	275.9	0.00	0	Low sun, sun flare, long shadows
4	12.11.21 10:59	00:10	275.3	0.00	0	Low sun, sun flare, long shadows
5	12.11.21 13:00	00:09	274.8	0.00	0	Low sun, clouds, long shadows
6	12.11.21 14:56	00:09	275.4	0.04	0	Low sun, sun flare, long shadows
7	12.11.21 16:19	00:09	275.3	0.00	0	Bright, sun not visible
Total 8		01:04	2.2 km	2.03	0	

Table 6.6: Overview of the teach and each repeat run for the third generalization experiment (driven in a parking lot). The teach run is the run with ID 0.

	ID	Start Time	Duration [hh:mm]	Distance [m]	Dist. VO [m]	Failures [count]	Conditions
	0	11.11.21 11:34	-	223.2	-	-	Cloudy, light sun, windy
	1	11.11.21 11:49	00:07	222.8	0.0	0	Cloudy, light sun, windy
	2	11.11.21 17:50	00:08	220.1	0.0	0	Dark, windy
	3	12.11.21 09:40	00:07	222.8	0.0	0	Low sun, sun flare, long shadows
	4	12.11.21 11:15	00:07	223.0	0.0	0	Low sun, sun flare, long shadows
	5	12.11.21 13:15	00:07	221.9	0.0	0	Low sun, sun flare, long shadows
	6	12.11.21 15:15	00:07	221.3	0.0	0	Low sun, sun flare, long shadows
	7	12.11.21 16:34	00:07	222.9	0.0	0	Bright, sun not visible
Total	8		00:50	1.8 km	0.0	0	

in Table 6.6. In the following sections discuss the relevant metrics.

Feature Inlier Count

For robust localization across environmental change, we need good feature matching. As we have discussed earlier, one indication of good feature matching is a high number of feature matching inliers. In Figure 6.34, we present a box plot of the number of feature inliers across all 26 repeats of the first generalization experiment. As expected, we get the highest number of inliers during the day close to the time when the path was taught (13:26). The lowest number of inliers occurs when it is dark. We also see some dips during sunrise and sunset with low sun causing long shadows and sun flares. Overall, we consistently get a high enough number of inliers for localization. The plot shows the count only drops below the localization failure limit (six inliers) for a small portion of the frames when driving during the dark. In the same plot, we compare against the performance of handcrafted SURF, which gets a significantly lower number of inliers. The comparison is done offline by playing the data back in the VT&R offline tools. For several repeats, localization with SURF fails at the beginning of the path and VT&R is unable to start path following. As expected, SURF struggles to match features when the appearance changes beyond a few hours. A possible cause for SURF having a surprisingly low number of inliers even close to the teach time, is that a large part of the path was taught in an area with abundant vegetation under windy conditions, causing the grass and leaves to move. Moreover, trees and bushes also created large moving shadows during teach time, which added to the challenge. Finally, although the repeats were driven at a similar time of day to the teach, they happened up to several days later, causing further strain to SURF matching. We plot the median and quantile regions in Figure 6.35, where we spread out the repeats along the horizontal axis according to hat time of day the repeat was completed.

Since the path contains two sections that overlap with the paths in the training dataset and two new sections, we also look at the number of inliers for the areas contained in the training data compared to the ones outside. We divide up the path into an on-road part and an off-road part since the number of inliers varies between these two environments. In Figure 6.36 (a), we show a box plot for the off-road area. The number of inliers for the section that falls outside of the training data is consistently lower, but at the same time it behaves similarly to the inlier count for the area inside the training data across the different lighting conditions. Despite the inlier count being lower, we still get enough inliers in the

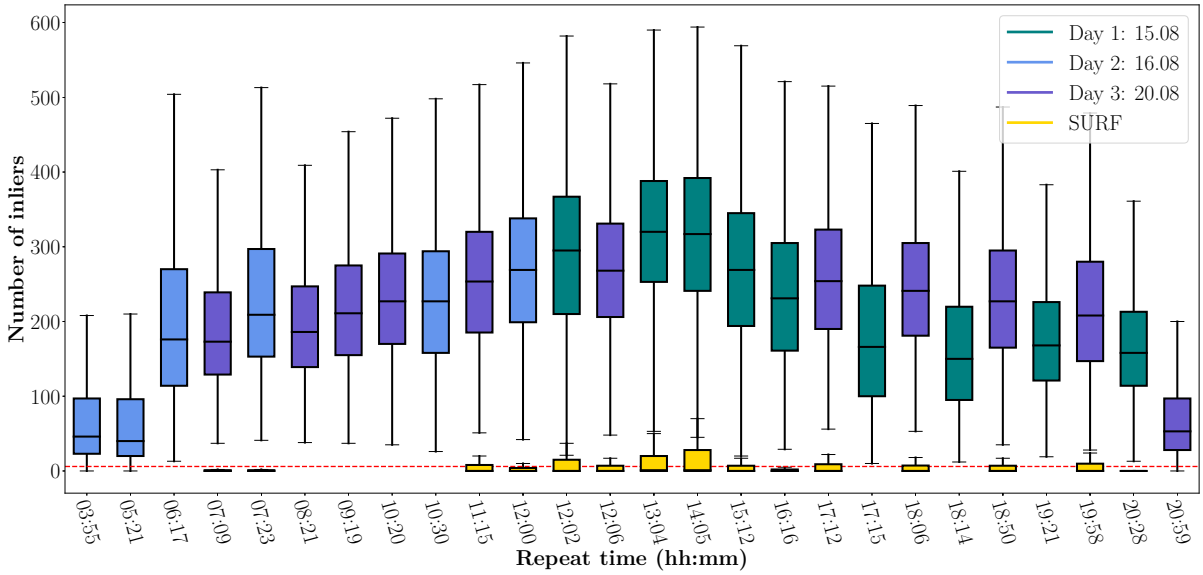


Figure 6.34: A box plot of the number of inlier feature matches for localization for every run of the first generalization experiment listed in Table 6.4. The path was taught at 13:26. The horizontal axis shows the time of day when the path was traversed and we colour the boxes according to the day the run was completed. The dotted red line indicates the threshold of six inliers required for localization. We compare to feature matching with SURF in yellow. Where data is missing for SURF, path following failed from the beginning. The highest number of inliers for the learned features occur during the day and the lowest when it is dark. Dips in the number of inliers also occur at sunrise and sunset.

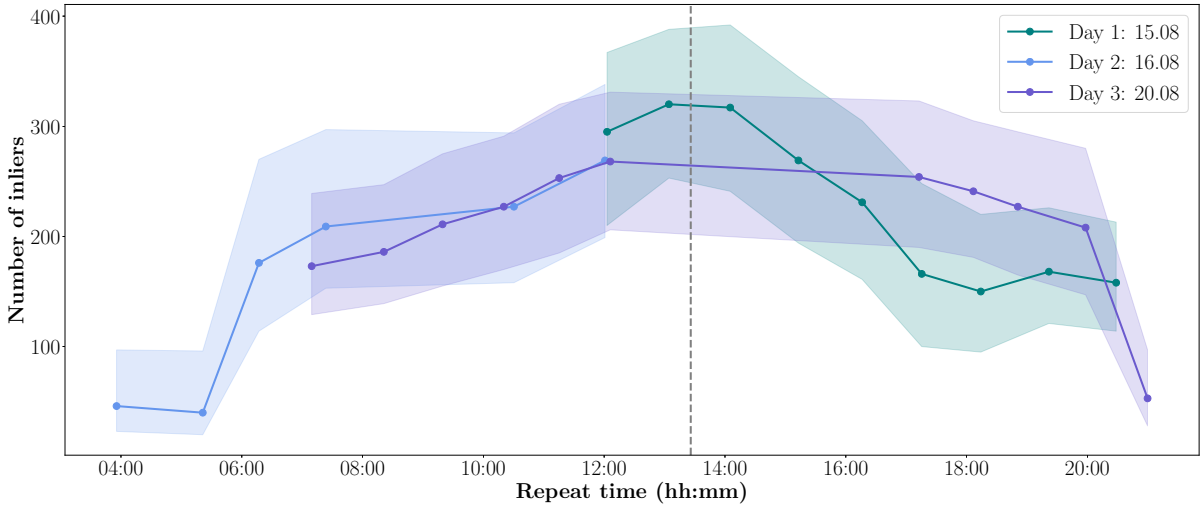
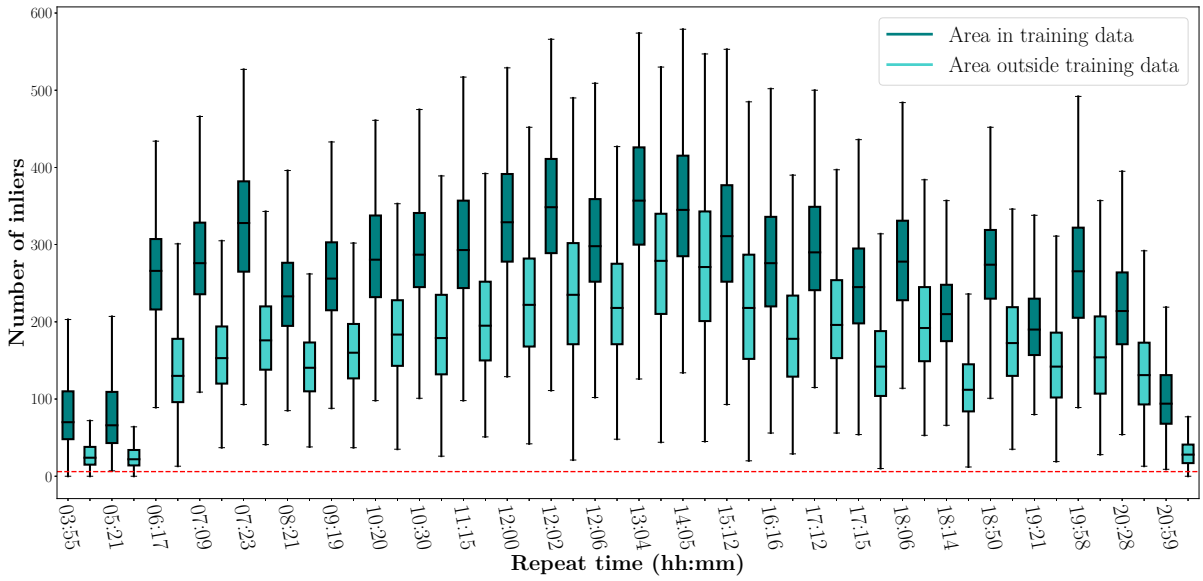
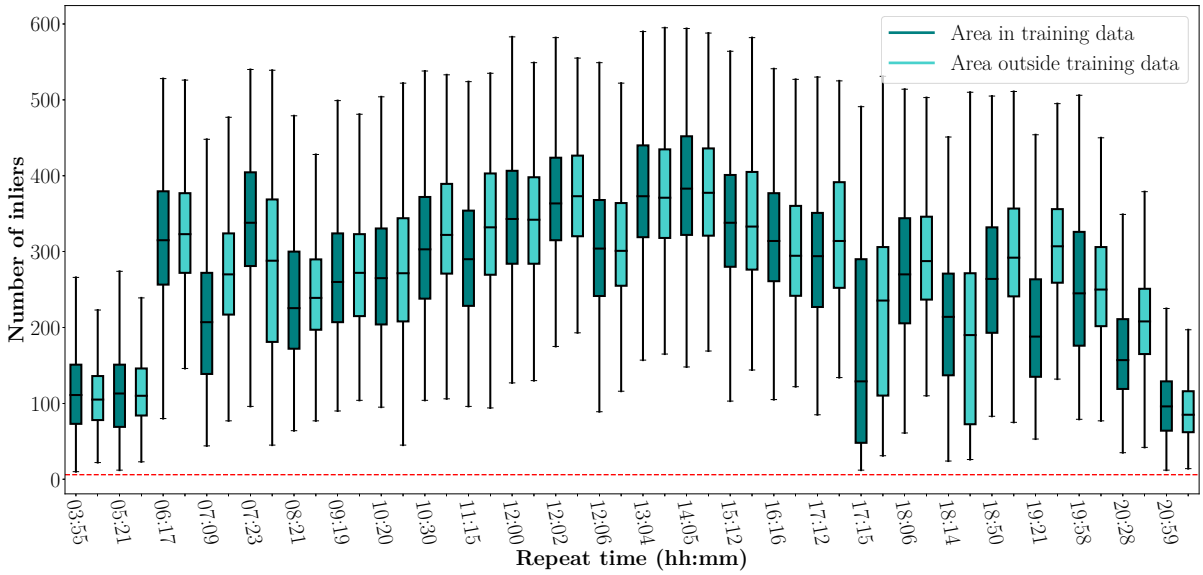


Figure 6.35: A plot of the median number of inlier feature matches for all runs. The shaded regions above and below the line represent the 0.25 and 0.75 quantiles, respectively. The horizontal axis shows the time of day when the path was traversed and we colour the lines according to the day the run was completed. The gray line marks when the path was taught (13:26). The highest number of inliers for the learned features occur during the day and the lowest when it is dark. Dips in the number of inliers also occur at sunrise and sunset.

new off-road area to localize across large lighting change. The same type of plot for the on-road part of the path is presented in Figure 6.36 (b). Here the number of inliers between the area in and outside the



(a) For the off-road data, there is a drop in the number of inliers for the area that falls outside of the training data, but we still get a high enough number of inliers for localization.



(b) For the on-road data, the number of inliers is similar for both cases as the new area is similar in appearance to the training data.

Figure 6.36: A box plot of the number of inlier feature matches for localization for every run of the first generalization experiment. The horizontal axis lists the time of day when the path was traversed. We distinguish between the areas of the path that are contained in and outside the training data and plot the off-road (a) and on-road (b) portions of the path separately. The red line indicates the threshold of six inliers needed for localization.

training data is much more similar. This is because the new area in the on-road section is more similar in appearance to the training data than in the off-road case. In fact, the number of inliers is sometimes a bit higher for the new area, which can be explained by the fact that the new area is less affected by low sun during sunset and sunrise.

We can get a more detailed view of the distribution of number of inliers for each repeat by plotting a CDF over keyframes. In Figure 6.37, we plot the CDF for each repeat, which we colour based on the time of day it was collected. Specifically, we count the number of hours difference to the teach run. Consistently with the box plot, we see that the conditions with the lowest number of inliers are the runs driven during the dark that are coloured in dark purple (early morning) or strong yellow (late night). Furthermore, the number of inliers is consistently high with over 90% of keyframes having at least 100 inliers across all repeats.

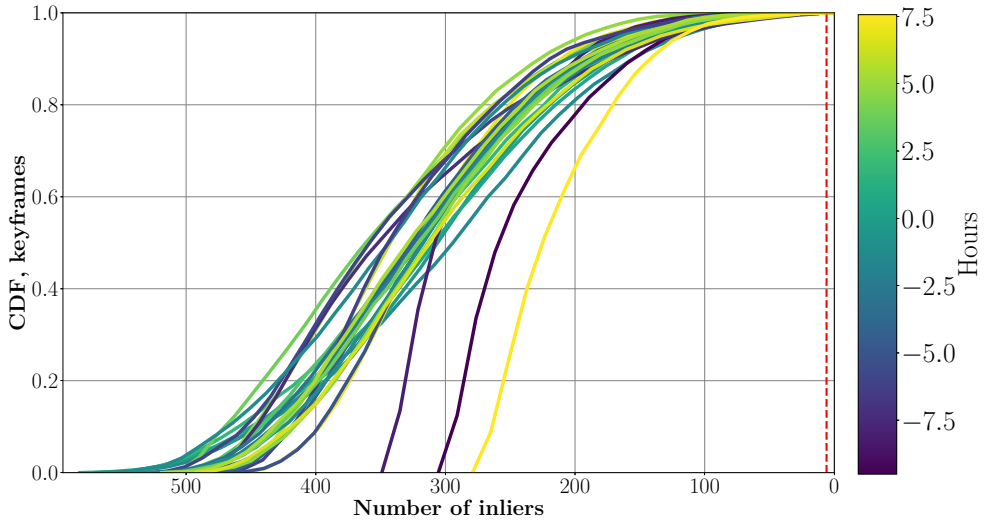


Figure 6.37: The CDF of keyframes with a given number of inliers for each repeat in the first generalization experiment. The red vertical line shows the localization failure threshold of six inliers. The lines are coloured according to the time of day that they were collected, given by the number of hours before or after the path was taught (13:26). As expected, the runs collected at the beginning of the day and at night (in dark blue and strong yellow) have the lowest number of inliers.

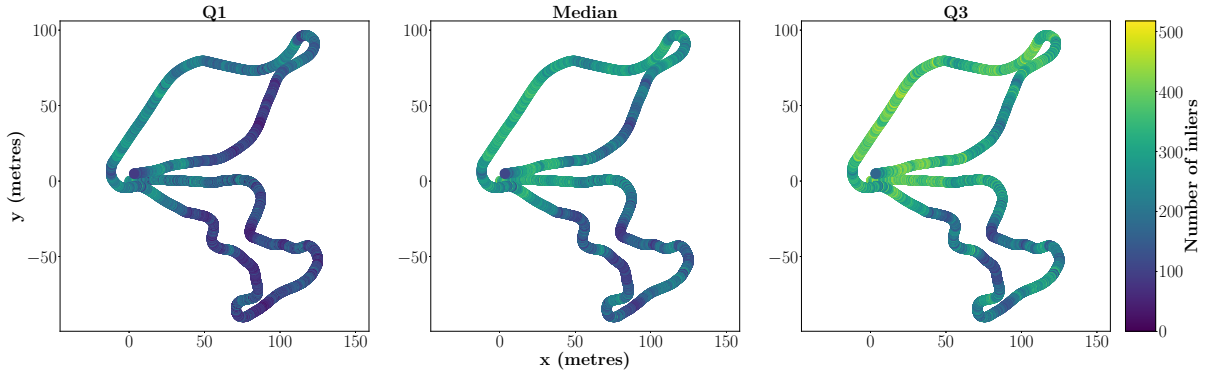


Figure 6.38: The plot shows the median number of inliers across all runs for each map vertex along the path, as well as the 0.25 (Q1) and 0.75 (Q3) quantiles. The number of inliers is higher in the on-road compared to the off-road part of the path.

Finally, Figure 6.38 presents the inlier median and quantiles over all runs for each vertex in the map. This means we plot the path in the xy-plane and colour each map coordinate according to the number of inliers. This visualization is especially informative for this path that contains both sections that overlap

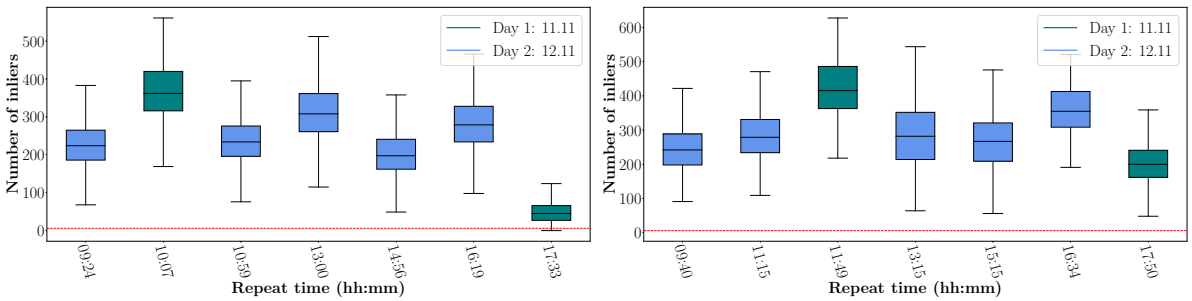


Figure 6.39: Box plots of the number of inlier feature matches for localization for every run of the second and third generalization experiments listed in Tables 6.5 and 6.6, respectively. The paths were taught at 09:52 and 11:34, respectively. The horizontal axis shows the time of day when the path was traversed, and we colour the boxes according to the day the run was completed. The red line indicates the threshold of six inliers required for localization.

