# Deep Learning for Long-Term Metric Visual Localization

by

Yuxuan Chen

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

University of Toronto Institute for Aerospace Studies
University of Toronto

Deep Learning for Long-Term Metric Visual Localization

Yuxuan Chen
Master of Applied Science

University of Toronto Institute for Aerospace Studies
University of Toronto
2023

# Abstract

Visual localization is the task of estimating camera pose in a known scene, which is an essential problem in robotics and computer vision. However, long-term visual localization is still a challenge due to the environmental appearance changes caused by lighting and seasons. While experience-based localization methods bridge the appearance gap by relying on intermediate experiences, they require collecting experiences continuously to capture the incremental appearance changes, and cannot directly generalize to a new path.

In this thesis, we tackle long-term localization using deep learning methods without relying on intermediate experiences. We first show that we can learn appearance-invariant sparse visual features using self-supervised learning that can be used in closed-loop path following across a full range of lighting change while preserving generalizability. We further explore deep image-to-image translation as a mean of improving long-term metric localization. We show that by transforming input images captured from different lighting conditions into a common target domain prior to feature matching substantially improves upon localization accuracy.

# Acknowledgements

First of all, I would like to extend my deepest gratitude to my advisor, Professor Tim Barfoot for all his ideas, support, and guidance. Under his supervision, I have gained new understanding in the field of robotics.

I thank all my colleagues in the Autonomous Space Robotics Laboratory, whose contributions and collaboration have been essential to the completion of this research. Special thanks to Mona Gridseth, Yuchen Wu, Binbin Xu, Frederike Dümbgen, Jordy Sehn, and Alec Krawciw. It has been a pleasure to work with all of you.

Finally, I would like thank my parents and my partner Stanley for never ceasing to believe in me. And thank you to all of my friends for their continued support.

# Contents

**1 Introduction** **1**

1.1 Motivation . . . . . . . . . . 1

1.2 Contribution . . . . . . . . . . 2

1.3 Thesis Overview . . . . . . . . . . 3

**2 Background** **5**

2.1 Camera Model . . . . . . . . . . 5

2.1.1 Monocular Camera Model . . . . . . . . . . 5

2.1.2 Stereo Camera Model . . . . . . . . . . 6

2.2 Pose Estimation . . . . . . . . . . 7

2.3 Visual Teach & Repeat . . . . . . . . . . 11

2.3.1 Single-Experience Localization . . . . . . . . . . 12

2.3.2 Multi-Experience Localization . . . . . . . . . . 14

2.4 UTIAS Datasets . . . . . . . . . . 15

2.4.1 UTIAS In-the-Dark . . . . . . . . . . 16

2.4.2 UTIAS Multiseason . . . . . . . . . . 17

**3 Self-Supervised Feature Learning for Long-Term Localization** **19**

3.1 Introduction . . . . . . . . . . 19

3.2 Related Work . . . . . . . . . . 20

3.3 Methodology . . . . . . . . . . 23

3.3.1 Place Recognition . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
3.3.2 Feature Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
3.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
3.4.1 Datasets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
3.4.2 Training Data Generation using Topological Localization . . . 36
3.4.3 Training and Inference . . . . . . . . . . . . . . . . . . . . . . . 39
3.4.4 Offline Localization: Learned Features Comparison . . . . . . 40
3.4.5 Offline Localization: Ablation Study . . . . . . . . . . . . . . 41
3.4.6 Online VT&R: Closed-Loop Experiments . . . . . . . . . . . . 42
3.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

**4 Image Transformation using Neural Style Transfer 48**
4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
4.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
4.3 Methodology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
4.3.1 Image Transformation . . . . . . . . . . . . . . . . . . . . . . . 51
4.3.2 Feature Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
4.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
4.4.1 Datasets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.4.2 Training and Inference . . . . . . . . . . . . . . . . . . . . . . . 58
4.4.3 Comparison Between TransNet and FeatNet . . . . . . . . . . 59
4.4.4 Comparison Between FeatNet and SURF . . . . . . . . . . . . 60
4.4.5 Comparison Between TransNet and Colourspace Transformation 61
4.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

**5 Conclusion 64**
5.1 Summary and Contributions . . . . . . . . . . . . . . . . . . . . . . 64
5.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65

# List of Tables

3.1 Comparison of the path-following errors and the median number of inliers 39

4.1 Comparison of the path-following errors and average number of inliers for UTIAS In-the-Dark . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