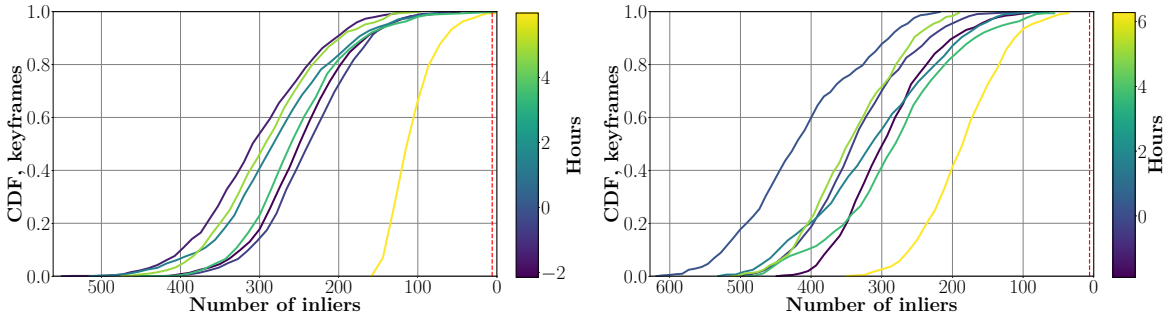
with and fall outside the training data. First of all, we see that the half of the path that falls in the structured, on-road area around the Mars Dome (above $y = 0$ in the plot), predictably has the highest number of inliers. Consistent with the lighting-change experiment in Section 6.7.2, the number of inliers decreases a bit for the section where we drive on grass and have less permanent structures in view (the straight stretch after the sharp turn located in the top right corner of the plot). We also see that the new section of the on-road path (not seen in training data), has a similarly high number of inliers to the section that falls in the training data (as we saw in Figure 6.36 comparing number of inliers per repeat).

The bottom loop in the plot (below $y = 0$) covers the off-road area. Let us consider coordinate (0, 0) to be the start of the loop. Then the start and end of this loop covers the area included in the UTIAS Multiseason training data (as seen in Figure 6.16) and we observe that the number of inliers is higher in this area than for the rest of the loop that falls outside the training data. Again, this is consistent with the comparison of inliers inside and outside the training data in Figure 6.36.

Next we look at the feature matching inliers for the last two generalization experiments. We present box plots of the feature matching inliers for the second and third generalization experiments conducted in a field and a parking lot, respectively, in Figure 6.39. Despite teaching and repeating the paths in areas that have no spatial overlap with the training data, we get a high number of inliers across the different times of day. Moreover, we have relied on training data collected in the summer 2016 and January to May 2017, which means that seasonal appearance is also different. As before, we see that the repeats after dark (at 17:33 and 17:50, respectively) are the most challenging, though only causing localization failures in the experiment in the field. In the field there is no artificial lighting other than the robot's headlights, while the parking lot has outdoor lights.

In order to get more detail on the number of inliers for each run, we plot the CDF over keyframes for both paths in Figure 6.40 and colour the lines according to the time of day when the repeats were completed. As expected, the repeats after dark (indicated by strong yellow lines) have the lowest number of inliers. Even so, the path in the parking lot gets 100 or more inliers for at least 90% of the keyframes for the repeat after dark. For the after-dark repeat in the field, we get at least 100 inliers for approximately 65% of the keyframes, but as we can see from the figure only few keyframes cross the localization threshold of six inliers. Again, it is evident that the off-road environment, especially after dark, is a more challenging scenario for localization than the more structured on-road environment.

Finally, we plot the number of inliers against different locations along the two paths in Figure 6.41.



(a) Generalization II (field). (a) Generalization III (parking lot).

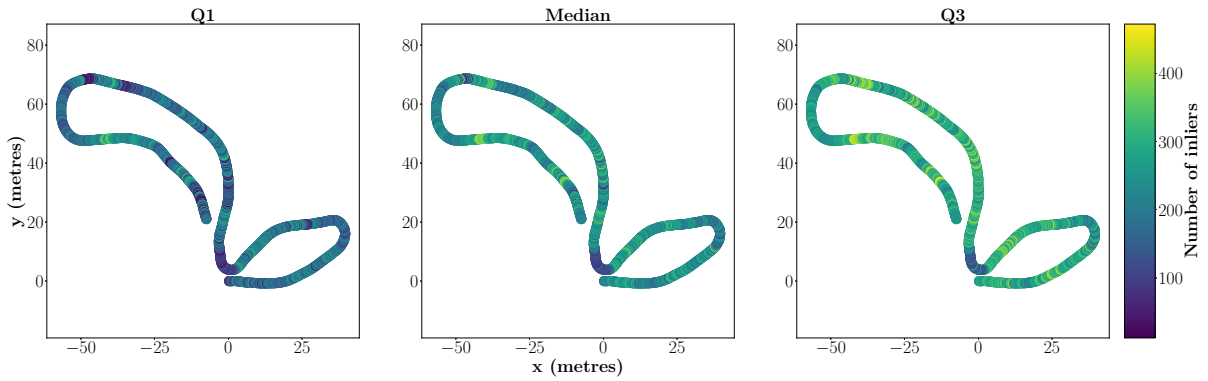
Figure 6.40: The CDF of keyframes with a given number of inliers for each repeat in the third generalization experiment. The red vertical line shows the localization failure threshold of six inliers. The lines are coloured according to the time of day that they were collected, given by the number of hours before or after the teach was collected.

The number of inliers is fairly similar along the whole path in both instances, except that it drops in one sharp turn for the experiment conducted in the field. This shows we get robust feature matching consistently along two paths that were collected in very different environments that have no spatial overlap with the training data.

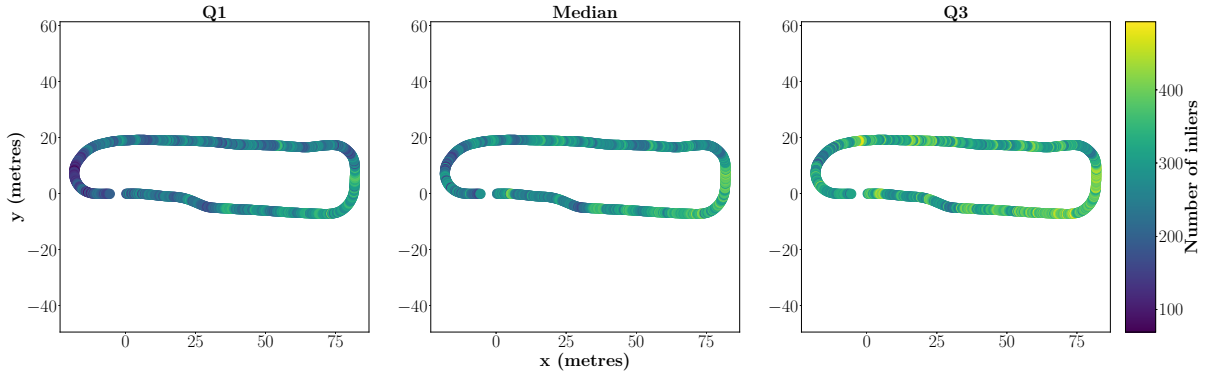
Feature Inlier Distribution

As explained in Section 6.6, in addition to a high number of inliers, we prefer that features are well distributed across the image. We provide qualitative results by illustrating the feature distribution with heat maps as we did for the lighting-change experiment in Section 6.7.2. In Figure 6.42, we include three heat maps from the off-road area not included in the training data. In all these examples, there were large shadows or sections with very high brightness in the map vertex from the teach pass (similar to day images included in the figure). We can see that the areas with large shadows generally gets fewer feature matching inliers across all the heat maps (see left half of each heat map). In comparison, the first two scenarios ((a) and (b)) in Figure 6.43, which are also from the off-road area outside of training data, did not have large shadows in the map vertex. In this case the feature matching inliers are distributed more evenly across the image. Although, large shadows in the map vertices can cause some reduction in feature matching in the off-road environment, we still get enough inliers for successful path following. From both Figures 6.42 and 6.43, it is clear that the number of inliers is significantly lower during night time, especially in the challenging, unstructured, off-road area. However, localization failures that may occur are still few enough that we can continue path following with the help of dead-reckoning from VO.

All of the examples contained in Figure 6.43, are from the off-road area. The first two, (a) and (b), are from locations not contained in the training data, while (c), has overlap with the UTIAS Multiseason training dataset. We see that the latter example has a higher number of inliers compared to the first two, though a shadow in the map image is causing some reduction in inliers in the bottom left corner. The second example, (b), presents a particularly challenging scenario during night time driving because the robot is looking toward an open area with no bushes or trees close enough to be visible in the range of the head lights. For the same example, we can also see that the sun flares during some of the sunset runs have affected matching in the upper part of the image illustrated by an area of light-blue bins.



(a) Generalization II (field).



(b) Generalization III (parking lot).

Figure 6.41: The plot shows the median number of inliers across all runs for each map vertex along the path for the second and third generalization experiments, as well as the 0.25 (Q1) and 0.75 (Q3) quantiles.

Finally, Figure 6.44, shows examples from the on-road area, where the first two locations ((a) and (b)) are not contained in the training data. Overall, it is clear that we get more features matches across a larger part of the image in the structured on-road examples versus the unstructured off-road examples. In example (a), interestingly, we get consistently few features matches on the blue container, which is not contained in the training data, though we get more matches on the building in example (b). The last heat map, (c), is from a location contained in the training data. Here, the grass causes a reduction in feature matches close to the robot for the sunset and night time conditions.

In addition to looking at qualitative examples, we can also compare the on-road and off-road areas by looking at the depth of detected features. In Figure 6.45, we show a box plot of the depth of all landmarks generated from feature-matching inliers across each repeat. Moreover, we differentiate between landmarks from the off-road and on-road sections of the path. Across all repeats, we detect a higher proportion of features farther away from the robot in the on-road area compared to the off-road area. The distribution of feature depth does not change significantly for most runs. However, we see that driving at night in the off-road area results in a lower proportion of features farther from the robot, which is explained by driving in a completely dark area with headlights that have limited range. This is less of an issue in the on-road area due to the presence of artificial light from buildings and street lamps. In fact, we see the opposite trend in the on-road area. There, the proportion of features with larger depth is higher during night-time. This is likely caused by feature matching becoming more difficult

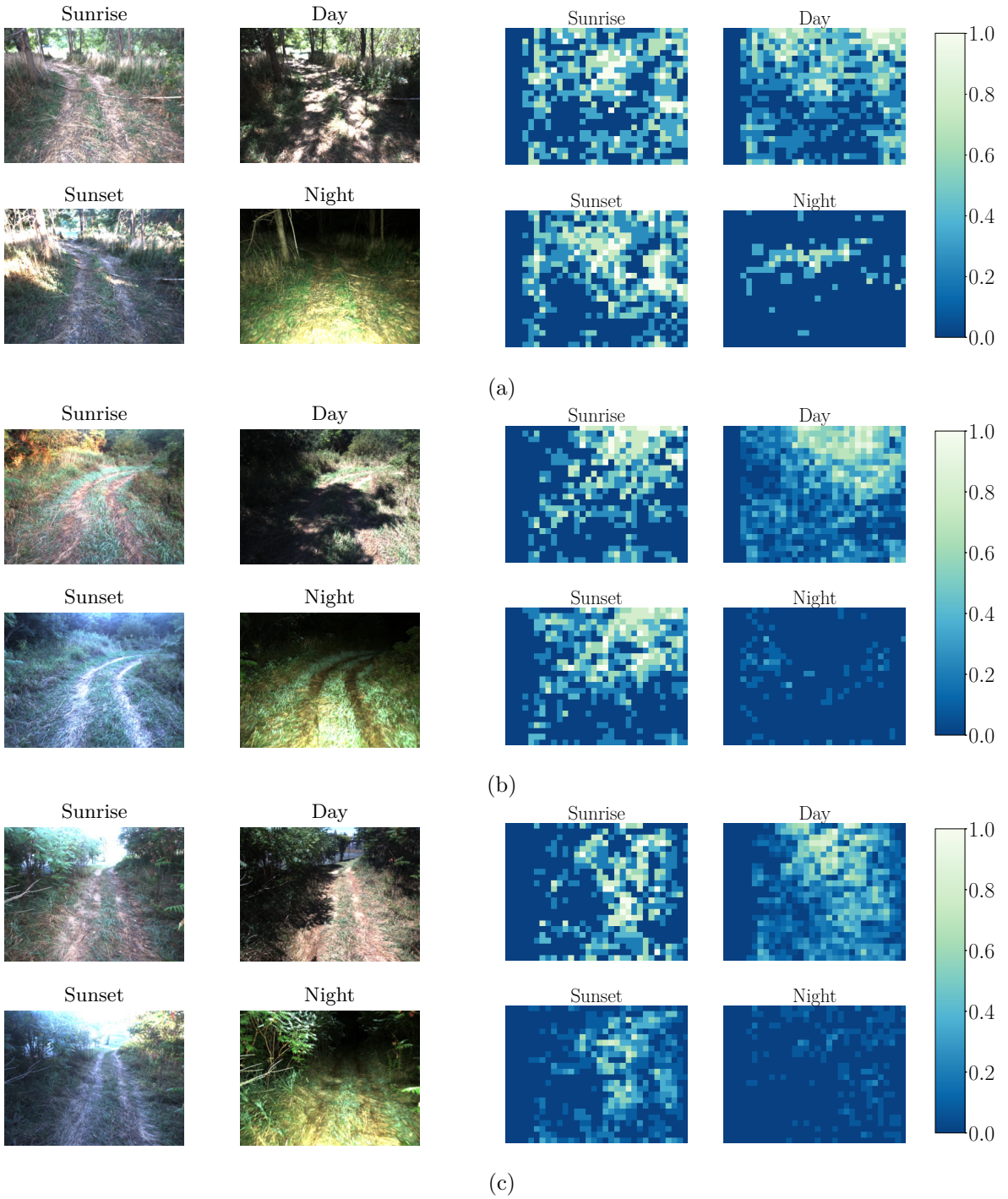


Figure 6.42: Distribution of feature matching inliers for one map vertex along the path. We count the number of inliers that fall into the 16×16 windows in the heat map for all live keyframes localized to the given map vertex. We normalize by the number of localized keyframes. On the left we show one example live image localized to the map vertex for each time of day.

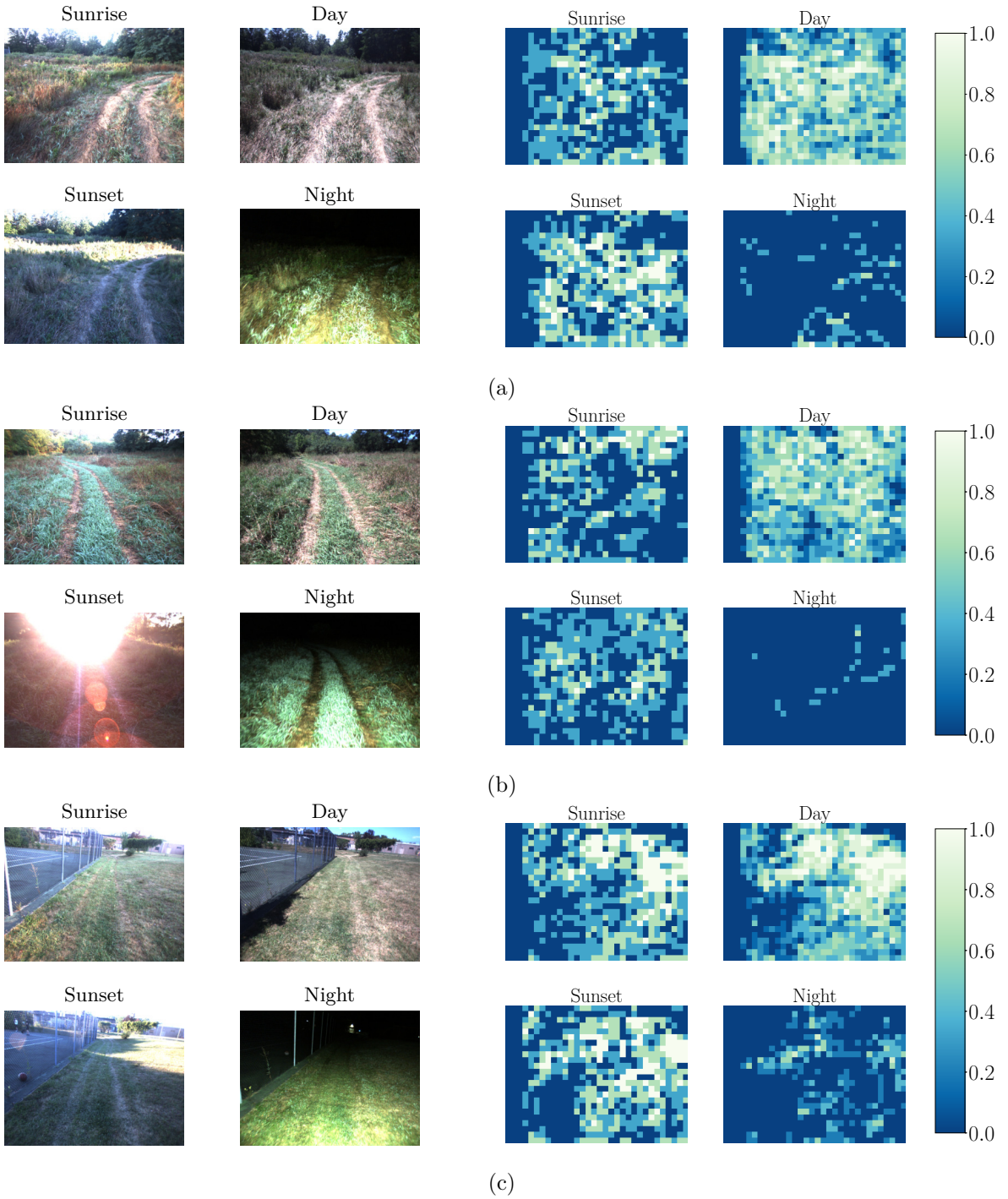


Figure 6.43: Distribution of feature matching inliers for one map vertex along the path. We count the number of inliers that fall into the 16×16 windows in the heat map for all live keyframes localized to the given map vertex. We normalize by the number of localized keyframes. On the left we show one example live image localized to the map vertex for each time of day.

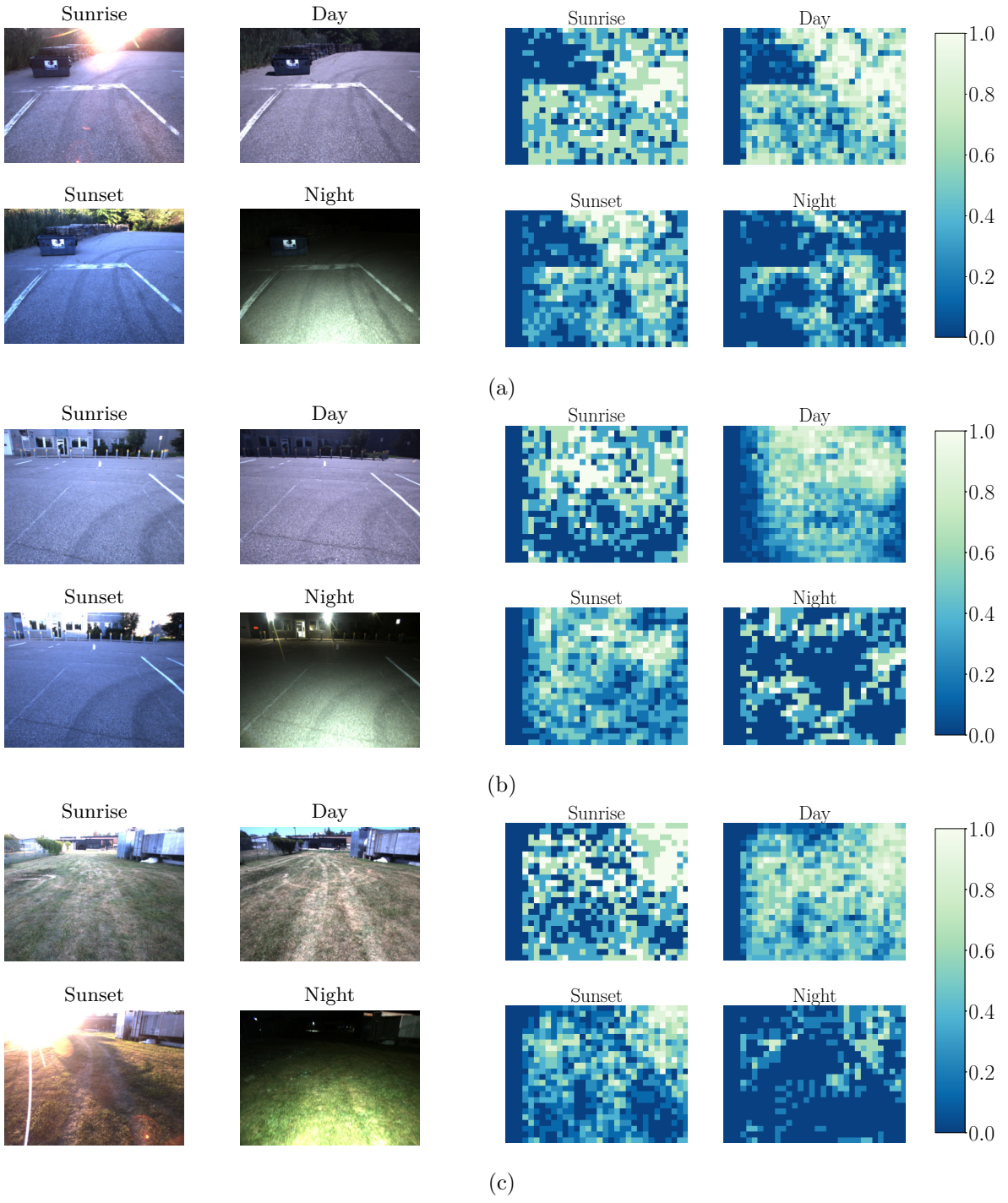


Figure 6.44: Distribution of feature matching inliers for one map vertex along the path. We count the number of inliers that fall into the 16×16 windows in the heat map for all live keyframes localized to the given map vertex. We normalize by the number of localized keyframes. On the left we show one example live image localized to the map vertex for each time of day.