4.2 Comparison of the path-following errors and average number of inliers for Oxford RobotCar . . . . . . . . . . . . . . . . . . . . . . . . . . . 63

# List of Figures

1.1 Real-time deployment of the Grizzly ground robot under different lighting conditions . . . 3

2.1 Stereo Camera Model Configuration . . . 7
2.2 VT&R overview . . . 12
2.3 Single-experience localization . . . 15
2.4 Multi-experience localization . . . 16
2.5 The Clearpath Grizzly Robot . . . 17
2.6 Training paths of the two UTIAS datasets . . . 18

3.1 Overall pipeline for self-supervised feature learning . . . 21
3.2 Direct image correspondences . . . 24
3.3 Experience-Association Graph . . . 26
3.4 Sample image correspondences generated by seqSLAM . . . 29
3.5 Feature Network Architecture . . . 31
3.6 Keypoint matching . . . 32
3.7 Feature Pipeline . . . 34
3.8 Frame-level topological matching results . . . 35
3.9 Distribution of distances between sampled image pairs . . . 36
3.10 Training and testing paths . . . 37
3.11 Median feature inlier count v.s. lateral and yaw angle errors . . . 37

3.12 Selected camera views . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

3.13 The mean number of inliers for each repeat of the offline experiments 44

3.14 The mean number of inliers for each repeat of the closed loop experiments 45

3.15 The mean number of inliers for each repeat of the closed loop experiments 46

3.16 VT&R live demo . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

4.1 High-level Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

4.2 Detailed Overall Pipeline . . . . . . . . . . . . . . . . . . . . . . . . 50

4.3 Content and Style Loss Overview. . . . . . . . . . . . . . . . . . . . . 52

4.4 Qualitative Results. . . . . . . . . . . . . . . . . . . . . . . . . . . 57

# List of Algorithms

1 Direct Image Correspondences . . . . . . . . . . . . . . . . . . . . . . . . 28
2 Indirect Image Correspondences . . . . . . . . . . . . . . . . . . . . . . . . 29
3 Differentiable Point Matching [1] . . . . . . . . . . . . . . . . . . . . . . . . 34

# Acronyms

| | |
|---|---|
| **CNN** | Convolutional Neural network. |
| **DNN** | Deep Neural Network. |
| **MEL** | Multi-Experience Localization. |
| **RANSAC** | Random Sample Consensus. |
| **ReLU** | Rectified Linear Unit. |
| **RMSE** | Root Mean Squared Error. |
| **SLAM** | Simultaneous Localization and Mapping. |
| **SURF** | Speeded-Up Robust Features. |
| **SVD** | Singular Value Decomposition. |
| **UTIAS** | University of Toronto Institute for Aerospace Studies. |
| **VO** | Visual Odometry. |
| **VT&R** | Visual Teach and Repeat. |

# Notation

| | |
|---|---|
| $a$ | A scalar quantity |
| $\mathbf{a}$ | A column vector |
| $\mathbf{A}$ | A matrix |
| $\mathbf{1}$ | The identity matrix |
| $\mathbf{0}$ | The zero matrix |
| $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$ | A vectrix representing a reference frame in three dimensions |
| SO(3) | The special orthogonal group |
| SE(3) | The special Euclidean group |
| $\boldsymbol{C}_{ba}$ | A $3 \times 3$ rotation matrix (member of SO(3)) |
| $\boldsymbol{T}_{ba}$ | A $4 \times 4$ transformation matrix (member of SE(3)) |

# Chapter 1

# Introduction

## 1.1 Motivation

Long-term visual localization is an essential problem in robotics and computer vision, but remains challenging due to the drastic environmental appearance changes caused by lighting and seasons as shown in Figure 1.1.

Visual Teach and Repeat (VT&R) [2] achieves outdoor autonomous navigation by building a locally consistent visual map using inexpensive visual sensors (i.e., stereo cameras). Similar to other traditional point-based localization approaches, VT&R finds correspondences between local features extracted from images by applying hand-crafted descriptors (e.g., SIFT, SURF, ORB [3–5]), then recovers the full 6-DoF camera pose. However, such hand-crafted features are not robust under extreme appearance changes.

To address this, Multi-Experience Localization (MEL) [2, 6] uses intermediate experiences to handle gradual appearance change of the environment. During a more challenging repeat, it retrieves the most relevant experiences to bridge the appearance gap and localize to the initial taught path. However, it is difficult for MEL to deal with rapid scene changes as it would require a large number of intermediate experiences to capture the incremental appearance change. In addition, the collected intermediate experiences are specific to each map, and can not be generalized to new areas, which makes the deployment quite restrictive in real life.

In Chapter 3, we aim to tackle the challenge of long-term localization in outdoor environments using deep learning techniques, removing the need for intermediate bridging experiences. In Chapter 3, we first show that we can train a neural network to learn visual features in a self-supervised setting, which can be integrated with a classical pose estimator in the VT&R pipeline. We demonstrated the usefulness of the learned features during real-time closed-loop path following, and showed that the learned features can generalize well to new areas across a full range of lighting change.

In Chapter 4 of this thesis, we improve on this work by learning a nonlinear image transformation using neural style transfer, which transforms input images to a target domain prior to feature matching. We adopt a combination of an image transformation network and a feature-learning network to improve long-term localization performance. Given night-to-day image pairs, the image transformation network transforms the night images into day-like conditions prior to feature matching; the feature network learns to detect keypoint locations with their associated descriptor values, which can be passed to a classical pose estimator to compute the relative poses. In this chapter, we examine the effectiveness of combining style transfer and feature learning, and show that such a combination substantially improves the localization accuracy on long-term vision datasets.

## 1.2 Contribution

The novel contributions of this thesis are organized as follows:

- In Chapter 3, we propose a novel self-supervised feature learning framework that improves the robustness of visual localization for VT&R, which achieves

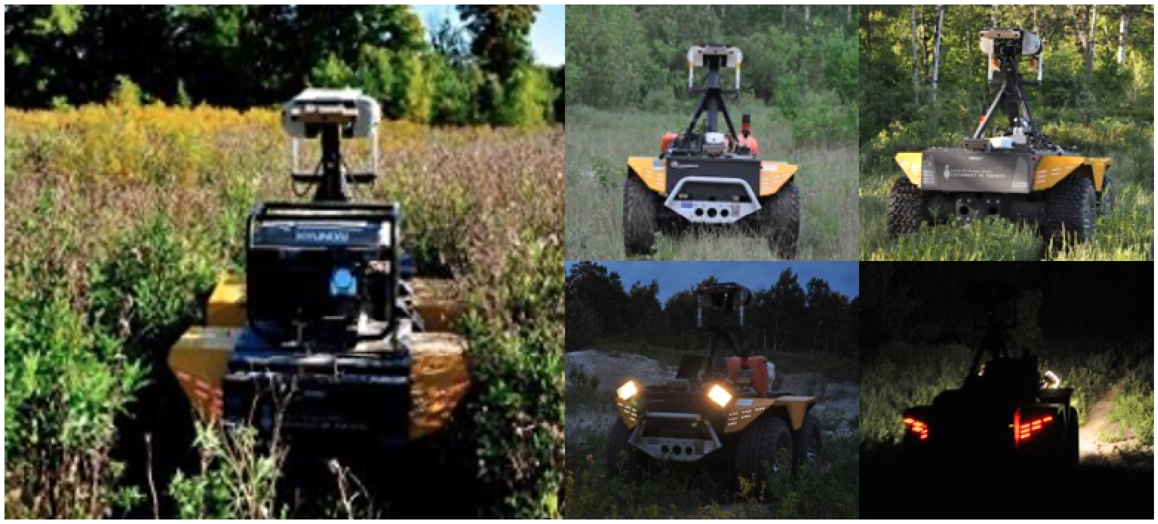


Figure 1.1: **Grizzly ground robot** autonomously repeats a taught path with learned features despite severe lighting changes.

long-term localization while preserving generalizability [7].

- In Chapter 4, we propose an end-to-end differentiable pipeline that incorporates an image transformation network and a feature learning network to improve the localization performance [8].

## 1.3 Thesis Overview

This thesis is structured as follows; Chapter 2 provides a background discussion of concepts in sensor models and 3D state estimation that we rely on in the following chapters of the thesis. In addition, we provide a brief overview of the VT&R pipeline, as well as the datasets we rely on that were collected using Multi-experience VT&R in 2017.

In Chapter 3, we train a deep neural network to learn sparse visual features in a self-supervised manner. The learned features are fed into the classical pose estimator to estimate the relative pose between two input images. We apply the learned features in the VT&R pipeline to perform closed-loop long-term metric localization across a full range of lighting change.