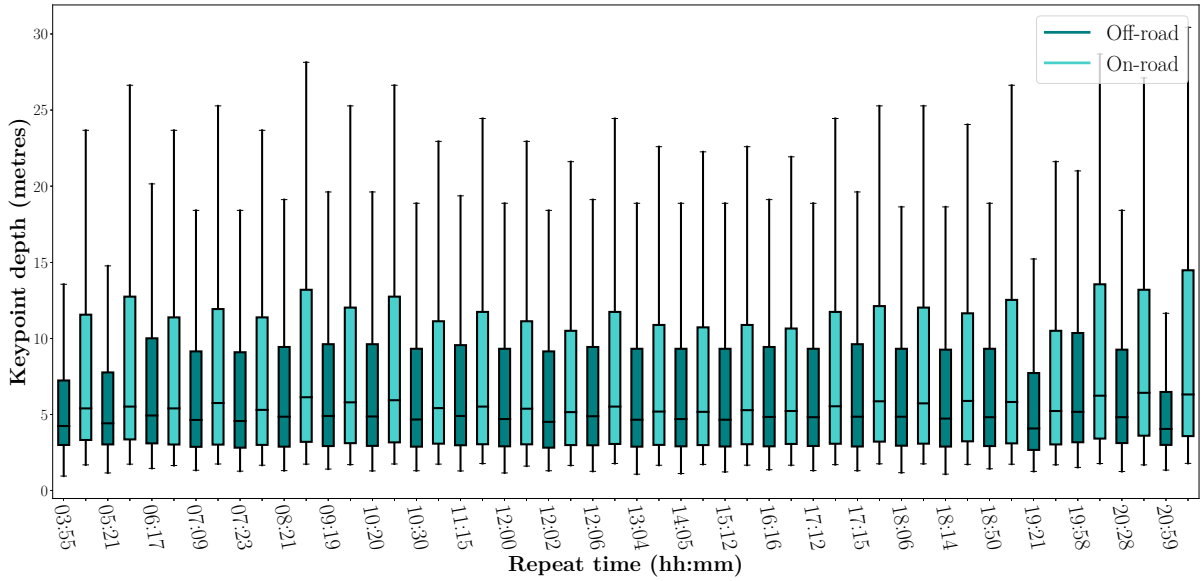
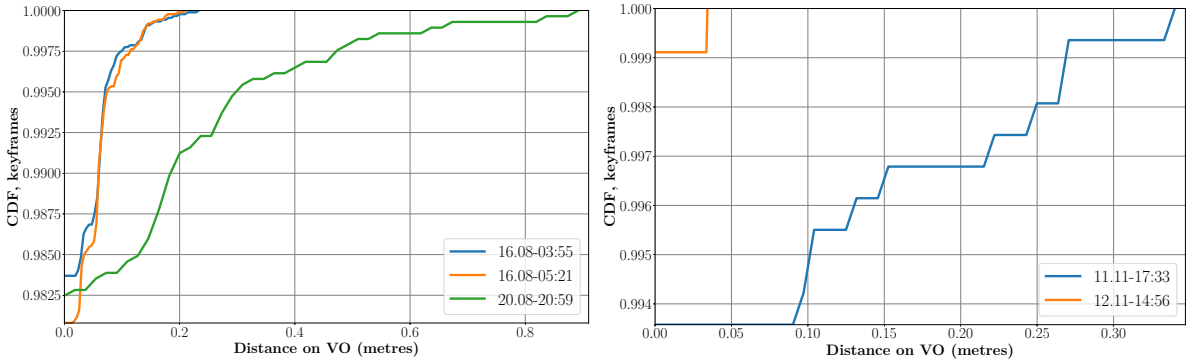


Figure 6.45: We show a box plot of the depth of keypoints for all feature-matching inliers for each repeat. We compare the on-road and off-road areas.



(a) Generalization I experiment.

(b) Generalization II experiment.

Figure 6.46: A CDF over keyframes for the distance VT&R relies on VO to compensate for localization failures. The horizontal axis lists the distance VT&R has relied on dead-reckoning for a keyframe.

close to the robot, especially when driving on grass, while we retain features on farther-away buildings or other structures (see Figure 6.44 (b) and (c). Finally, we also see a drop in the proportion of father-away features for the off-road area at 19:21, which is caused by large sun flares at sunset.

To summarize, we see that the distribution of matched features in the image varies between different environments (on-road and off-road), but can also be affected by challenging lighting conditions in the map such as large shadows or very bright spots. MEL has the advantage that it keeps adding new conditions if the initial teach provides some challenging appearance conditions. With the learned features the initial teach is the only experience available for localization. Even so, we see so far that we get enough inliers and they are sufficiently spread out across the image that we accomplish successful path following across a full range of lighting change. Finally, we also saw that there is some difference between the on-road and off-road areas during night-time, where detecting features with larger depths is more difficult in a dark area where the robot's headlights provide the only significant source of light.

Distance on Dead-Reckoning

Whenever the localization fails due to an insufficient number of inliers, we rely on dead-reckoning with VO until localization is regained. Figure 6.46 shows a CDF over live keyframes for the distance that we rely on VO for any repeat that has localization failures. Plot (a) show that we get localization failures for three runs in the first generalization experiment, all of which were driven during the dark. We never rely on VO for any longer than 1.0 metre and so localization is quickly regained in all cases. Moreover, we only rely on VO for less than 2% live of keyframes in any of the three night-time runs. The second localization experiment driven in the field, (b), has two runs with localization failures, one after dark and one in the afternoon. VO is used for a maximum of 0.4 metres and for less then 2% of the keyframes. The third generalization experiment completed in the parking lot had no localization failures. Overall, we see that localization failures are rare and mostly occur after dark. None of the localization failures have caused failure in path following as we rely on VO until localization is quickly reestablished.

Autonomy Rate

For all three generalization experiments autonomous path following succeeded for all runs without any manual intervention and so the experiments were completed with 100% autonomy rate, see Tables 6.4, 6.5, and 6.6.

Conclusion

In this experiment, we used learned features trained on the UTIAS Multiseason and UTIAS In-The-Dark data from 2017 and 2016, respectively. We tested the features' ability to generalize to areas and appearance conditions not seen in the training data. We successfully repeated a 760 metre path that contains both a challenging off-road area as well as a more structured on-road area. We repeated the path with a 100% autonomy rate across a full range of lighting change. Additionally, we teach a path in a field and another one in a parking lot that we repeat seven times each for different lighting conditions with a 100% autonomy rate. These two paths had no spatial overlap with the training data. After analyzing the feature matching inliers, we see that the structured on-road areas generally get more feature matching inliers than more challenging off-road areas. Driving on a road surrounded by buildings or other structures is easier for vision-based path following with learned features than traversing off-road areas with abundant vegetation and fewer permanent visual markers. We get a reduction in inliers in areas outside the training data compared to areas contained in the training data. However, across all the experiments and all conditions, we get enough inliers for successful path following with VT&R.

6.7.4 Experiment 4: Seasonal Change

Our final experiment attempts to extend testing of the learned features to seasonal appearance change. We use the same path taught for the first generalization experiment on August 14th and repeat it a total of 72 times until November 19th, 97 days after the map was created. We keep using the same network trained with data from the UTIAS Multiseason and UTIAS In-The-Dark datasets, which were gathered from January until May 2017 and in summer 2016, respectively. This means that we encounter seasonal appearance conditions not seen in the training data. As before, the path covers both areas that are spatially contained in the training datasets and sections which are not, see Figure 6.16 for an aerial view of the path. We repeated the path for different lighting conditions over the three months. Out of

Table 6.7: Overview of the repeat runs completed in September for the seasonal-change experiment.

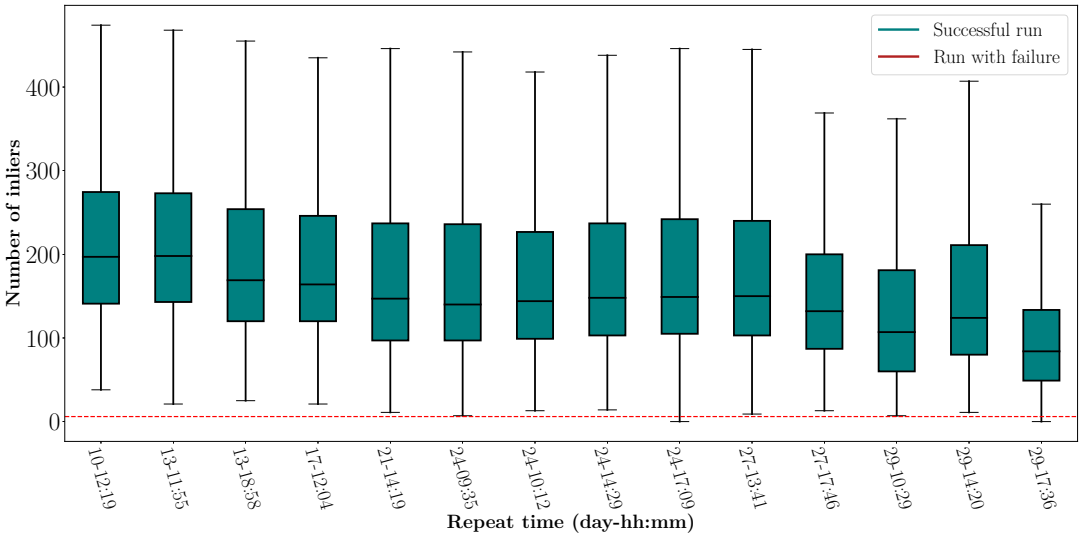
	ID	Start Time	Duration [hh:mm]	Distance [m]	Dist. VO [m]	Failures [count]	Conditions
	1	10.09.21 12:19	00:26	765.9	0.00	0	Sun, cloud, light sun flare, shadows
	2	13.09.21 11:55	00:27	765.2	0.00	0	Sun, light sun flare, shadows
	3	13.09.21 18:58	00:25	768.2	0.00	0	Low sun
	4	17.09.21 12:04	00:22	762.7	0.00	0	Cloudy
	5	21.09.21 14:19	00:23	763.4	0.00	0	Sun, light sun flare, shadows, windy
	6	24.09.21 09:35	00:29	766.2	0.00	0	Cloud, sun, light sun flare, shadows
	7	24.09.21 10:12	00:28	765.1	0.00	0	Cloud, sun, light sun flare, shadows
	8	24.09.21 14:29	00:21	763.6	0.00	0	Cloudy
	9	24.09.21 17:09	00:22	762.8	0.26	0	Cloudy
	10	27.09.21 13:41	00:22	762.4	0.00	0	Sun, light sun flare, shadows
	11	27.09.21 17:46	00:26	762.3	0.00	0	Low sun, sun flare, long shadows
	12	29.09.21 10:29	00:22	762.0	0.00	0	Low sun, sun flare, long shadows
	13	29.09.21 14:20	00:17	762.0	0.00	0	Sun, light sun flare, shadows
	14	29.09.21 17:36	00:17	762.3	0.36	0	Low sun, sun flare, long shadows
Total	14		05:27	9.9 km	0.62	0	

the 72 runs, 15 runs incurred path following failures, for which we, in most cases, restarted and kept driving. Most of the failures occurred while driving after dark or during low sun, suggesting that the compound lighting and seasonal change became difficult for the learned features in some cases. Detailed information on all runs of the experiment is listed in three tables that we have divided by month; Tables 6.7, 6.8, and 6.9.

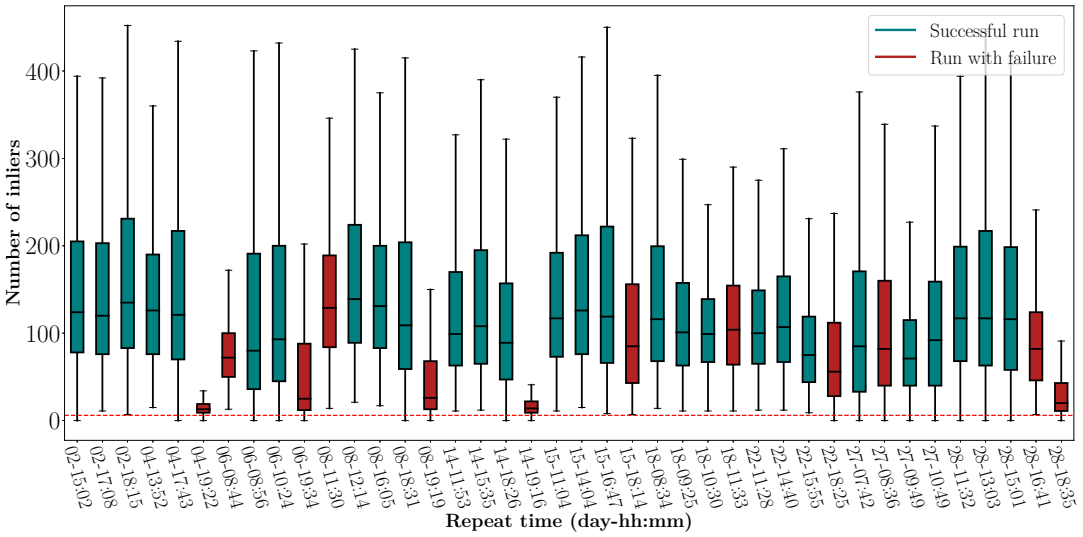
Feature Inlier Count

We start by investigating the feature matching inlier count for each run of the experiment. Due to space constraints, we create box plots for each month separately, see Figure 6.47. Starting with the first repeat on September 10th, the median number of inliers is approximately 200. As time passes through September and the later months, if we ignore he night-time repeats, we see that med median number of inliers decreases to around 100 and then also below that in some cases. Although there are some fluctuations between individual repeats, we see that the median number of inliers gradually decreases across the months. The runs that contain path-following failures are marked in red colour in the plot. We note that a few of the runs with failures have a fairly high number of inliers. In some cases path following failed not due to the lack of features but because the robot drove too close to trees or bushes near the path, and we had to manually intervene. All the path-following failures will be discussed in more detail in the later Autonomy Rate section.

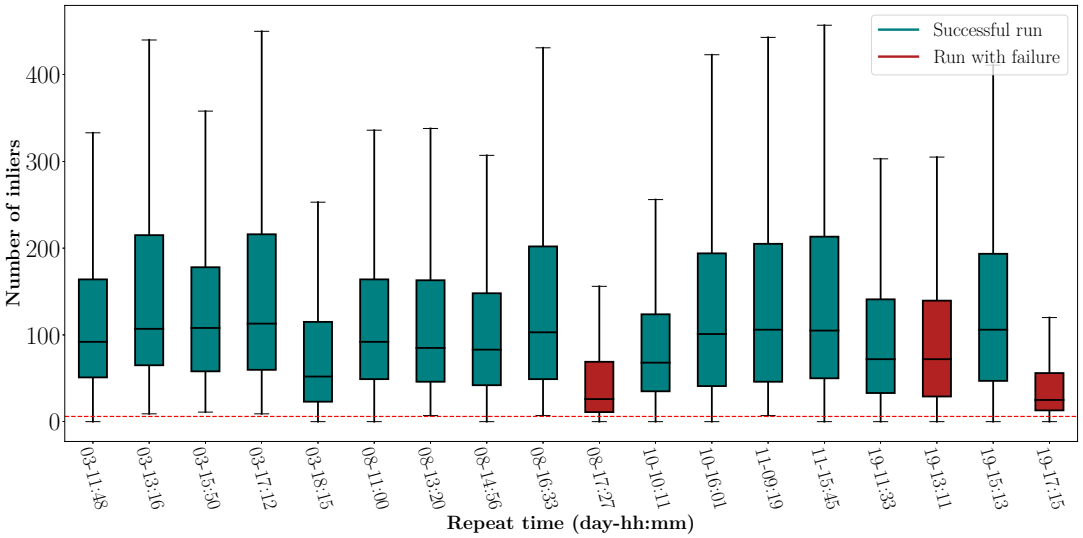
We get the lowest number of inliers when repeating the path after dark. The reason that the repeats on the 4th (at 19:22), 6th (at 08:44), 14th (at 19:16), and 28th (at 18:35) of October have lower medians than the remaining runs, is that we stopped repeating the path after the first failure. This means that the repeat was cut short before (4th, 6th, 14th) or early on along (28th) the on-road part of the path that generally gets more inliers. In the first two cases, we did not attempt re-staring VT&R. On October 14th and 28th, we were unable to restart after a failure. During October, seven out twelve repeats with path-following failures occurred just before or after dark, whereas this was the case for two of three



(a) September



(b) October



(c) November

Figure 6.47: A box plot of the number of inlier feature matches for localization for every run of the experiment listed in Tables 6.7, 6.8 and 6.9. The horizontal axis lists the date and time of day when the path was traversed.

Table 6.8: Overview of the repeat runs completed in October for the seasonal-change experiment.