In Chapter 4, we learn a deep image transformation that performs domain adaptation to the input images. Given night-to-day image pairs, the image transformation network transforms the night images into day-like conditions that are explicitly optimized for keypoint matching. In this chapter, we further investigate if it is better to learn features, image transformations, or both.

Finally, Chapter 5 summarizes the conclusions and novel contributions of the thesis, and discuss some work that could be done in the future for further improvements.

# Chapter 2

# Background

This section presents some common topics related to different chapters of the thesis. We start by discussing selected topics of the sensor model and three-dimensional geometry that are relevant to our work. In addition, we give a high-level overview of VT&R pipeline and details on the datasets we use to train networks.

## 2.1 Camera Model

We present the camera sensor model as defined in Chapter 6 of *State Estimation for Robotics* [9].

### 2.1.1 Monocular Camera Model

We start with discussing the basic frontal projection model for a monocular perspective camera. Given the coordinates of a 3D point, $\boldsymbol{p} = [x\ y\ z]^T$, in the sensor frame, we can project it into its corresponding image frame coordinates, $\boldsymbol{q} = [u\ v]^T$. The perspective camera model, $\boldsymbol{s} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, is defined as follows:

$$\boldsymbol{q} = \begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{s}(\boldsymbol{p}) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{P}} \underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{K}} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.1}$$

where $\boldsymbol{P}$ is the projection matrix that removes the bottom row from the homogenous point representation, and $\boldsymbol{K}$ is the intrinsic parameter matrix of the camera containing the focal length, expressed in units of horizontal pixels, $f_u$, and vertical pixels, $f_v$, and the coordinates of the optical centre $(c_u, c_v)$.

However, the forward measurement model for a monocular camera is not invertible due to the missing depth information $z$.

### 2.1.2 Stereo Camera Model

In order to recover depth information $z$ of the point in the 3D space, we define a stereo camera model that combines observations from two perspective cameras, where the two cameras are rigidly connected by a known and fixed transform. 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, $\underrightarrow{\mathcal{F}}_l$.

We can define the forward measurement model for a stereo camera, $\boldsymbol{g} : \mathbb{R}^3 \to \mathbb{R}^4$, that maps a point, $\boldsymbol{P}$, in the sensor frame, $\underrightarrow{\mathcal{F}}_s$, with coordinates, $\boldsymbol{P} = [x \ y \ z]^T$, to stereo image coordinates, $\boldsymbol{P} = [u_l \ v_l \ u_r \ v_r]^T$:

$$\boldsymbol{q} = \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix} = \boldsymbol{g}(\boldsymbol{p}) = \underbrace{\begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u b \\ 0 & f_v & c_v & 0 \end{bmatrix}}_{\boldsymbol{M}} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \boldsymbol{M} \frac{1}{\boldsymbol{c}^T \boldsymbol{p}} \boldsymbol{p}, \tag{2.2}$$

where $\boldsymbol{c}^T = [0 \ 0 \ 0 \ 1]$. We assume that both cameras have the same intrinsic proper-