ID	Start Time	Duration [hh:mm]	Distance [m]	Dist. VO [m]	Failures [count]	Conditions
15	02.10.21 15:02	00:30	762.6	0.31	0	Sun, light sun flare, shadows
16	02.10.21 17:08	00:21	762.3	0.0	0	Low sun, sun flare, long shadows
17	02.10.21 18:15	00:22	762.0	0.0	0	Bright, sun not visible
18	04.10.21 13:52	00:22	766.0	0.0	0	Cloudy, wet ground
19	04.10.21 17:43	00:22	763.3	0.60	0	Cloudy, wet ground
20	04.10.21 19:22	00:08	372.9	88.22	1	Dark, wet ground
21	06.10.21 08:44	00:05	178.1	0.70	1	Low sun, sun flare, long shadows
22	06.10.21 08:56	00:28	762.9	3.30	0	Cloudy
23	06.10.21 10:24	00:22	761.3	2.23	0	Cloudy
24	06.10.21 19:34	00:35	654.6	34.57	2	Dark
25	08.10.21 11:30	00:23	761.9	0.0	1	Cloudy
26	08.10.21 12:14	00:22	762.3	0.0	0	Cloudy
27	08.10.21 16:05	00:22	763.6	0.0	0	Cloudy
28	08.10.21 18:31	00:22	763.2	0.48	0	Bright, sun not visible
29	08.10.21 19:19	00:23	760.0	32.36	1	Dark
30	14.10.21 11:53	00:22	761.9	0.0	0	Sun, light sun flare, shadows
31	14.10.21 15:35	00:22	762.5	0.0	0	Sun, light sun flare, long shadows
32	14.10.21 18:26	00:22	761.5	3.47	0	Bright, sun not visible
33	14.10.21 19:16	00:08	275.9	53.54	1	Dark
34	15.10.21 11:04	00:22	762.9	0.0	0	Cloudy
35	15.10.21 14:04	00:22	763.7	0.0	0	Cloudy
36	15.10.21 16:47	00:22	763.4	0.0	0	Cloudy
37	15.10.21 18:14	00:23	766.5	1.75	1	Dusk, cloudy, wet ground
38	18.10.21 08:34	00:22	764.2	0.0	0	Cloudy
39	18.10.21 09:25	00:22	764.0	0.0	0	Mostly cloudy, some sun
40	18.10.21 10:30	00:22	763.7	0.0	0	Clouds, sun, light sun flare, long shadows
41	18.10.21 11:33	00:25	762.4	0.0	1	Clouds, sun, light sun flare, shadows
42	22.10.21 11:28	00:22	765.1	0.0	0	Clouds, sun, shadows
43	22.10.21 14:40	00:22	760.8	0.0	0	Sun, clouds, light sun flare, shadows
44	22.10.21 15:55	00:22	765.8	0.0	0	Low sun, sun flare, long shadows
45	22.10.21 18:25	00:25	760.5	10.83	2	Dusk to dark
46	27.10.21 07:42	00:22	764.9	3.11	0	Bright, sun not visible
47	27.10.21 08:36	00:23	763.0	1.00	1	Low sun, sun flare, long shadows
48	27.10.21 09:49	00:23	765.2	1.32	0	Low sun, sun flare, long shadows
49	27.10.21 10:49	00:28	764.6	0.34	0	Low sun, light sun flare, long shadows
50	28.10.21 11:32	00:24	761.9	0.25	0	Clouds, sun, light sun flare, shadows
51	28.10.21 13:03	00:24	762.4	0.16	0	Cloudy
52	28.10.21 15:01	00:24	764.4	1.27	0	Sun, light sun flare, shadows
53	28.10.21 16:41	00:28	768.7	0.0	1	Low sun, sun flare, long shadows
54	28.10.21 18:35	00:20	464.6	26.60	1	Dark
Total	40	14:48	20.6 km	266.41	14	

failures in November. Of the remaining six repeats with failures, five occurred during low sun causing long shadows and sun flares. However, not all runs with low sun caused path-following failures. In fact, more often than not they did not result in failures (fourteen successful runs versus five runs with failures). The compound appearance change from seasonal changes, such as vegetation changing colour and leaves falling, together with the lighting change, caused a low number of feature matching inliers

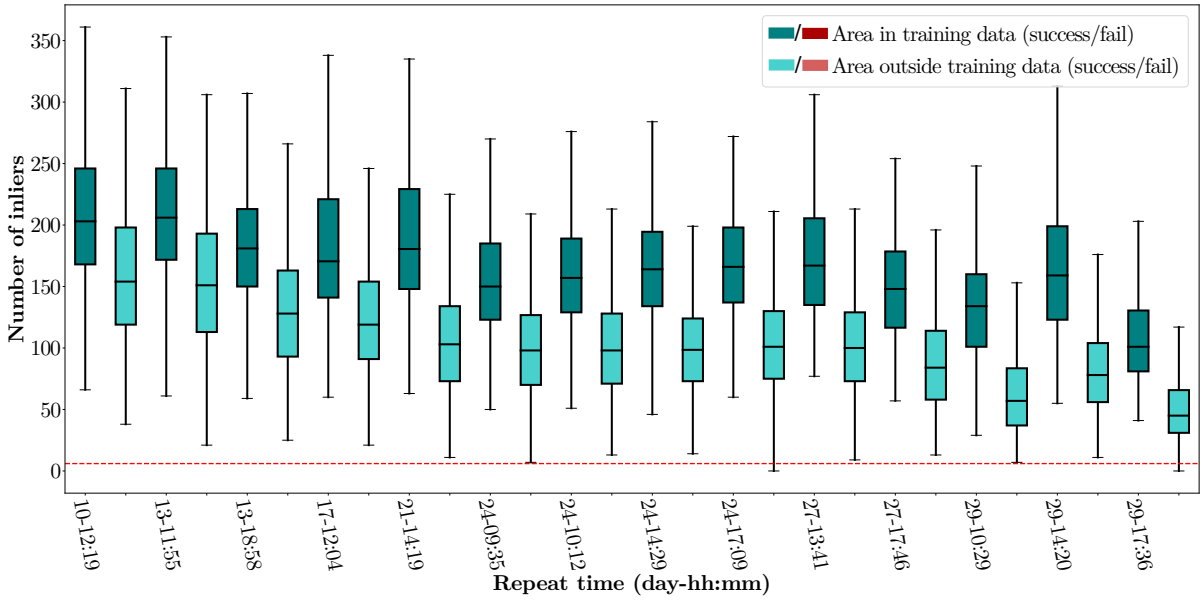
Table 6.9: Overview of the repeat runs completed in November for the seasonal-change experiment.

	ID	Start Time	Duration [hh:mm]	Distance [m]	Dist. VO [m]	Failures [count]	Conditions
	55	03.11.21 11:48	00:25	761.8	0.27	0	Low sun, light sun flare, long shadows
	56	03.11.21 13:16	00:25	764.0	0.00	0	Cloud, sun, shadows
	57	03.11.21 15:50	00:24	764.3	0.00	0	Cloudy
	58	03.11.21 17:12	00:24	763.4	0.00	0	Cloudy
	59	03.11.21 18:15	00:23	770.6	5.85	0	Dusk to dark
	60	08.11.21 11:00	00:23	758.6	0.30	0	Low sun, light sun flare, long shadows
	61	08.11.21 13:20	00:24	762.6	0.00	0	Low sun, light sun flare, long shadows
	62	08.11.21 14:56	00:25	764.0	0.11	0	Low sun, sun flare, long shadows
	63	08.11.21 16:33	00:25	763.2	0.00	0	Bright, sun not visible
	64	08.11.21 17:27	00:29	761.6	45.57	4	Dark
	65	10.11.21 10:11	00:24	759.0	1.22	0	Low sun, light sun flare, long shadows
	66	10.11.21 16:01	00:28	765.5	0.69	0	Bright, sun not visible
	67	11.11.21 09:19	00:24	762.9	0.00	0	Cloudy
	68	11.11.21 15:45	00:24	765.2	0.88	0	Mostly cloudy, low sun
	69	19.11.21 11:33	00:25	762.9	1.06	0	Low sun, clouds, light sun flare, long shadows
	70	19.11.21 13:11	00:28	766.5	5.09	2	Low sun, clouds, light sun flare, long shadows
	71	19.11.21 15:13	00:25	763.1	0.65	0	Clouds, low sun, sun flare, long shadows
	72	19.11.21 17:15	00:27	758.0	32.75	3	Dark
Total	18		07:32	13.7 km	94.49	10	

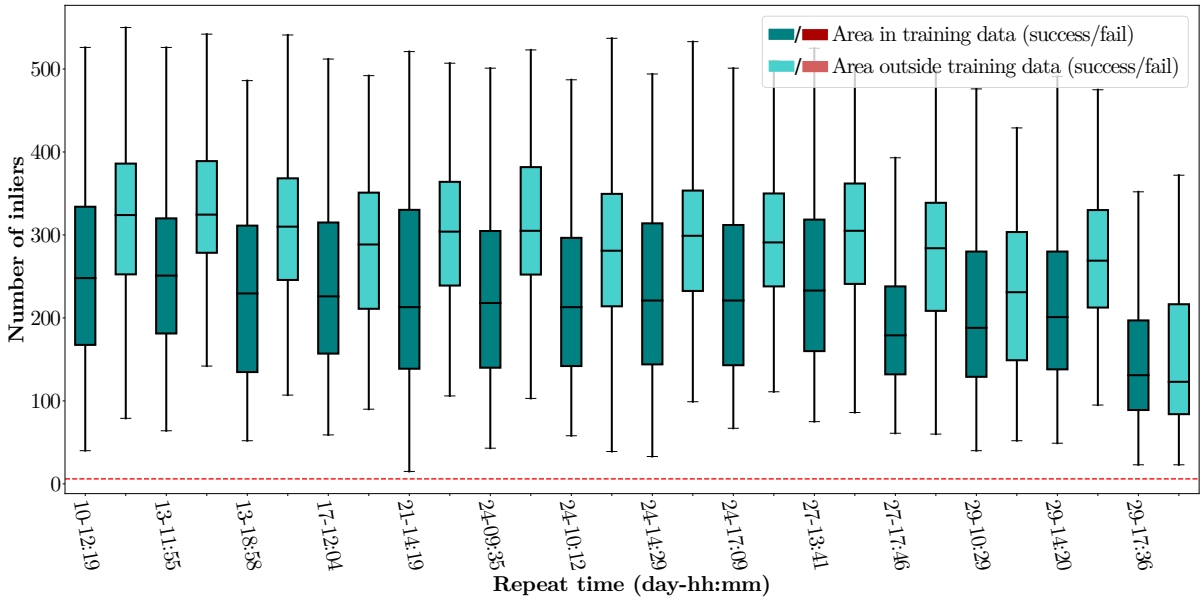
leading to one or more path following failures for these repeats. Despite the number of matched feature inliers decreasing over time, the median stays high in general and a only a small number of frames fall below the localization failure threshold. As mentioned earlier, localization failures do not necessarily lead to path-following failures as we can rely on VO.

Since the experiment path has parts that overlap with the training data and parts that fall outside the training data, we compare the number of inliers between the different sections. We present the box plots in Figures 6.48, 6.49, and 6.50, each for a different month. In the same way as for the first generalization experiment in Section 6.7.2, we see that the number of feature matching inliers is lower for the off-road data that falls outside of the training data compared to the part of the path that has overlap with the training data. At the same time, the inlier counts mostly vary in the same way across the different repeats. For the on-road data, we see that the area outside the training data get a slightly higher inlier count than the areas overlapping with the training data, which again is consistent with the generalization experiment. The new area is less affected by sun flare when the sun is low in the morning and evenings and the robot does not drive on grass as it does for part of the path that overlaps with the training data. We have seen throughout the experiments that feature matching across appearance change is easier on asphalt than on grass.

After looking at box plots of the inlier count across all repeats, we plot the CDF of live keyframes over the number of inlier matches for all repeats in Figure 6.51. This provides more detail about the inliers within each run. Because we have a lot of repeats, we colour each one either according to the number of days since the path was taught (a) or by the time of day that the repeat was completed (b). Although we see that the earliest repeats from September (dark blue) all have a high number of inliers (a), not all repeats towards the end of the experiment, in November, are among the ones with the lowest number of inliers. Clearly, the number of days since the teach is not the only indicator of how well features match. Consequently, we also look at the time of day, i.e. the lighting condition, under which



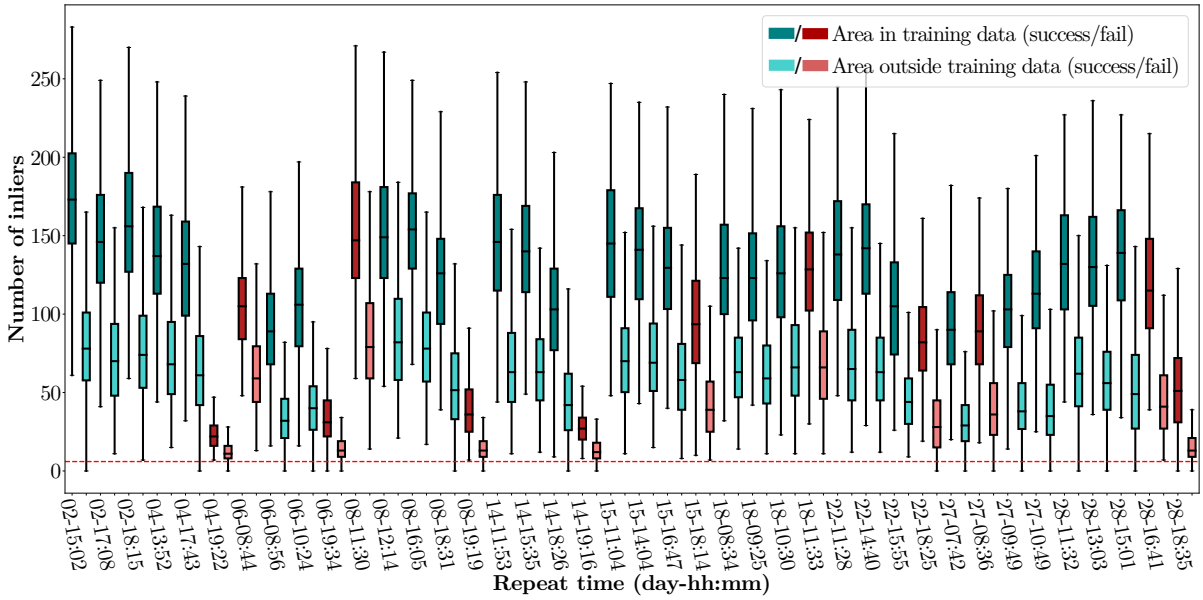
(a) Off-road.



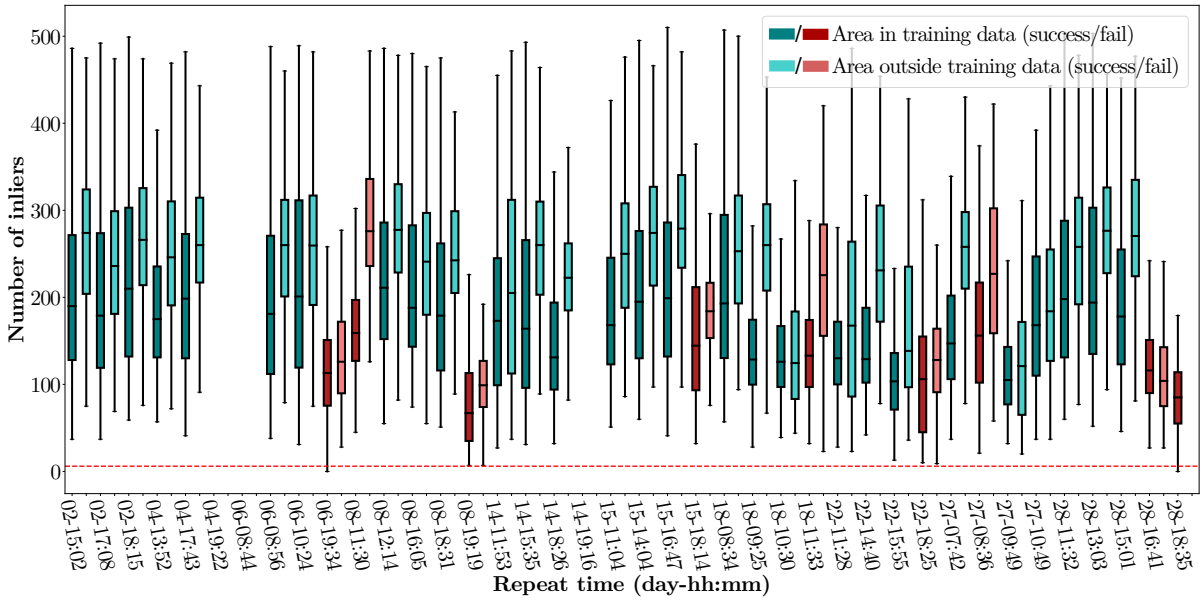
(b) On-road.

Figure 6.48: A box plot of the number of inlier feature matches divided for the areas inside and outside the training data. The inliers are computed for every run of the experiment during September listed in Table 6.7. The horizontal axis lists the date and time of day when the data was collected.

the repeat was completed (b). Although not all dark blue (early) or yellow (late) repeats are among the ones with the lowest number of inliers, a lot of the runs with poorer performance come from these times. Taken together, we see that the feature matching is affected both by seasonal change and the lighting condition when a repeat was completed. This is consistent with the box plots of inliers in Figure 6.47, where there is some fluctuation between runs at different time of day, but the number of inliers gradually tends downwards over the three months.



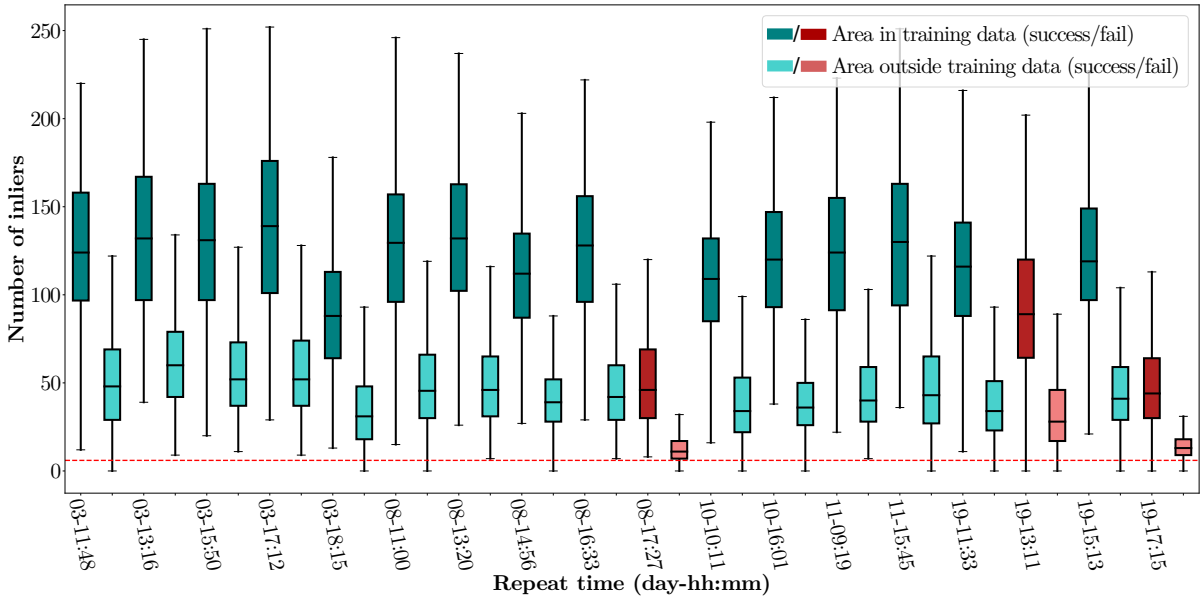
(a) Off-road.



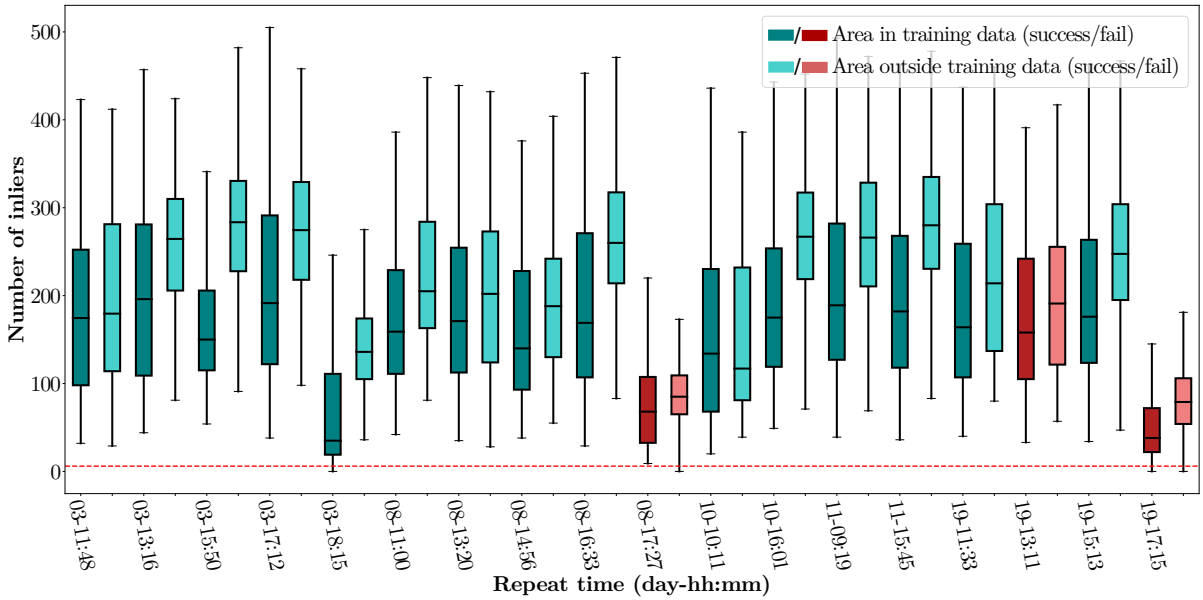
(b) On-road.

Figure 6.49: A box plot of the number of inlier feature matches divided for the areas inside and outside the training data. The inliers are computed for every run of the experiment during October listed in Table 6.8. The horizontal axis lists the date and time of day when the data was collected.Missing boxes in (b) are due to runs that failed before reaching he on-road of the path and did not restart.

Finally, we look at the number of inliers for map vertices along the path. In Figure 6.52, we have divided up the data by month. Consistently with Figure 6.38 from the first generalization experiment, we see that we generally get a higher number of inliers in the structured, on-road part of the path. The number of inliers is the lowest in the off-road area that was not included in the training data. The off-road area that was part of the training data (start and end of the loop below $y = 0$), does a little better. Relative to the on-road area, however, it does not maintain as high a number of inliers. Similarly, we can



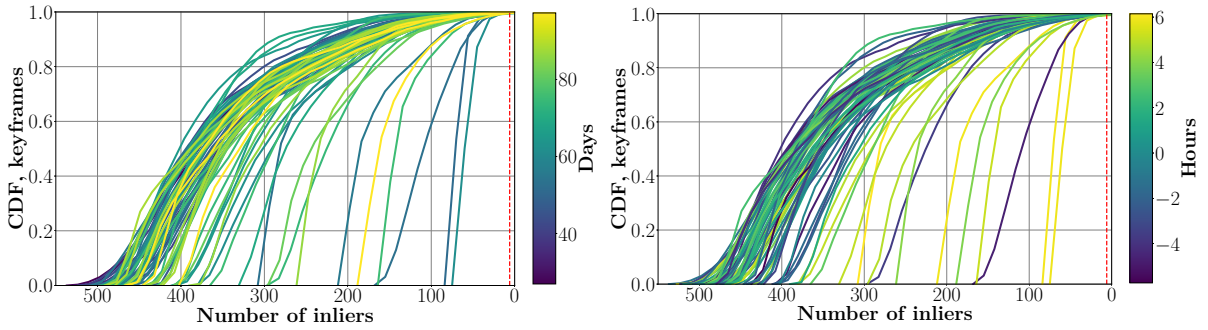
(a) Off-road.



(b) On-road.

Figure 6.50: A box plot of the number of inlier feature matches divided for the areas inside and outside the training data. The inliers are computed for every run of the experiment during November listed in Table 6.9. The horizontal axis lists the date and time of day when the data was collected.

see that the part of the structured area around the Mars Dome, where the robot drives on the lawn, now does worse relative to the areas driven on asphalt, than was the case in the generalization experiment. These observations point towards seasonal change being more difficult for the learned features in the off-road areas, and also when driving on the lawn by the Mars Dome. This can be explained by the fact that the vegetation in off-road areas makes seasonal changes to grass, trees, and bushes dominant, while the structured area contain roads, buildings, fences, storage units and so on that do not significantly



(a) Repeats coloured by days since teach.

(b) Repeats coloured by time of day of collection.

Figure 6.51: The CDF of keyframes with a given number of inliers for each repeat in the seasonal experiment. The red vertical line shows the localization failure threshold of six inliers. The repeats in (a) are coloured according to the number of days since the path was taught, while in (b) they are coloured by the time of day that path was repeated, given by the number of hours before or after the time the path was taught.

change over time. If we look at the plots over the different months, we see that the number of inliers gradually reduces over time for the whole path.

Distance on Dead-Reckoning

Since the number of feature matching inliers decrease over the seasonal experiment, we expect to have more cases of localization failures. where the robot relies on dead-reckoning. We plot the CDF of live keyframes for each repeat, where localization failures occurred, see Figure 6.53. Similarly to the feature inlier CDF, we provide two plots. In one we colour the repeat according to the number of days since the teach (a) and in the other by the time of day the repeat was completed (b). As explained earlier, the feature matching is affected how far along in the experiment we are, but also more "locally" by the time of day when the path was traversed. We get a similar behaviour for the distance driven on dead-reckoning. From the plots, we see that we rely on VO for approximately a quarter of the keyframes (in repeats where localization failures occur) and the longest distance on VO is less than 5 metres.

Autonomy Rate

The seasonal experiment is the first one, where we get path-following failures and therefore do not maintain a 100% autonomy rate. A path-following failure happens if we need to manually intervene to stop the robot because it has driven too far to the side of path and is not able to recover to get back on the path by itself. In some cases we had to stop the robot although it was not very far to the side of the path. However, it was driving in an area with, for instance, trees close to the path and would otherwise crash into them. In Tables 6.7, 6.8, and 6.9, we list the number of failures that occur for each repeat. Out of the 72 repeats, 15 (or 21%) contain one or more path following failures. The first path-following failure occurred on October 4th, 51 days after the path was taught on August 14th.

First we look at the conditions under which these failures happen. Nine of the runs with failures occur just as it is getting dark or after dark. We have a total of ten runs for this condition and so it is clear that the learned features struggle when seasonal change compounds with the most drastic illumination change. Four of the runs with failures happen when there is low sun causing sun flares and

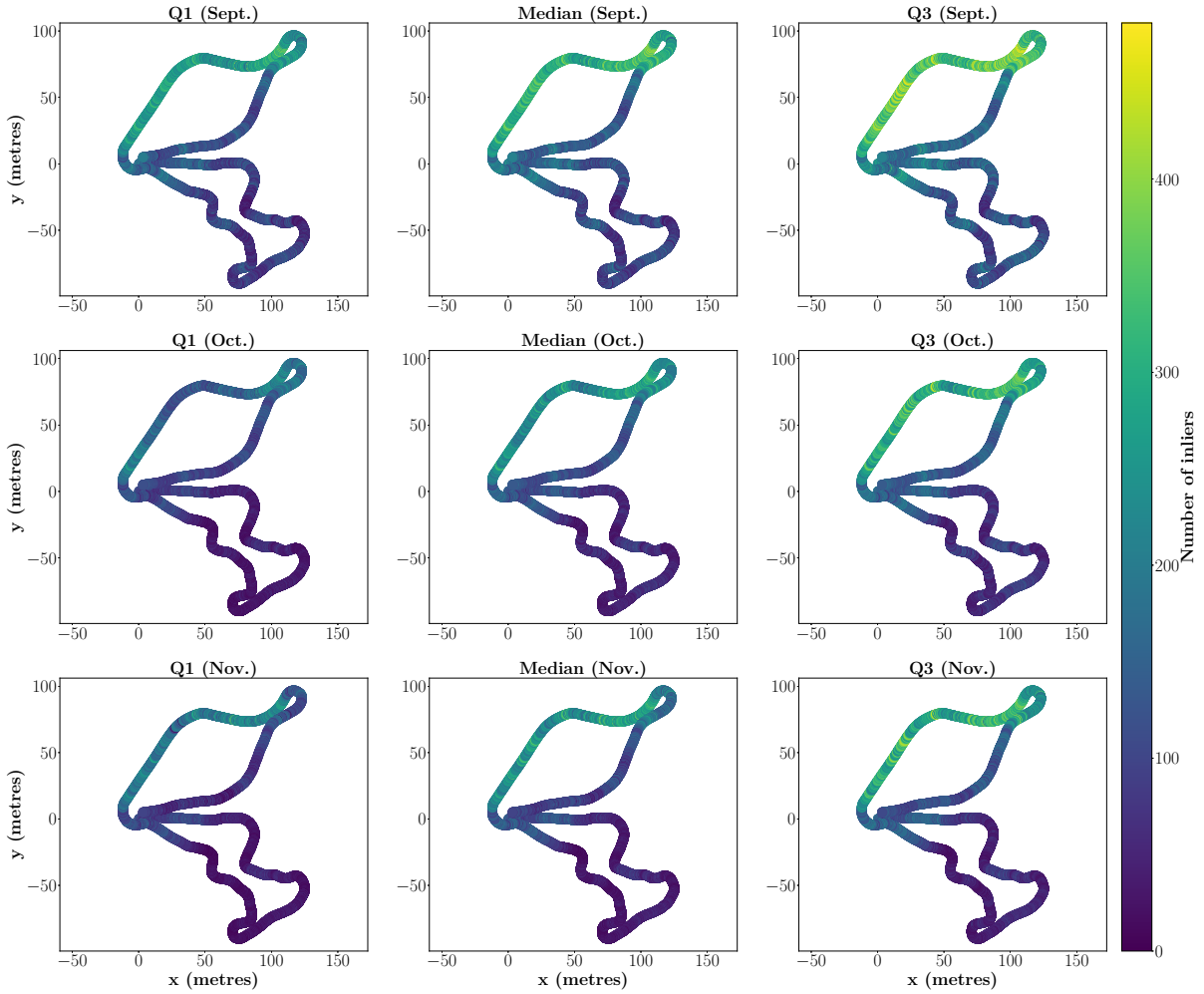
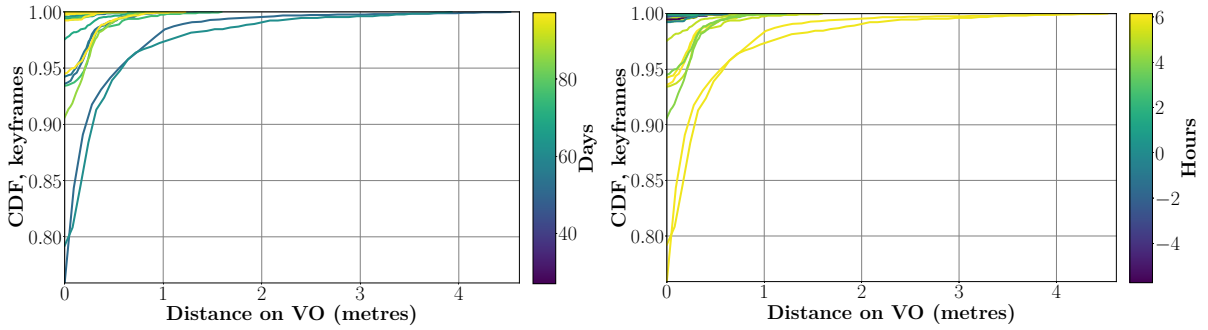


Figure 6.52: The plot shows the median number of inliers across all runs for each map vertex along the path, as well as the 0.25 (Q1) and 0.75 (Q3) quantiles. The inliers are plotted separately for each month.

long shadows. Again we see that that challenging illumination change on top of seasonal change makes localization more difficult. The robot repeats the path successfully a total of twenty times with low sun, indicating that VT&R completes most of the repeats under this condition, unlike the repeats after dark. Finally, the last two runs with failures happens once when there is some light sun flare and once during cloudy conditions. Hence, the lighting conditions are not very challenging in these two cases.

Out of the fifteen runs with failure, ten runs have one path following failure, three runs have two failures, one run has three failures, and one run has four failures. One run with two failures happened during low sun, while the remaining four runs with two or more failures happened after dark. The two very first times we encountered failures (once on October 4th and once on October 6th), we did not restart VT&R after the failure to continue the path following and so both these runs have one failure each. Afterwards, we updated our procedure and restarted VT&R after every manual intervention to continue the traversal. In two of the subsequent runs, on October 14th at 19:16 and on October 28th at 18:35, we were unable to continue after the failures due to difficulty reestablishing stable localization under challenging dark conditions.

In Figure, 6.54, we approximately mark the location of where path following failed along the path



(a) Repeats coloured by days since teach.

(a) Repeats coloured by time of day of collection.

Figure 6.53: A CDF over keyframes for the distance VT&R relies on dead-reckoning with VO to compensate for localization failures. The horizontal axis lists the distance VT&R has relied on dead-reckoning for a keyframe. The repeats in (a) are coloured according to the number of days since the path was taught, while in (b) they are coloured by the time of day that the path was repeated, given by the number of hours before or after the time the path was taught.

so that we can get an idea of whether failures have anything in common and, if so, which areas have the most failures. We will explain the errors in each location together. Let us assume that we start at the beginning of the path, indicated by the red arrow, and move along the path from there. The first failure is marked in green. In this failure the robot had only drifted about 0.5 - 1.0 metres to the left of the path, an offset from which VT&R with learned features can recover. We had to manually intervene, however, because the robot was about to drive into a large rock close to the side of the path. The next two failures occur on a turn. In both cases the alignment to the map (i.e., the map vertex considered the closest) suffers at the beginning of the turn resulting in the robot taking the turn too early. This causes the robot to drive into the bushes to its right and we had to manually intervene. In this particular turn, the teach images show that the ground has a lot of flattened dry grass. If we compare to the two failed repeats, we can see how much the ground has changed, see Figure 6.55. This causes a problem, especially, for night-time driving as the headlights mostly light up the ground in front. Few farther-away points that may have more visual stability will be lit up.

We move further along the path and look at the next eight failures, which all have the same root cause. Leading up to these failure locations, we have a longer, mostly straight, stretch of path. The live keyframe has poor alignment with the map at the end of this straight stretch and before entering the first turn. Specifically, the live frames localize to map vertices behind them on the path while the feature matching does not reflect this longitudinal offset well enough. This leads to the robot starting the turn too late, which subsequently causes it to drive into some bushes or too close to trees resulting in the failures seen in Figure 6.54. For run 64 (purple), we see multiple failures in this area. After the first failure, it was difficult to get a good alignment and start again, while operating after dark.

There can be a few reasons for the poor feature matching in the longitudinal direction in this area. Firstly, the straight stretch of path is fairly open with a lot of grass and some trees on the left side. This provides fewer stable points for localization that often lie near the horizon. The ground in front of the robot is subject to larger appearance changes. Furthermore, the area before the turn has large shadows in the map, which also makes feature matching more difficult, as seen in Figure 6.56. During the training of the learned features, we found that the longitudinal direction was more challenging to estimate well compared to the lateral offset and heading angle (concrete examples are provided in

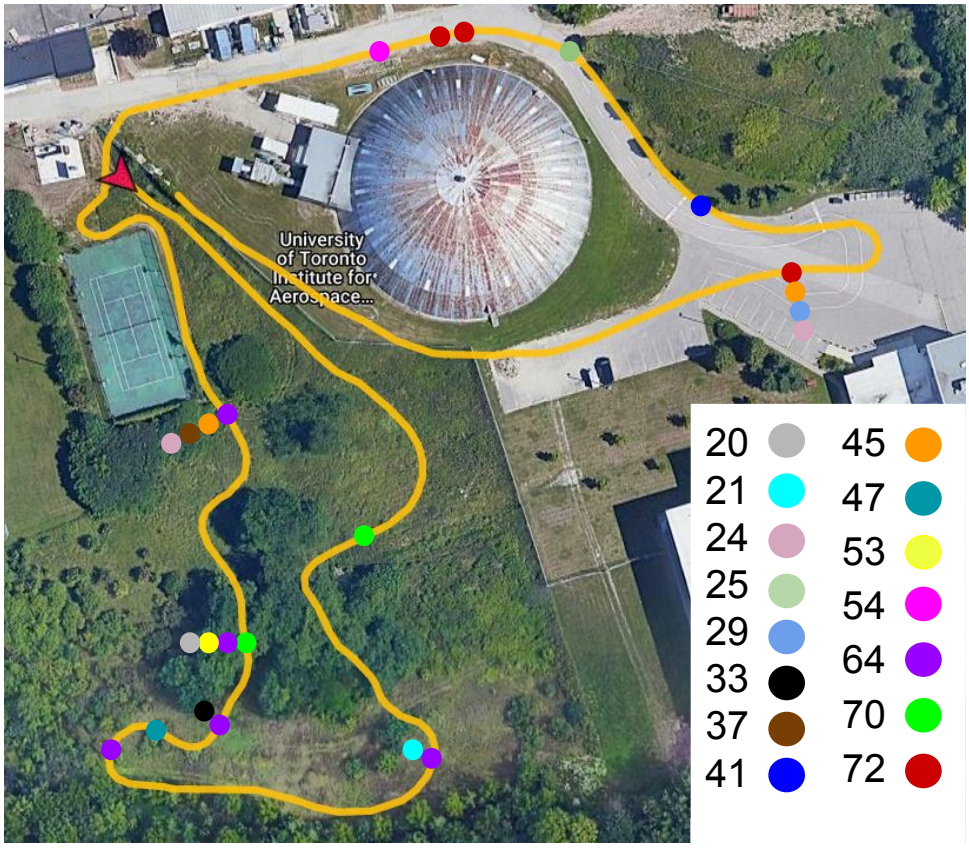


Figure 6.54: The image marks the approximate location of all the path-following failures that occurred during the seasonal-change experiment. Each coloured dot has a corresponding run ID, which can be looked up in Tables 6.7, 6.8 and 6.9 to find more information about the repeat. Five out of the total fifteen runs, have more than one failure.

Appendix A). Since VT&R looks at a window of frames for localization and relies on VO to propagate the pose guess forward, the longitudinal estimate does not have to be perfect in order for VT&R to localize and move the localization window forward. Lateral offset and heading are more important for successful path following.

A second issue compounds the problems in this case such that a poor longitudinal estimates results in poor localization to the map. When a path is taught, VT&R stores a vertex in the map approximately every 0.3 metres. If the number of feature matching inliers for VO drops below a threshold, VT&R will create a new vertex. If it is windy, an area with tall grass, bushes, and trees will be especially challenging for VO, and the number of matched features drops. In our case, it was windy when the path was taught, and VT&R created many vertices close to each other on this part of the path. During localization, we work with a fixed-size window of frames and cannot increase its size due to real-time performance considerations. All together, the non-ideal longitudinal estimate combined with having many vertices in the map, causes the closest vertex in the map (that we want to localize to) to "lag behind". While driving straight this does not matter, but it causes the robot to take the next turn too late. In order to fix this problem, we would look at ways to improve the learned features' performance in the longitudinal direction under challenging conditions. Furthermore, we could also improve the keyframe distribution in the map by adjusting the feature matching threshold to better fit the off-road environment.

Figure 6.55: We compare a map image (left) to images from the same location in the two failed repeats from October 6th at 08:44 (middle) and November 8th at 17:27 (right). The grass on the ground has changed significantly.

Figure 6.56: We compare a map image (left) before the turn to images from the same location in two repeats that failed on October 27th at 08:36 (middle) and October 14th at 19:16 (right).

The next four failures occur together in the same location just below the tennis court, see Figure 6.54. Here the robot drives through a narrow passage with tall branches on each side. In each of the failures, the robot has drifted a little too far left (about 0.5 metres) before entering this area, and we need to manually intervene so it does not drive into the bushes. Figure 6.57 shows what the area looks like in the map and from two repeats that fail.

Next, we move to the on-road part of the path. The first three (pink and red) and last four failures along the road have the same behaviour. The robot is repeating the path well without any significant lateral offset. We also get a consistently high number of feature matching inliers, which is expected in this structured area. Then suddenly the robot turns quickly to one side in a twist-like motion, and we manually intervene to stop the repeat. If we look at the last few live keyframes preceding the abrupt motion, the live features match to the map features in a way that is consistent with the expected motion. That is, the feature matches do not indicate that the predicted pose should suddenly change directions. All the failures happen on repeats as it is getting dark or after dark. After studying the VT&R output logs and the feature matches, we do not have a clear answer to the cause of this problem. In the logs, we get messages about uncertain poses not meeting constraints, but no clear and consistent indication of what precedes the issue. We would need to spend more time to debug to find the underlying problem.

The last two failures are for runs 25 (green) and 41 (blue), which were driven on October 8th at 11:30 and on October 18th at 11:33, respectively. These stand out from the others because they are the only ones that occurred under nominal lighting conditions in the middle of the day. Both failures happened along the road in a structured environment and we were getting a normally high number of feature matching inliers. In both cases, the robot gradually drifted off the path and did not return to

Figure 6.57: We compare a map image (left) from the start of the narrow path segment to images from the same location in two repeats that failed on October 15th at 18:14 (middle) and October 6th at 19:34 (right).

the path before we manually intervened to stop the traversal. When looking into the localization of the live frames, and comparing to a successful run in the same area under similar conditions, the live frames of the failed runs were not localized equally well to the map. While the successful run had a good estimate of the closest vertex in the map for localization, the two failed runs were localizing to map vertices slightly ahead and slightly behind the true closest vertex, respectively. Going into a soft turn, the offset to the map caused the robot to start drifting slowly with respect to the path and was not recovering immediately. When the path tracker issues velocity commands to the robot, it balances the scheduled speed and the commanded speed, where the latter is computed based on the pose estimate. Weighting the influence of these two values affects how the robot drives and must also be adjusted based on the desired operating speed of the robot. We think that by better adjusting this parameter, we can make the robot react faster to the commanded speed and better adjust for such slow drifts.

In summary, we have looked at the different failures in path following over a long-term experiment spanning three months. We found that the features struggled in some situations when seasonal change was compounded with large illumination change such as driving after dark and when the sun was low in the sky. Looking more closely at the failures has given us some pointers to future improvements for both the learned features themselves and their implementation in VT&R. In particular, we want to explore ways to improve the features' performance in estimating pose for the longitudinal direction. We can also make adjustments to parameters in VT&R to reduce the number of map vertices created in off-road areas, and to improve the velocities sent to the robot during autonomous repetition.

Conclusion

In the seasonal change experiment, we have repeated the path taught on August 14th for three months until November 19th. The experiment provides not only areas that are spatially different from the training data, but also unfamiliar appearance conditions through late summer and autumn. The results show we have generally been able to keep a high number of feature-matching inliers, but that the number of inliers has been gradually decreasing during the experiment. We encountered path-following failures, showing that the compounding effect of large lighting change (driving after dark, low sun) and seasonal change caused challenges for the learned features in 15 out of 72 runs. As in the first generalization experiment, feature matching is easier in an area with asphalt road and surrounding permanent structures, compared to off-road conditions with vegetation that changes continually.

6.8 Conclusions and Future Work

In this chapter, we have applied the learned features for relative pose estimation presented in Chapter 5 to long-term localization in VT&R. With these learned features, we have been able to improve on the learned pose regression presented in Chapter 4, where we computed the relative pose between images from different appearance conditions directly using a neural network. Although we were able to tackle large appearance change, the method did not generalize well to new paths. In this chapter, we have shown that the learned features can both handle localization across large appearance change and generalize to new paths not seen in the training data. Moreover, we complete closed-loop path following with VT&R without the need for intermediate bridging experiences.