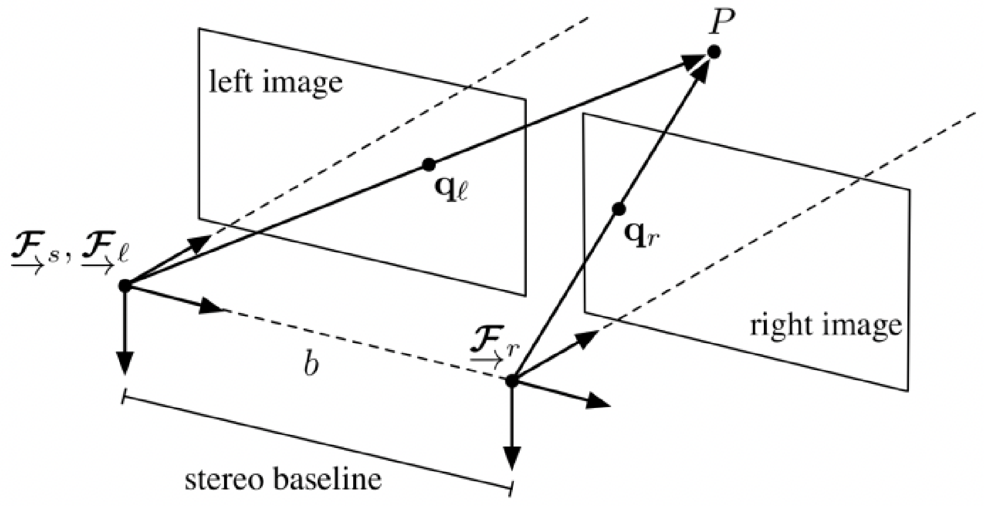

P
left image
$\mathbf{q}_\ell$
$\underrightarrow{\mathcal{F}}_s, \underrightarrow{\mathcal{F}}_\ell$
$\mathbf{q}_r$
right image
$\underrightarrow{\mathcal{F}}_r$
$b$
stereo baseline

Figure 2.1: **Stereo Camera Model Configuration**. The cameras are separated by the baseline, $b$, along the x-axis. This figure is taken from [9].

ties.

In order to recover the coordinates of the point, $\boldsymbol{P}$, we invert the stereo camera model, which maps the stereo image coordinates to a homogeneous point:

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \boldsymbol{g}^{-1}(\mathbf{q}) = \begin{bmatrix} \frac{b}{d}(u_l - c_u) \\ \frac{bf_u}{df_v}(v_l - c_v) \\ \frac{b}{d}f_u \\ 1 \end{bmatrix}. \tag{2.3}$$

We perform stereo matching to the disparity, $d = u_l - u_r$.

## 2.2 Pose Estimation

The following material is based on Chapter 8 of *State Estimation for Robotics* [9]. Rotations in three dimensions can be represented as elements of the special orthogonal group, which is defined as the set of $3 \times 3$ rotation matrices as follows:

$$SO(3) = \{\boldsymbol{C} \in \mathbb{R}^{3\times 3} | \boldsymbol{C}\boldsymbol{C}^T = \boldsymbol{1}, \det \boldsymbol{C} = 1\}. \tag{2.4}$$

Poses are represented using the special Euclidean group by combining a three-dimensional rotation $\boldsymbol{C}$ with a three-dimensional translation $\boldsymbol{r}$, which is defined as the set of $4 \times 4$ transformation matrices as follows:

$$SE(3) = \left\{ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{r} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \boldsymbol{C} \in SO(3), \boldsymbol{r} \in \mathbb{R}^3 \right\}. \tag{2.5}$$

Given two sets of 3D point measurements, we need to solve a point-cloud alignment problem to find the relative transform in Chapter 3 and 4 of this thesis.

Assume that we have two frames, one attached to a map frame $\underrightarrow{\mathcal{F}}_m$, and one to the live robot camera, $\underrightarrow{\mathcal{F}}_l$. For both of these frames, we have $M$ measurements of a set of landmark points, $\boldsymbol{P}_i$, given in the respective frames, namely $\boldsymbol{r}_m^{p_i m}$ and $\boldsymbol{r}_l^{p_i l}$, where $i = 1 \cdots M$. The goal is to find the rotation matrix, $\boldsymbol{C}_{lm}$, and translation, $\boldsymbol{r}_m^{lm}$, that will align the two sets of points. We define:

$$\boldsymbol{y}_i = \boldsymbol{r}_m^{p_i m}, \quad \boldsymbol{p}_i = \boldsymbol{r}_l^{p_i l}, \quad \boldsymbol{r} = \boldsymbol{r}_m^{lm}, \quad \boldsymbol{C} = \boldsymbol{C}_{lm}, \tag{2.6}$$

$$\boldsymbol{y} = \frac{1}{w}\sum_{i=1}^{M} w_i \boldsymbol{y}_i, \quad \boldsymbol{p} = \frac{1}{w}\sum_{i=1}^{M} w_i \boldsymbol{p}_i, \quad w = \sum_{i=1}^{M} w_i, \tag{2.7}$$

where $w_i$ are scalar weights for each point. We define the error term for each point as:

$$\boldsymbol{e}_i = \boldsymbol{y}_i - \boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{r}), \tag{2.8}$$

We minimize the following cost function:

$$J(\boldsymbol{C},\boldsymbol{r}) = \frac{1}{2}\sum_{i=1}^{M} w_i \boldsymbol{e}_i^T \boldsymbol{e}_i = \frac{1}{2}\sum_{i=1}^{M} w_i(\boldsymbol{y}_i - \boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{r}))^T(\boldsymbol{y}_i - \boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{r})), \tag{2.9}$$