We started our experiments by comparing the learned features to other state-of-the-art methods and showed that our approach had the fastest inference time and second lowest memory consumption, which are both important to real-time operation. We introduced the learned features into the VT&R system and conducted three closed-loop experiments. The networks used were trained using the UTIAS In-The-dark and UTIAS Multiseason datasets, which were gathered four and five years prior, respectively. The first closed-loop experiment shows that we can repeat a path in a familiar area across a full range of lighting change, including driving after dark, with a 100% autonomy rate. In our second experiment, we again showed that the learned features can tackle a full range of lighting-change with a 100% autonomy rate, however, we also demonstrated that they generalized when we repeated paths containing unfamiliar areas. Beyond new areas, the experiment also contained seasonal appearance conditions that differed from the training data. In the third, and final, experiment, we operating on the path we taught in mid-August until mid-November. This experiment also included unfamiliar areas and seasonal appearance conditions. We were able to maintain a high number of feature matching inliers across the three months, but did not achieve a 100% autonomy rate. Compounding seasonal and large lighting change causes path following failure in 15 out of 72 repeats (or 21%).

The experiments tell us that learning visual features for use in a classical pose estimator clearly outperforms the method that tried to learn the whole pose estimation pipeline. Moreover, learning features makes it easy to integrate the approach in the existing VT&R system by simply adding another feature implementation alongside the existing SURF. Finally, our neural network feature detector is fast and uses low enough memory that we can operate in real-time on a robot. To the best of our knowledge, Sun et al. (2021) provide the only other example of testing learned features outdoors on a ground robot for long-term localization. Their experiments are much less extensive, only testing day to night localization over a small time frame on two paths in structured environments (a parking lot and by a church).

Future work can be done to improve the learned features for long-term localization. As seen from our last experiment, the learned features need improved robustness conditions when large seasonal and lighting change occur simultaneously. This would include work to improve the feature matching performance in the longitudinal direction. The experiments can be extended to test against larger seasonal change like snow, or to test generalization in new environments different to those we encounter at UTIAS. Finally, we can also do more systematic experiments to determine the learned features' robustness to being off the path and to find out at which point the viewpoint change becomes too large.

As we have discussed earlier, the VT&R datasets have a lot of samples with small viewpoint change. Therefore it would be beneficial to spend time to improve the resulting training dataset. Possible avenues include re-balancing the data, changing the sampling to allow for larger relative poses, or artificially augmenting the samples. Naturally, it would also make sense to try unsupervised learning without using

ground truth poses at all. This would get around our issue of computing poses in $SE(2)$ instead of $SE(3)$ due to lack of accuracy in some runs of the dataset.

In terms of code method implementation, we can update the learned features in VT&R to use the learned scores during pose estimation. Furthermore, we can implement dense matching in VT&R and compare it to sparse nearest neighbour matching. Further work such as improved parallelization should also be done to make the implementation more efficient so that we can drive faster. Brief testing with a newer and faster laptop with a larger GPU has already shown us that the features work well when driving at 1.5 m/s during nominal lighting conditions on consecutive days. However, we should also test that this holds for more challenging appearance conditions and possibly higher speeds. Finally, we want to tune some parameters in the path-tracking controller to improve driving behaviour for our system.

6.9 Novel Contribution

- We learn visual features for localization and incorporate them in the existing VT&R path-following pipeline.
- We are the first to complete autonomous path following with learned features across a full range of lighting change, while operating in unfamiliar areas not seen in the training data, and across seasonal change.
- We are the first to perform autonomous path following in off-road environments with abundant vegetation and few permanent structures and show in our experiment that these are more challenging for localization than structured scenes.

6.10 Associated Material

Publication

Gridseth and Barfoot (2022). Keeping an Eye on Things: Deep Learned Features for Long-Term Visual Localization. In the *2022 IEEE Robotics and Automation Letters (RAL)*.

Code

The code for training and testing deep learned features for localization in PyTorch is available at https://github.com/utiasASRL/deep_learned_visual_features

Videos

- Deep learned features: https://youtu.be/mqDivnIQjl0
- Deep learned features (conference presentation): https://youtu.be/JsEBs9Y5XVg
- Deep learned features and VT&R3 (demo): https://youtu.be/ZYkxevdw7DA

Chapter 7

Conclusions and Future Work

In this chapter we summarize the contributions made in this thesis. We also discuss possible avenues for future work.

7.1 Thesis Summary

In this thesis we have worked towards increasing the robustness of visual long-term localization in VT&R without relying on intermediate bridging experiences. Our aim was to tackle visual localization across lighting and seasonal change, especially in challenging off-road environments. We started our work, in Chapter 3, by using a direct method together with a robust cost function for pose estimation in localization. The direct method used all pixels in the image with a sufficiently high gradient and avoids the work of extracting and matching sparse features. We showed improved image alignment under nominal lighting conditions in a challenging scenario with tall moving grass and other vegetation. The use of the robust cost function, however, was not sufficient to tackle large illumination change. The results led us to shift focus to deep learning as a tool to tackle large appearance change in images.

In our first learned approach, described in Chapter 4, we trained a deep neural network to directly regress relative pose between two images. We used the same network architecture to regress relative pose both for VO and localization, by training networks with different ground truth poses. Deep learning enabled us to localize across a full range of lighting change in a structured environment and across large seasonal change from snow cover to green grass in an off-road environment. Although the approach handled large appearance change, it did not generalize to new paths not seen during training.

In Chapter 5, we aimed to add more structure to the learned method. Instead of learning the whole pose estimation pipeline, we learned sparse visual features that can be used with a traditional pose estimation approach. In other words, we used deep learning for the challenging front-end, while relying on an existing method for the part of the pipeline that already has well-known solutions. Moreover, the entire pose estimation pipeline is differentiable, which allows for end-to-end training with ground truth poses. We showed that the learned features can be used successfully for VO in a selection of on-road and off-road environments under different conditions such as driving during the dark or with snow on the ground. More importantly, the learned features also generalized to new paths not contained in the training data, which is an improvement over the previous work from Chapter 4.

The final part our work, in Chapter 6, applies the learned features to localization, which is a more

challenging task due to the potential appearance change between image frames. Unlike most previous work, we did not only test the features for stand-alone localization in datasets. Instead, we introduced the learned features into the full VT&R system that combines VO, localization, and path tracking to facilitate closed-loop path following with the Clearpath Grizzly robot. We completed closed-loop experiments with a total of 87.8 km of autonomous driving on both on-road and off-road paths. The experiments showed that the learned features achieved path following with a 100% autonomy rate across a full range of lighting change, including driving after dark, without relying on intermediate bridging experiences. Furthermore, we tested localization for three months of seasonal change from mid-August until mid-November. In this experiment 21% of runs experienced one or more path-following failures, mainly due to the compound effect of seasonal and large lighting change from driving after dark or during low sun. Finally, the closed-loop experiments also demonstrated that the learned features generalized to new paths not observed during training. From our analysis of feature inliers and path-following failures, we see that localization across large appearance change is more challenging in an off-road environment dominated by vegetation compared to a on-road environment with asphalt-covered road and surrounding permanent structures.

7.2 Novel Contributions

The following novel contributions, towards using deep learning for long-term localization in VT&R, have been achieved in this thesis.

- We replaced sparse feature-based visual localization in VT&R with a direct method and were the first to demonstrate direct localization in a challenging off-road example with tall grass and dense vegetation, but limited appearance change (Gridseth and Barfoot, 2019).

- We regressed relative pose for single-experience visual localization across large appearance change directly from a pair of images using a deep neural network. We also regressed poses for VO with the same network architecture and combined localization and VO for visual path following. We were the first to tackle large seasonal appearance change in a challenging off-road environment with deep learned pose regression (Gridseth and Barfoot, 2020).

- We estimated the pose for single-experience visual localization using sparse features extracted with a deep neural network. We included the learned features for localization into the VT&R pipeline and were the first to achieve real-time autonomous path following across a full range of lighting change with a 100% autonomy rate with deep learned features. We extended the experiment to include three months of seasonal change from summer through autumn, where 79% of runs have no path-following failures. We demonstrated successful generalization to new areas not seen in the training data (Gridseth and Barfoot, 2022).

7.3 Future Work

We have demonstrated that we can use learned features for real-time single-experience localization for closed-loop path following with VT&R. There are improvements that can be made in the design and implementation of our method as well as further testing that can be completed. In this section, we

suggest improvements and possible avenues for future work. We divide into future work relating to the data used for training and testing, the algorithm behind our approach, the implementation of learned features in VT&R, and experiments.

7.3.1 Data

As discussed in previous chapters, we have used the UTIAS Multiseason and UTIAS In-The-Dark datasets to train our learned methods. The data were collected by previous students while using multi-experience VT&R for autonomous path following with the Clearpath Grizzly robot. Since VT&R accomplishes accurate path following with centimeter-level error on kilometer-scale repeats (Clement et al., 2017), a lot of the available live and map image pairs have a small viewpoint offset unless the the data is from a turn in the path. We partially remedied this issue by picking pose graph vertices that were topologically farther apart to get larger variation in distance between frames and also increase viewpoint change. More work can still be done to improve the training data. One option is to re-balance the training data set such that we get a higher proportion of samples that have a larger viewpoint change. Another possibility is to synthetically augment the training samples by applying a transform to the ground truth pose and warp the image accordingly. A more sophisticated approach to augmenting training data could be to use a neural radiance field (NeRF), which is a fully connected neural network that generates novel synthetic views of 3D scenes. Training a NeRF requires sets of images of an input scene and interpolates between them for rendering. Therefore we would need additional data with more varied viewpoints to train such a network. A final option would be to teach a path and then repeat with offsets to increase viewpoint change. That way we could force larger offsets to the path for the sake of data gathering.

Instead of using VT&R to generate the pose ground truth, it is also possible to rely on GPS. In order to get accurate measurements, we need to combine the GPS on the Grizzly robot with an RTK base station, which takes a lot of work, and the signal may be impeded by tree cover or large distances. Moreover, the GPS estimates the robot's position but does not provide orientation information. Alternatively, we could imagine using only the GPS on the robot for approximate position estimates. In this case, we would have to consider how much influence a pose loss would get at training time compared to other signals.

A question we have not explored in this work, is how much data is required to train visual features for localization. The data we use takes time to gather, especially since we rely on ground truth poses for supervised training. Could we get away with using less data than we have so far? How much data do we need in order to capture different environmental conditions and different scenes?

7.3.2 Algorithm

There are many options to further improve on the learning method we are using, as well as opportunities to try different approaches altogether. As mentioned in Sections 6.7.4 and 6.8, we especially want to make improvements to the longitudinal part of our pose estimates on straight stretches of the path. We can explore adjustments to the network design and do a thorough comparison of performance in feature matching as well as run-time. This comparison of run-time and memory consumption should be conducted in the VT&R pipeline on a laptop to gauge the real-world performance on the target system.

Since Barnes and Posner (2020) work with birds-eye view RADAR data, they do not face the com-

plication of scale present in images. We want to explore whether different ways of computing descriptors can improve the learned features' performance in longitudinal pose estimation. Currently, we extract the output of different encoder layers and concatenate them together to form descriptors. This results in descriptor vectors with few elements from the shallow layers with high spatial resolution and many elements from the deeper layers with low resolution. In our method, it is possible that the deep layers become dominant. Although deep features may provide long-term robustness, features from layers with higher spatial resolution could be useful to facilitate accurate metric localization. Recall from Section 5.2.1 that D2-Net (Dusmanu et al., 2019) faces issues with accuracy because the descriptors and keypoints are both extracted at a deeper layer with lower resolution. There are several approaches to descriptor extraction and keypoint detection worth exploring. One simple option involving minimal change, is to apply a convolution to the encoder outputs before concatenating them together. That way we can adjust the size of and better balance each layer's contribution to the final descriptor vector. Since the direct methods such as GN-Net (von Stumberg et al., 2020a) or BA-Net (Tang and Tan, 2019) estimate pose at each spatial resolution of the network, they also extract descriptors at each level. S2DNet (Germain et al., 2020) also extracts descriptors at each level. In the same way as our method, the authors match sparse source descriptors to dense descriptors in the target. This leaves them with correspondence maps from different levels that they can then aggregate together. ASLFeat (Luo et al., 2020) remedies extracting the descriptor at a deep layer, with multilevel keypoint detection, while R2D2 (Revaud et al., 2019) uses dilated convolutions to maintain spatial resolution.

We have relied on supervised learning using relative poses for ground truth. Such data is time consuming to collect because we must estimate the ground truth poses, where possible options include using: a GPS-INS system, a different sensor such as LiDAR, or the camera sensor together a classical pose estimation algorithm like multi-experience VT&R. The other option is to work with unsupervised learning instead such as Tang et al. (2020) that learn visual features for VO or Yoon et al. (2021) that learn features LiDAR odometry.

When training the network that produces visual features, we supply one pair of image frames and one relative pose for supervision. During closed-loop operation, VT&R relies on a window of frames for localization and pose estimation. A natural next step would be to align our training method more closely with the final use of the learned features and hence use a sequence of frames during training as demonstrated by Yoon et al. (2021).

Although we compute dense descriptors and scores for an image, our work detects keypoints and uses sparse features for localization. Another is to combine dense descriptor and score learning with a direct localization method, like the one described in Chapter 3. This approach has shown promising results for localization across appearance change in outdoor urban environments (von Stumberg et al., 2020a,b; Kasper et al., 2020). Since direct pose estimation uses the whole image, it has the advantage of avoiding sparse keypoint detection and feature matching. The downside is that including the the inverse compositional model for pose optimization (as described in Chapter 3) in VT&R, would require a large change to the localization pipeline.

As discussed in the previous section, it would be interesting to further investigate what type and amount of data we need for learning robust visual features. To that end, online learning is a relevant avenue to explore. If we have a set of data to train visual features, could we use data collected during robot operation to further train the neural network and improve the visual features? In particular, would this be helpful when moving operation to a new and unknown environment? Pan et al. (2020) use online

imitation learning to train a network to provide control outputs from image inputs. In particular, they improve performance over time by using data collected during autonomous operation to learn to drive faster and more accurately. Thomas et al. (2021) use data from each session of a robot navigating in a simulated environment to train offline and improve their algorithm for LiDAR point cloud segmentation. As we saw in Section 5.2.3, Tang et al. (2020) improve learned features for VO over time by having the VO back-end label data, which they use for further training.

7.3.3 Implementation

There are several improvements that can be made to our initial implementation of learned features in VT&R. As discussed in Section 6.4.3, we use the existing sparse nearest neighbour feature matching in VT&R to match the learned features from different images. During training we matched features densely and computed target pseudo points. We could implement this in the VT&R pipeline, though one possible downside is increased memory usage from keeping dense descriptors around after the feature extraction step. Furthermore, we also did not have enough time before closed-loop experiments to include the learned scores as weights in the pose estimation. From the experiments, we saw that the pose estimation worked well, but including the scores has the potential to improve performance further by better weighing the importance of different points.

The learned features added additional work to the VT&R pipeline, and we had to slow down the robot from 1.0 m/s to 0.8 m/s for localization to keep up (as explained in Section 6.4.3). Although we were able to drive at 1.5 m/s during a limited experiment with a newer Lenovo laptop with a 12 GB GPU and faster processing, it would still be worth making code updates to improve processing speed. Currently, learned features are detected for incoming frames from the camera. Localization, however, is not done for every frame, but instead only for new keyframes. With some work, it should be possible to adjust the visual processing pipeline such that we only extract learned features for keyframes before they are passed to localization.

We use the VT&R version 3 (VT&R3) software implementation of the VT&R system, which has seen a complete update from the previous VT&R2. Some of these changes facilitate improvements in processing time for VT&R with learned features. VT&R3 now has the option to be more flexible about when localization is run. Previously, the VT&R pipeline would run localization for all new keyframes. The new code allows the system to skip localization when delays occur, but it also allows VT&R to localize frames that are not keyframes. This flexibility may provide relief during potential slow-downs. The second large change allows for easier parallelization of tasks. Parallelizing more tasks in the visual processing pipeline could be a useful step to improve run time when using learned features.

7.3.4 Experiments

The experiments we conducted can be extended to investigate the same topics in more detail, and we can add new experiments. The seasonal experiment can be run for a longer time to include larger changes. For instance, we did not get snow in time before the experiment was concluded. Work can also be done to further explore the network's ability to generalize to new environments. Although we added new paths with new appearance, all our tests were conducted at UTIAS. If we wanted to test generalization to a more drastically different environment, we would have to test in a new location.

Beyond expanding the experiments for seasonal changes or generalization performance, it would also

be beneficial to more thoroughly test the learned features robustness to faster driving as well as being off the path. Since we had to slow down the robot in the initial implementation of learned features for VT&R, we have only tested the features while the robot drives at 0.8 m/s. Our limited test of driving 1.5 m/s under nominal lighting conditions makes us hopeful that the features will also work at the same speed under more challenging appearance conditions. This can, however, only be confirmed by driving at higher speeds under a full range of difficult scenarios.

During our experiments, the robot most often recovered from driving off the path unless we had to manually intervene because an object was in the way. However, we did not have the time to do a more rigorous test to uncover the features' exact level of robustness to viewpoint changes caused by moving away from the path. We could conduct an experiment solely focused on recovering from off-path motion, where we test gradually from what distance the robot is able to recover.

Appendix A

Network Training and Testing

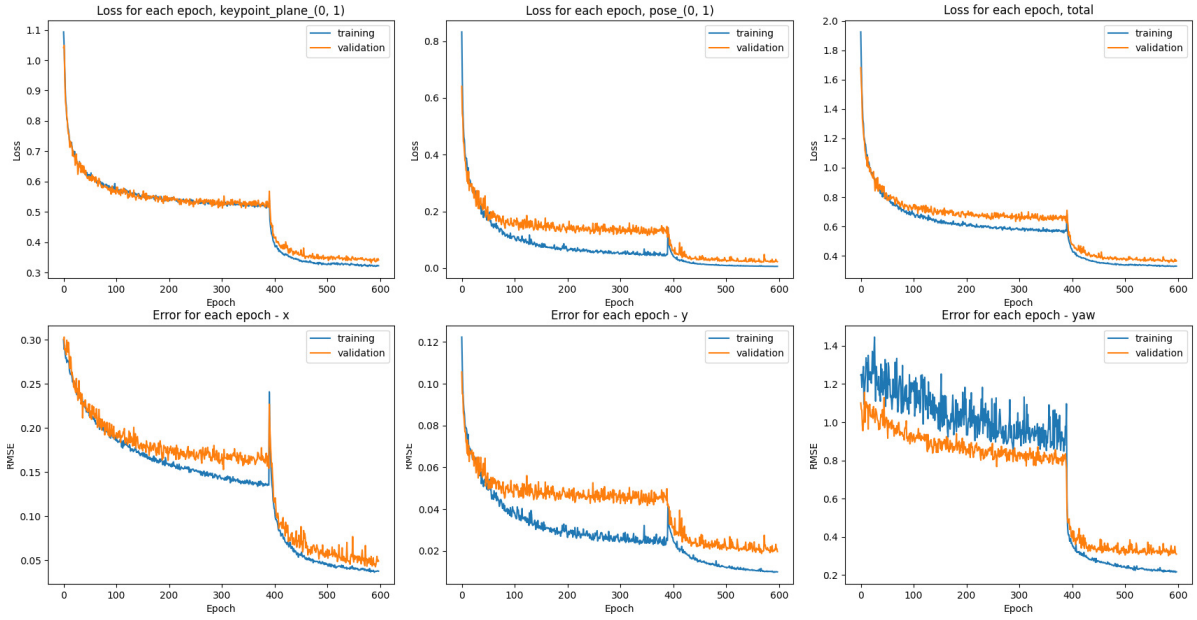


Figure A.1: The top row of plots show the training and validation loss for one of the networks we have trained. We display the keypoint loss, the pose loss, and the combined total loss. The losses for each 'epoch' listed along the x-axis, are computed after parsing 10,000 samples randomly chosen from the training dataset and 2,500 samples randomly chosen from the validation dataset. The network was trained for 391 epochs on the UTIAS Multiseason data, before training on the UTIAS In-The-Dark dataset. The switch between datasets causes the sudden change in loss values after epoch 391. The bottom row shows the training and validation error in each DOF. We display longitudinal (x), lateral (y), and heading (yaw) errors. The vertical axes are in metres for x and y and in degrees for yaw.

In this section, we provide additional results from training and testing one of our networks to learn features for localization. The network was trained on both the UTIAS In-The-Dark and UTIAS Multiseason datasets as explained in Section 6.4.2. We started by training the network on the UTIAS Multiseason dataset and then fine-tuned with samples for the UTIAS In-The-Dark dataset. We trained on a total of $391 \times 10{,}000 = 3.91$ million samples from the UTIAS Multiseason dataset, which took approximately 14.5 days, and another $194 \times 10{,}000 = 1.94$ million samples from UTIAS In-The-Dark, which took a bit over 6.5 days. Figure A.1 shows the training and validation loss as well as the RMSE for

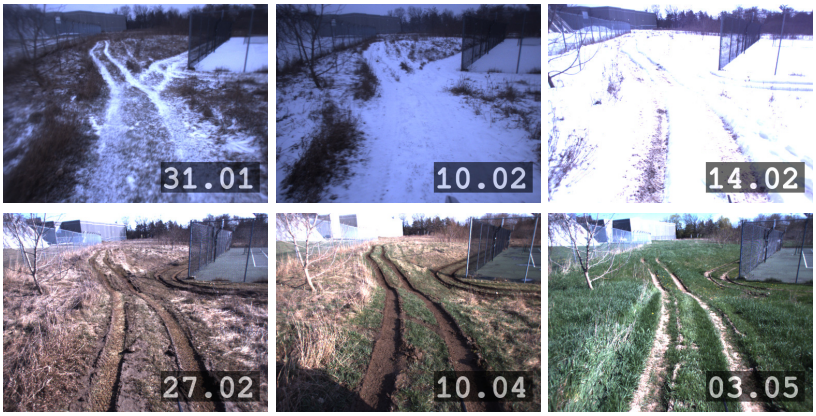


Figure A.2: Images taken by the robot from the held-out runs of the UTIAS Multiseason dataset that we use to test the learned features. Each image is labeled with the collection date (dd.mm) of the corresponding run. The images on the top row are localized to the corresponding image on the bottom row, with appearance change difficulty increasing from left to right.

each pose DOF separately. The training loss tells us whether training converges, while the pose errors show whether training actually improves pose estimation.

Before using the learned features in closed loop with VT&R, we needed to determine if they can provide sufficiently accurate pose estimates. This can be done by using held-out runs in the test set and computing poses in sequence for complete runs. We use the test runs from the offline experiment in Sections 6.5.3 and 6.7.1, that compares the performance of different feature-learning approaches on held-out runs from the UTIAS Multiseason dataset. We test the different conditions against each other by picking one run to be the teach and another to be the repeat. The pose estimation is done in PyTorch with our learning pipeline. This means that we are only using one pair of stereo images and do not use on a window of several stereo image frames. Moreover, we test localization standalone without a prior from VO. In Figures A.3, A.4, and A.5, we plot the estimated and target poses in individual DOF for selected pairs of runs. We choose three pairs of runs with increasingly difficult appearance change, where the last experiment localizes winter with snow cover to spring with green grass. Images from the robot camera that illustrate the conditions for each run are shown in Figure A.2.

Since we are doing basic pose estimation with one pair of frames at the time, we do not expect our predicted poses to be as smooth as the targets that are computed by the full VT&R system, which uses a window of frames and VO during localization. From the plots, we see that the estimated poses for the DOF that matter most for localization (lateral translation and heading), hover around the ground truth. The estimates, however, are less smooth than the targets and have a few large outliers. When the features are used in the VT&R pipeline, we believe that the additional checks on feature matching as well as the more advanced localization pipeline, will result in smoother pose estimates. From the plots, we see that the longitudinal direction is the difficult to estimate. Given that VT&R combines localization with VO, longitudinal estimates are less important than lateral offset and heading for path following. We believe that the learned features will work in closed-loop with VT&R.

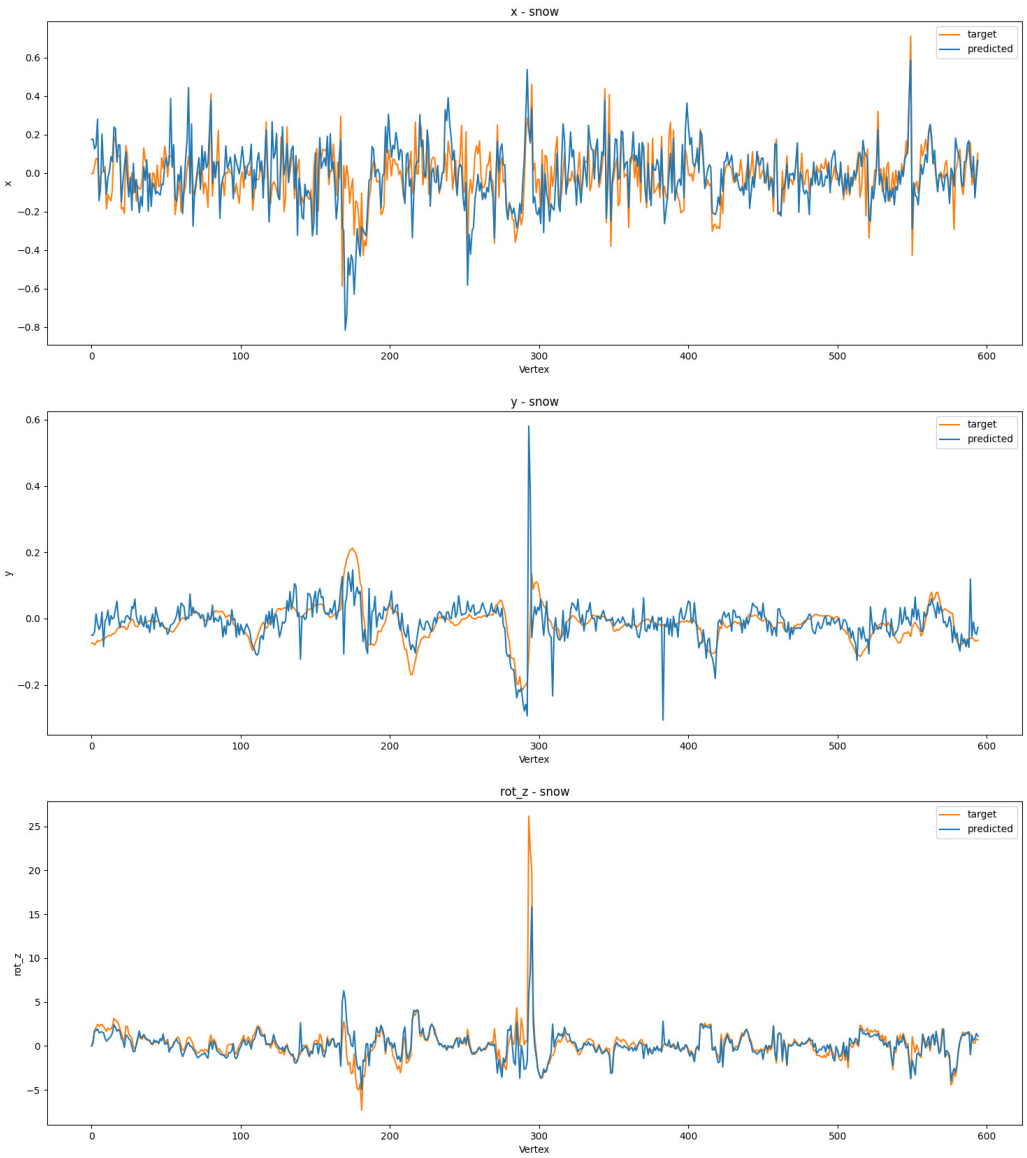


Figure A.3: These plots show the estimated and target poses in individual DOF for localizing the run from 31.01 (dd.mm) to the run from 27.02. We display longitudinal (x), lateral (y), and heading (rot_z) values. The vertical axes are given in metres for x and y and in degrees for heading. The horizontal axis list the index of the map vertex along the path.

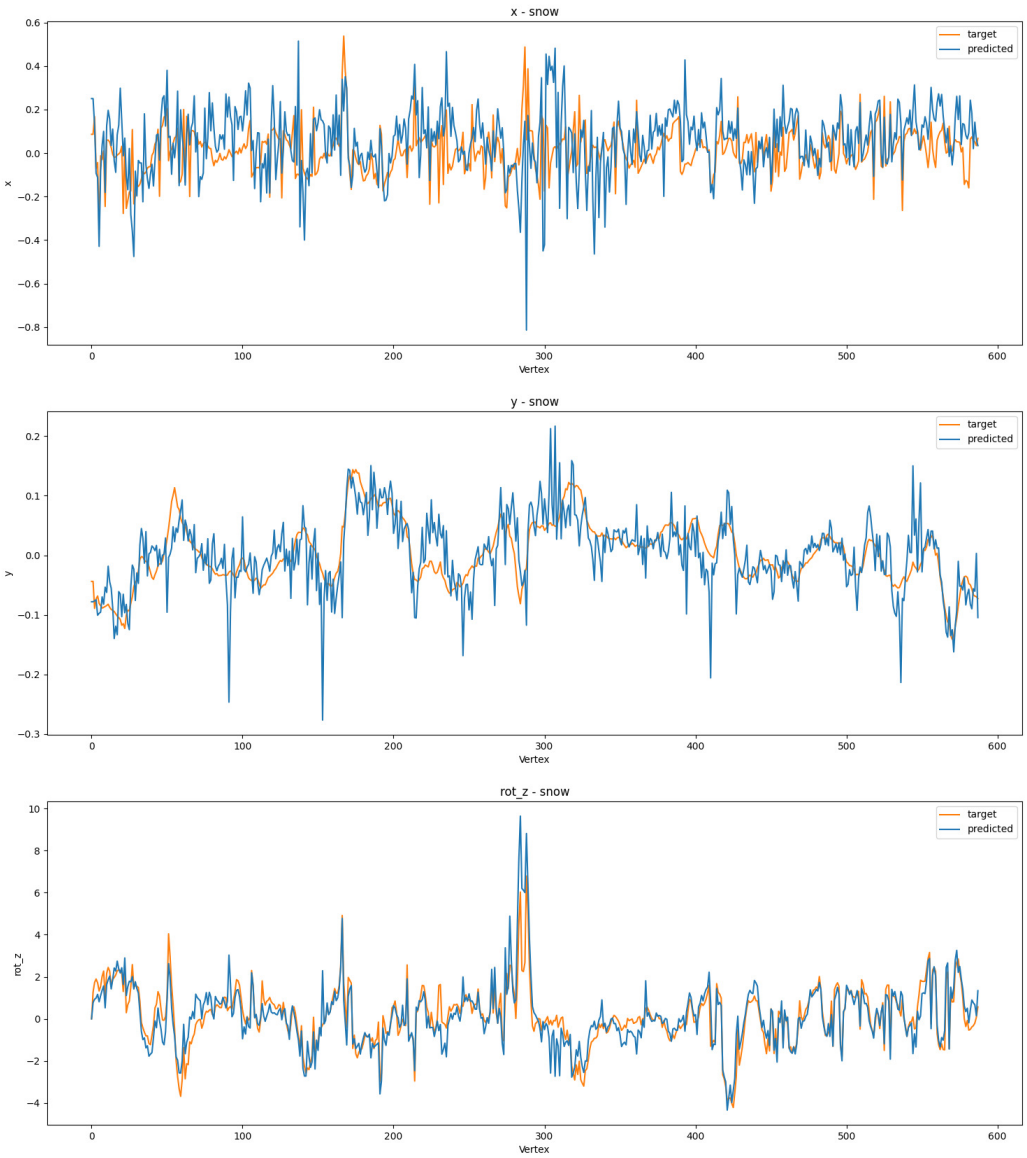


Figure A.4: These plots show the estimated and target poses in individual DOF for localizing the run from 10.02 (dd.mm) to the run from 10.04. We display longitudinal (x), lateral (y), and heading (rot_z) values. The vertical axes are given in metres for x and y and in degrees for heading. The horizontal axis list the index of the map vertex along the path.

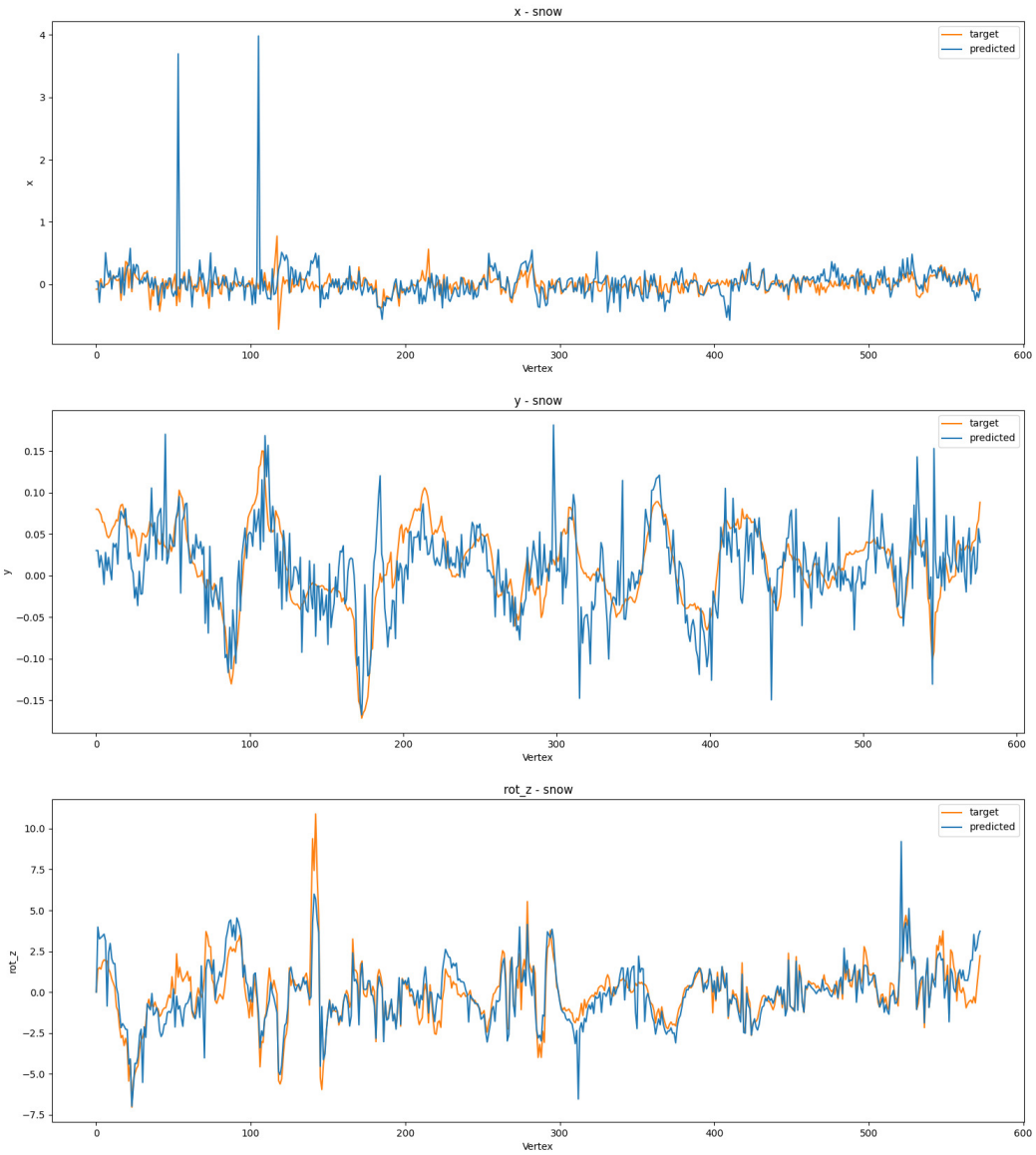


Figure A.5: These plots show the estimated and target poses in individual DOF for localizing the run from 14.02 (dd.mm) to the run from 03.05. We display longitudinal (x), lateral (y), and heading (rot_z) values. The vertical axes are given in metres for x and y and in degrees for heading. The horizontal axis list the index of the map vertex along the path.

Bibliography

Alismail, H., Kaess, M., Browning, B., and Lucey, S. (2017). Direct visual odometry in low light using binary descriptors. *IEEE Robotics and Automation Letters (RAL)*, 2(2):444–451. (ref. page 29)

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5297–5307. (ref. pages 42 and 73)

Badino, H., Huber, D., and Kanade, T. (2011). The CMU Visual Localization Data Set. `http://3dvis.ri.cmu.edu/data-sets/localization`. (ref. page 26)

Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1090–1097. (ref. pages 28, 29, 31, and 33)

Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255. (ref. page 29)

Balntas, V., Li, S., and Prisacariu, V. (2018). Relocnet: Continuous metric learning relocalisation using neural nets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 751–767. (ref. page 43)

Barfoot, T. D. (2017). *State Estimation for Robotics*. Cambridge University Press. (ref. pages 9, 11, 12, 13, 15, and 32)

Barnes, D. and Posner, I. (2020). Under the radar: Learning to predict robust keypoints for odometry estimation and metric localisation in radar. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9484–9490. (ref. pages 53, 54, 56, 57, 58, 59, 60, 61, 64, 70, 74, 75, 76, 86, and 135)

Bateux, Q., Marchand, E., Leitner, J., Chaumette, F., and Corke, P. (2018). Training deep neural networks for visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. (ref. page 43)

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 404–417. (ref. pages 2, 16, and 53)

Bhowmik, A., Gumhold, S., Rother, C., and Brachmann, E. (2020). Reinforced feature points: Optimizing feature detection and description for a high-level task. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 4948–4957. (ref. pages 56, 58, 60, 73, 74, and 75)

Blanco, J.-L., Moreno, F.-A., and Gonzalez-Jimenez, J. (2014). The málaga urban dataset: High-rate stereo and lidars in a realistic urban scenario. *The International Journal of Robotics Research (IJRR)*, 33(2):207–214. (ref. page 26)

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research (IJRR)*, 35(10):1157–1163. (ref. page 59)

Cai, M., Shen, C., and Reid, I. D. (2018). A hybrid probabilistic model for camera relocalization. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12. (ref. page 43)

Carlevaris-Bianco, N., Ushani, A. K., and Eustice, R. M. (2015). University of Michigan North Campus long-term vision and lidar dataset. *The International Journal of Robotics Research (IJRR)*, 35(9):1023–1035. (ref. pages 26 and 27)

Caron, G., Dame, A., and Marchand, E. (2014). Direct model based visual tracking and pose estimation using mutual information. *Image and Vision Computing*, 32(1):54–63. (ref. page 30)

Ceriani, S., Fontana, G., Giusti, A., Marzorati, D., Matteucci, M., Migliore, D., Rizzi, D., Sorrenti, D. G., and Taddei, P. (2009). Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4):353–371. (ref. page 26)

Chen, C., Wang, B., Lu, C. X., Trigoni, N., and Markham, A. (2020). A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence. *arXiv preprint arXiv:2006.12567*. (ref. pages 53, 54, 55, and 72)

Chen, Z., Jacobson, A., Sünderhauf, N., Upcroft, B., Liu, L., Shen, C., Reid, I., and Milford, M. (2017). Deep learning features at scale for visual place recognition. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230. (ref. page 26)

Christiansen, P. H., Kragh, M. F., Brodskiy, Y., and Karstoft, H. (2019). Unsuperpoint: End-to-end unsupervised interest point detector and descriptor. *arXiv preprint arXiv:1907.04011*. (ref. pages 56 and 76)

Churchill, W. and Newman, P. (2013). Experience-based navigation for long-term localisation. *The International Journal of Robotics Research (IJRR)*, 32(14):1645–1661. (ref. pages 2 and 18)

Clark, R., Wang, S., Markham, A., Trigoni, N., and Wen, H. (2017). Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6856–6864. (ref. page 43)

Clement, L. and Kelly, J. (2018). How to train a cat: Learning canonical appearance transformations for direct visual localization under illumination change. *IEEE Robotics and Automation Letters (RAL)*, 3(3):2447–2454. (ref. pages 30 and 42)

Clement, L., Kelly, J., and Barfoot, T. D. (2017). Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery. *Journal of Field Robotics*, 34(1):74–97. (ref. pages 4, 24, 44, 46, and 135)

Comport, A., Malis, E., and Rives, P. (2010). Real-time quadrifocal visual odometry. *The International Journal of Robotics Research (IJRR)*, 29(2-3):245–266. (ref. pages 28 and 29)

Corke, P., Paul, R., Churchill, W., and Newman, P. (2013). Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 2085–2092. (ref. pages 2 and 18)

Costante, G., Mancini, M., Valigi, P., and Ciarfuglia, T. A. (2015). Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation. *IEEE Robotics and Automation Letters (RAL)*, 1(1):18–25. (ref. page 43)

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research (IJRR)*, 27(6):647–665. (ref. page 59)

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839. (ref. page 59)

Dame, A. and Marchand, E. (2013). Using mutual information for appearance-based visual path following. *Robotics and Autonomous Systems*, 61(3):259 – 270. (ref. page 30)

de Ruiter, A. H. and Forbes, J. R. (2013). On the solution ofwahba's problem on so (n). *The Journal of the Astronautical Sciences*, 60(1):1–31. (ref. page 15)

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2016). Deep image homography estimation. *arXiv preprint arXiv:1606.03798*. (ref. page 43)

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018a). Self-improving visual odometry. *arXiv preprint arXiv:1812.03245*. (ref. pages 54, 56, and 59)

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018b). Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 224–236. (ref. pages 55, 56, and 84)

Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., and Sattler, T. (2019). D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8092–8101. (ref. pages 56, 73, 84, and 136)

Dymczyk, M., Stumm, E., Nieto, J., Siegwart, R., and Gilitschenski, I. (2016). Will it last? learning stable features for long-term visual localization. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 572–581. (ref. page 42)

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., and Dehmer, M. (2020). An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence*, 3. (ref. page 6)

Engel, J., Koltun, V., and Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625. (ref. pages 28, 29, and 59)

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 834–849. (ref. pages 28, 29, 30, and 35)

Engel, J., Stückler, J., and Cremers, D. (2015). Large-scale direct slam with stereo cameras. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 1935–1942. (ref. page 30)

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. (ref. page 17)

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). Svo: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. (ref. page 29)

Furgale, P. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560. (ref. pages 1, 2, 15, 16, and 17)

Furgale, P. and Tong, C. H. (2010). Speeded up speeded up robust features. `http://asrl.utias.utoronto.ca/code/gpusurf/`. Accessed: 2022-01-15. (ref. page 16)

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. (ref. pages 26 and 59)

Geman, S., McClure, D. E., and Geman, D. (1992). A nonlinear filter for film restoration and other problems in image processing. *CVGIP: Graphical models and image processing*, 54(4):281–289. (ref. page 35)

Germain, H., Bourmaud, G., and Lepetit, V. (2020). S2dnet: Learning accurate correspondences for sparse-to-dense feature matching. *arXiv preprint arXiv:2004.01673*. (ref. pages 56, 73, and 136)

Germain, H., Lepetit, V., and Bourmaud, G. (2021). Neural reprojection error: Merging feature learning and camera pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 414–423. (ref. pages 56, 57, 58, 60, 73, and 74)

Gladkova, M., Wang, R., Zeller, N., and Cremers, D. (2021). Tight integration of feature-based relocalization in monocular direct visual odometry. *arXiv preprint arXiv:2102.01191*. (ref. pages 56, 72, 73, 74, and 75)

Gridseth, M. and Barfoot, T. (2019). Towards direct localization for visual teach and repeat. In *Proceedings of the Conference on Computer and Robot Vision (CRV)*, pages 97–104. (ref. pages 4, 5, 29, 40, and 134)

Gridseth, M. and Barfoot, T. D. (2020). Deepmel: Compiling visual multi-experience localization into a deep neural network. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1674–1681. (ref. pages 4, 5, 42, 50, and 134)

Gridseth, M. and Barfoot, T. D. (2022). Keeping an eye on things: Deep learned features for long-term visual localization. *IEEE Robotics and Automation Letters (RAL)*, 7(2):1016–1023. (ref. pages 4, 5, 72, 132, and 134)

Griffith, S., Chahine, G., and Pradalier, C. (2017). Symphony lake dataset. *The International Journal of Robotics Research (IJRR)*, 36(11):1151–1158. (ref. page 27)

He, K., Lu, Y., and Sclaroff, S. (2018). Local descriptors optimized for average precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 596–605. (ref. page 73)

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916. (ref. page 44)

Hirose, N., Xia, F., Martín-Martín, R., Sadeghian, A., and Savarese, S. (2019). Deep visual mpc-policy learning for navigation. *IEEE Robotics and Automation Letters (RAL)*, 4(4):3184–3191. (ref. page 72)

Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341. (ref. pages 13, 31, 63, 80, and 81)

Holland, P. W. and Welsch, R. E. (1977). Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827. (ref. page 35)

Huang, H., Ye, H., Sun, Y., and Liu, M. (2020). Monocular visual odometry using learned repeatability and description. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 8913–8919. (ref. pages 56 and 59)

Iyer, G., Krishna Murthy, J., Gupta, G., Madhava Krishna, K., and Paull, L. (2018). Geometric consistency for self-supervised end-to-end visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 380–388. (ref. pages 43 and 54)

Jaramillo, C., Taguchi, Y., and Feng, C. (2017). Direct multichannel tracking. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 347–355. (ref. page 29)

Jau, Y.-Y., Zhu, R., Su, H., and Chandraker, M. (2020). Deep keypoint-based camera pose estimation with geometric constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 4950–4957. (ref. pages 54, 56, 59, and 60)

Kasper, M., Nobre, F., Heckman, C., and Keivan, N. (2020). Unsupervised metric relocalization using transform consistency loss. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1736–1745. (ref. pages 56, 57, 58, 60, 72, 73, 74, 75, and 136)

Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4762–4769. (ref. page 43)

Kendall, A. and Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6555–6564. (ref. pages 43 and 45)

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946. (ref. pages 26, 43, and 73)

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. (ref. pages 46, 66, and 78)

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 1097–1105. (ref. pages 43 and 44)

Kumar, A., Gupta, S., Fouhey, D., Levine, S., and Malik, J. (2018). Visual memory for robust path following. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. (ref. page 72)

Laskar, Z., Melekhov, I., Kalia, S., and Kannala, J. (2017). Camera relocalization by computing pairwise relative poses using convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 929–938. (ref. pages 43 and 73)

Li, D., Shi, X., Long, Q., Liu, S., Yang, W., Wang, F., Wei, Q., and Qiao, F. (2020). Dxslam: A robust and efficient visual slam system with deep features. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 4958–4965. (ref. pages 56 and 59)

Li, R., Wang, S., and Gu, D. (2018a). Ongoing evolution of visual slam from geometry to deep learning: Challenges and opportunities. *Cognitive Computation*, 10(6):875–889. (ref. page 42)

Li, R., Wang, S., Long, Z., and Gu, D. (2018b). Undeepvo: Monocular visual odometry through unsupervised deep learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291. (ref. page 54)

Li, Y., Snavely, N., Huttenlocher, D., and Fua, P. (2012). Worldwide pose estimation using 3d point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 15–29. (ref. page 26)

Li, Z. and Snavely, N. (2018). Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2041–2050. (ref. page 26)

Linegar, C. (2016). *Vision-only localisation under extreme appearance change*. PhD thesis, University of Oxford. (ref. page 3)

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. (ref. page 29)

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 674–679. (ref. page 29)

Luo, Z., Zhou, L., Bai, X., Chen, H., Zhang, J., Yao, Y., Li, S., Fang, T., and Quan, L. (2020). Aslfeat: Learning local features of accurate shape and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6589–6598. (ref. pages 56, 73, and 136)

Lv, Z., Dellaert, F., Rehg, J. M., and Geiger, A. (2019). Taking a deeper look at the inverse compositional algorithm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4581–4590. (ref. pages 56, 57, 58, 60, and 75)

Ma, J., Jiang, X., Fan, A., Jiang, J., and Yan, J. (2021). Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, 129(1):23–79. (ref. page 55)

MacTavish, K., Paton, M., and Barfoot, T. D. (2017). Visual triage: A bag-of-words experience selector for long-term visual route following. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2065–2072. (ref. pages 2, 19, and 20)

MacTavish, K., Paton, M., and Barfoot, T. D. (2018). Selective memory: Recalling relevant experience for long-term visual localization. *Journal of Field Robotics*, 35(8):1265–1292. (ref. pages 18 and 27)

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15. (ref. pages 3, 26, 27, 42, and 74)

Mahjourian, R., Wicke, M., and Angelova, A. (2018). Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5667–5675. (ref. page 54)

Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186. (ref. page 66)

Martull, S., Peris, M., and Fukui, K. (2012). Realistic cg stereo image dataset with ground truth disparity maps. In *ICPR workshop TrakMark2012*, volume 111, pages 117–118. (ref. page 59)

Masatoshi, A., Yuuto, C., Kanji, T., and Kentaro, Y. (2015). Leveraging image-based prior in cross-season place recognition. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5455–5461. (ref. page 26)

Melekhov, I., Ylioinas, J., Kannala, J., and Rahtu, E. (2017). Relative camera pose estimation using convolutional neural networks. In *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 675–687. (ref. pages 41, 43, and 44)

Melekhov, I., Ylioinas, J., Kannala, J., and Rahtu, E. (2018). Image-Based Localization Using Hourglass Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 870–877. (ref. page 43)

Milford, M. J. and Wyeth, G. F. (2012). Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1643–1649. (ref. page 26)

Mohanty, V., Agrawal, S., Datta, S., Ghosh, A., Sharma, V. D., and Chakravarty, D. (2016). Deepvo: A deep learning approach for monocular visual odometry. *arXiv preprint arXiv:1611.06069*. (ref. page 43)

Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163. (ref. page 59)

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*, page 807–814. (ref. page 6)

Naseer, T. and Burgard, W. (2017). Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 1525–1530. (ref. page 43)

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). Dtam: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327. (ref. pages 28, 29, 38, and 39)

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA. (ref. page 6)

Oliveira, G. L., Radwan, N., Burgard, W., and Brox, T. (2020). Topometric localization with deep learning. In *Robotics Research*, pages 505–520. (ref. page 43)

Ono, Y., Trulls, E., Fua, P., and Yi, K. M. (2018). Lf-net: learning local features from images. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 6234–6244. (ref. pages 55 and 56)

OpenCV (2019). StereoBinarySGBM. `https://docs.opencv.org/3.4.8/d2/d85/classcv_1_1StereoSGBM.html`. Version: 3.4.8. (ref. page 13)

Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E. A., and Boots, B. (2020). Imitation learning for agile autonomous driving. *The International Journal of Robotics Research (IJRR)*, 39(2-3):286–302. (ref. page 136)

Pandey, G., McBride, J. R., and Eustice, R. M. (2011). Ford campus vision and lidar data set. *The International Journal of Robotics Research (IJRR)*, 30(13):1543–1552. (ref. page 26)

Park, S., Schöps, T., and Pollefeys, M. (2017). Illumination change robustness in direct visual slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4523–4530. (ref. page 30)

Pascoe, G., Maddern, W., Stewart, A. D., and Newman, P. (2015). Farlap: Fast robust localisation using appearance priors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6366–6373. (ref. page 30)

Pascoe, G., Maddern, W., Tanner, M., Piniés, P., and Newman, P. (2017). Nid-slam: Robust monocular slam using normalised information distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1446–1455. (ref. page 30)

Patel, M., Emery, B., and Chen, Y. Y. (2018). ContextualNet: Exploiting contextual information using LSTMs to improve image-based localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5890–5896. (ref. page 43)

Paton, M. (2018). *Expanding the Limits of Vision-Based Autonomous Path Following*. PhD thesis, University of Toronto. (ref. pages 2, 6, 17, 19, and 21)

Paton, M., MacTavish, K., Berczi, L.-P., van Es, S. K., and Barfoot, T. D. (2018). I can see for miles and miles: An extended field test of visual teach and repeat 2.0. In *Proceedings of Field and Service Robotics (FSR)*, pages 415–431. (ref. page 15)

Paton, M., MacTavish, K., Ostafew, C. J., and Barfoot, T. D. (2015). It's not easy seeing green: Lighting-resistant stereo visual teach and repeat using color-constant images. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1519–1526. (ref. pages 2, 18, 36, 41, and 46)

Paton, M., MacTavish, K., Warren, M., and Barfoot, T. D. (2016). Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 1918–1925. (ref. pages 2, 18, 36, 41, 46, and 64)

Peretroukhin, V. and Kelly, J. (2018). Dpc-net: Deep pose correction for visual localization. *IEEE Robotics and Automation Letters (RAL)*, 3(3):2424–2431. (ref. pages 43 and 54)

Peretroukhin, V., Wagstaff, B., and Kelly, J. (2019). Deep probabilistic regression of elements of so (3) using quaternion averaging and uncertainty injection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 83–86. (ref. page 43)

Piasco, N., Sidibé, D., Gouet-Brunet, V., and Demonceaux, C. (2019). Learning scene geometry for visual localization in challenging conditions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9094–9100. (ref. page 72)

Radwan, N., Valada, A., and Burgard, W. (2018). Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters (RAL)*, 3(4):4407–4414. (ref. page 43)

Raguram, R., Chum, O., Pollefeys, M., Matas, J., and Frahm, J.-M. (2013). Usac: A universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2022–2038. (ref. page 17)

Rawlings, J. and Mayne, D. (2009). *Model Predictive Control: Theory and Design*. (ref. page 18)

Revaud, J., Weinzaepfel, P., De Souza, C., Pion, N., Csurka, G., Cabon, Y., and Humenberger, M. (2019). R2d2: Repeatable and reliable detector and descriptor. *arXiv preprint arXiv:1906.06195*. (ref. pages 56, 73, 84, and 136)

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. (ref. page 8)

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571. (ref. page 80)

Saha, S., Varma, G., and Jawahar, C. V. (2018). Improved visual relocalization by discovering anchor points. In *Proceedings of the British Machine Vision Conference (BMVC)*, page 164. (ref. page 43)

Sarlin, P.-E., Cadena, C., Siegwart, R., and Dymczyk, M. (2019). From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12716–12725. (ref. pages 56, 59, 73, and 74)

Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4938–4947. (ref. page 56)

Sarlin, P.-E., Unagar, A., Larsson, M., Germain, H., Toft, C., Larsson, V., Pollefeys, M., Lepetit, V., Hammarstrand, L., Kahl, F., et al. (2021). Back to the feature: Learning robust camera localization from pixels to pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3247–3257. (ref. pages 56, 57, 58, 60, 72, 73, 74, and 75)

Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., et al. (2018). Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8601–8610. (ref. pages 26, 27, 42, 58, and 74)

Sattler, T., Weyand, T., Leibe, B., and Kobbelt, L. (2012). Image retrieval for image-based localization revisited. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12. (ref. pages 3 and 26)

Sattler, T., Zhou, Q., Pollefeys, M., and Leal-Taixe, L. (2019). Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3302–3312. (ref. pages 53, 54, and 73)

Smith, M., Baldwin, I., Churchill, W., Paul, R., and Newman, P. (2009). The new college vision and laser data set. *The International Journal of Robotics Research (IJRR)*, 28(5):595–599. (ref. pages 26 and 27)

Spencer, J., Bowden, R., and Hadfield, S. (2019). Scale-adaptive neural dense features: Learning via hierarchical context aggregation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6200–6209. (ref. pages 56 and 57)

Spencer, J., Bowden, R., and Hadfield, S. (2020). Same features, different day: Weakly supervised feature learning for seasonal invariance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6459–6468. (ref. pages 56, 57, 58, 72, 73, and 74)

Stewart, A. D. and Newman, P. (2012). Laps - localisation using appearance of prior structure: 6-dof monocular camera localisation using prior pointclouds. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2625–2632. (ref. page 30)

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 573–580. (ref. page 69)

Sun, L., Taher, M., Wild, C., Zhao, C., Majer, F., Yan, Z., Krajnik, T., Prescott, T., and Duckett, T. (2021). Robust and long-term monocular teach-and-repeat navigation using a single-experience map. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 2635–2642. (ref. pages 56, 72, 73, 74, 75, and 131)

Sünderhauf, N., Neubert, P., and Protzel, P. (2013). Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshops*, page 2013. (ref. page 26)

Sünderhauf, N., Shirazi, S., Jacobson, A., Dayoub, F., Pepperell, E., Upcroft, B., and Milford, M. (2015). Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 1–10. (ref. page 42)

Swedish, T. and Raskar, R. (2018). Deep visual teach and repeat on path networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1614–161409. (ref. page 72)

Tang, C. and Tan, P. (2019). Ba-net: Dense bundle adjustment networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. (ref. pages 56, 57, and 136)

Tang, J., Ambrus, R., Guizilini, V., Pillai, S., Kim, H., Jensfelt, P., and Gaidon, A. (2020). Self-Supervised 3D Keypoint Learning for Ego-Motion Estimation. In *Proceedings of the Conference on Robot Learning (CoRL)*. (ref. pages 54, 56, 59, 60, 136, and 137)

Thomas, H., Agro, B., Gridseth, M., Zhang, J., and Barfoot, T. D. (2021). Self-supervised learning of lidar segmentation for autonomous indoor navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 14047–14053. (ref. page 137)

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:376–380. (ref. pages 13 and 15)

Valada, A., Radwan, N., and Burgard, W. (2018). Deep Auxiliary Learning for Visual Localization and Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6939–6946. (ref. pages 43 and 73)

Venator, M., Himer, Y. E., Aklanoglu, S., Bruns, E., and Maier, A. (2021). Self-supervised learning of domain-invariant local features for robust visual localization under challenging conditions. *IEEE Robotics and Automation Letters (RAL)*, 6(2):2753–2760. (ref. pages 56, 57, 58, 73, and 74)

Verdie, Y., Yi, K., Fua, P., and Lepetit, V. (2015). Tilde: A temporally invariant learned detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5279–5288. (ref. page 56)

Viola, P. and Wells III, W. M. (1997). Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154. (ref. page 30)

von Stumberg, L., Wenzel, P., Khan, Q., and Cremers, D. (2020a). Gn-net: The gauss-newton loss for multi-weather relocalization. *IEEE Robotics and Automation Letters (RAL)*, 5(2):890–897. (ref. pages 56, 57, 72, 73, and 136)

von Stumberg, L., Wenzel, P., Yang, N., and Cremers, D. (2020b). Lm-reloc: Levenberg-marquardt based direct visual relocalization. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 968–977. (ref. pages 56, 57, 72, 73, and 136)

Wagstaff, B., Peretroukhin, V., and Kelly, J. (2020). Self-supervised deep pose corrections for robust visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2331–2337. (ref. page 54)

Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2017). Image-based localization using lstms for structured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 627–637. (ref. page 43)

Wang, Q., Zhou, X., Hariharan, B., and Snavely, N. (2020). Learning feature descriptors using camera pose supervision. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 757–774. (ref. pages 56 and 57)

Wang, S., Clark, R., Wen, H., and Trigoni, N. (2017). Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. (ref. page 26)

Wang, S., Clark, R., Wen, H., and Trigoni, N. (2018). End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research (IJRR)*, 37(4-5):513–542. (ref. pages 43 and 54)

Wang, X., Maturana, D., Yang, S., Wang, W., Chen, Q., and Scherer, S. (2019). Improving learning-based ego-motion estimation with homomorphism-based losses and drift correction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 970–976. (ref. page 54)

Wenzel, P., Wang, R., Yang, N., Cheng, Q., Khan, Q., von Stumberg, L., Zeller, N., and Cremers, D. (2020). 4Seasons: A cross-season dataset for multi-weather SLAM in autonomous driving. In *Proceedings of the German Conference on Pattern Recognition (GCPR)*, pages 404–417. (ref. pages 26, 27, 42, and 74)

Wolcott, R. W. and Eustice, R. M. (2014). Visual localization within lidar maps for automated urban driving. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183. (ref. page 30)

Wu, J., Ma, L., and Hu, X. (2017). Delving deeper into convolutional neural networks for camera relocalization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5644–5651. (ref. page 43)

Xu, B., Davison, A. J., and Leutenegger, S. (2020). Deep probabilistic feature-metric tracking. *IEEE Robotics and Automation Letters (RAL)*, 6(1):223–230. (ref. pages 56, 57, 58, 60, and 75)

Yang, N., Stumberg, L. v., Wang, R., and Cremers, D. (2020). D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1281–1292. (ref. page 53)

Yi, K. M., Trulls, E., Lepetit, V., and Fua, P. (2016). Lift: Learned invariant feature transform. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 467–483. (ref. pages 55 and 56)

Yoon, D. J., Zhang, H., Gridseth, M., Thomas, H., and Barfoot, T. D. (2021). Unsupervised learning of lidar features for use ina probabilistic trajectory estimator. *IEEE Robotics and Automation Letters (RAL)*, 6(2):2130–2138. (ref. page 136)

Yu, H., Ye, W., Feng, Y., Bao, H., and Zhang, G. (2020). Learning bipartite graph matching for robust visual localization. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 146–155. (ref. page 56)

Zhan, H., Garg, R., Weerasekera, C. S., Li, K., Agarwal, H., and Reid, I. M. (2018). Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 340–349. (ref. page 43)

Zhao, C., Sun, L., Krajník, T., Duckett, T., and Yan, Z. (2021). Monocular teach-and-repeat navigation using a deep steering network with scale estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, pages 2613–2619. (ref. pages 73, 74, and 75)