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

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

$$\boldsymbol{d} = \boldsymbol{r} + \boldsymbol{C}^T\boldsymbol{y} - \boldsymbol{p}, \tag{2.10}$$

such that we can rewrite the cost function as

$$J(\boldsymbol{C},\boldsymbol{d}) = \frac{1}{2}\sum_{i=1}^{M} w_i((\boldsymbol{y}_i - \boldsymbol{y}) - \boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{p}))^T((\boldsymbol{y}_i - \boldsymbol{y}) - \boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{p})) + \frac{1}{2}\boldsymbol{d}^T\boldsymbol{d}, \tag{2.11}$$

where the first term of the cost only depends on $\boldsymbol{C}$, and the second cost term only depends on $\boldsymbol{d}$. Thus, the second term can be minimized by setting $\boldsymbol{d} = \boldsymbol{0}$ as it only depends on $d$:

$$\boldsymbol{r} = \boldsymbol{p} - \boldsymbol{C}^T\boldsymbol{y} \tag{2.12}$$

We minimize the first term of the cost with respect to $\boldsymbol{C}$ in order to find the rotation by multiplying out parts of the cost term,

$$\begin{aligned} ((\boldsymbol{y}_i - \boldsymbol{y}) - \boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{p}))^T((\boldsymbol{y}_i - \boldsymbol{y}) - \boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{p})) = \\ (\boldsymbol{y}_i - \boldsymbol{y})^T(\boldsymbol{y}_i - \boldsymbol{y}) - 2((\boldsymbol{y}_i - \boldsymbol{y})^T\boldsymbol{C}(\boldsymbol{p}_i - \boldsymbol{p})) + (\boldsymbol{p}_i - \boldsymbol{p})^T(\boldsymbol{p}_i - \boldsymbol{p}), \end{aligned} \tag{2.13}$$

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

$$\begin{aligned}\frac{1}{w}\sum_{i=1}^{M} w_i\left((\boldsymbol{y}_i-\boldsymbol{y})^T\boldsymbol{C}(\boldsymbol{p}_i-\boldsymbol{p})\right) &= \frac{1}{w}\sum_{i=1}^{M} w_i\mathrm{tr}\left(\boldsymbol{C}(\boldsymbol{p}_i-\boldsymbol{p})(\boldsymbol{y}_i-\boldsymbol{y})^T\right)\\ &= \mathrm{tr}\left(\boldsymbol{C}\frac{1}{w}\sum_{i=1}^{M} w_i(\boldsymbol{p}_i-\boldsymbol{p})(\boldsymbol{y}_i-\boldsymbol{y})^T\right)\\ &= \mathrm{tr}(\boldsymbol{C}\boldsymbol{W}^T),\end{aligned} \tag{2.14}$$

where

$$\boldsymbol{W} = \frac{1}{w}\sum_{i=1}^{M} w_i(\boldsymbol{y}_i-\boldsymbol{y})(\boldsymbol{p}_i-\boldsymbol{p})^T. \tag{2.15}$$

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

$$J(\boldsymbol{C},\boldsymbol{\Lambda},\gamma) = -\mathrm{tr}(\boldsymbol{C}\boldsymbol{W}^T) + \mathrm{tr}(\boldsymbol{\Lambda}(\boldsymbol{C}\boldsymbol{C}^T-\boldsymbol{1})) + \gamma(\det\,\boldsymbol{C}-1), \tag{2.16}$$

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

$$\frac{\partial J}{\partial \boldsymbol{C}} = -\boldsymbol{W} + 2\boldsymbol{\Lambda}\boldsymbol{C} + \gamma\det\,\boldsymbol{C}\boldsymbol{C}^{-T} = -\boldsymbol{W} + \boldsymbol{L}\boldsymbol{C}, \tag{2.17}$$

$$\frac{\partial J}{\partial \boldsymbol{\Lambda}} = \boldsymbol{C}\boldsymbol{C}^T - \boldsymbol{1}, \tag{2.18}$$

$$\frac{\partial J}{\partial \boldsymbol{\gamma}} = \det\,\boldsymbol{C} - 1, \tag{2.19}$$

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

$$\boldsymbol{L}\boldsymbol{C} = \boldsymbol{W}. \tag{2.20}$$

At this point, we perform a SVD on the matrix $\boldsymbol{W}$ to find $\boldsymbol{C}$:

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{2.21}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are square, orthogonal matrices, and $\boldsymbol{D} = \mathrm{diag}(d_1, d_2, d_3)$ is a diagonal matrix of singular values, $d_1 \geq d_2 \geq d_3 \geq 0$. If a unique solution for $\boldsymbol{C}$ exists, it is:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \tag{2.22}$$

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

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

After estimating the rotation, $\hat{\boldsymbol{C}}_{lm}$, we can get the estimated translation:

$$\hat{\boldsymbol{r}}_m^{lm} = \boldsymbol{p} - \hat{\boldsymbol{C}}_{lm}, \tag{2.23}$$

and combine everything in the transformation matrix,

$$\hat{\boldsymbol{T}}_{lm} = \begin{bmatrix} \hat{\boldsymbol{C}}_{lm} & -\hat{\boldsymbol{C}}_{lm}\hat{\boldsymbol{r}}_m^{lm} \\ \boldsymbol{0}^T & 1 \end{bmatrix}. \tag{2.24}$$

## 2.3 Visual Teach & Repeat

VT&R is an accurate vision-based metric route-following algorithm developed for navigating in unstructured outdoor environments [2]. The overall work flow of the VT&R system is illustrated at a high level in Figure 2.2.

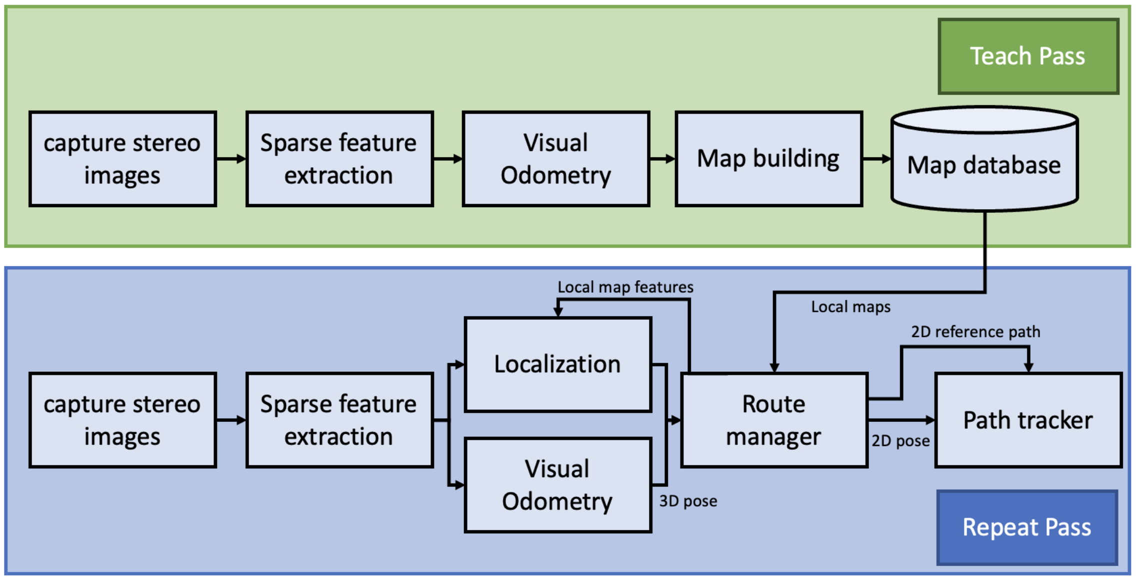


Figure 2.2: **VT&R high-level overview**. During the teach phase, image capture and feature extraction is followed by VO for pose estimation. The map stores vertices with 3D landmarks and the relative poses between vertices. During the repeat phase, the localization block computes a relative pose that provides the offset to the path relative the taught path, which is needed for path following.

In the teach phase, a user drives the robot manually to teach a path, while the system builds a local relative pose map that stores 3D landmarks that are triangulated from the extracted visual features. During the repeat phase, visual odometry and localization are combined to estimate the pose change of the robot relative to the visual map.

## 2.3.1 Single-Experience Localization

The overview of Single-Experience Localization is shown in Figure 2.3.

### Teach Phase

During the teach phase, the user manually drives the robot to teach a new path, where a relative local map of the path is created and stored as a pose graph. When the stereo camera captures a new stereo image, the individual images are rectified and converted to grayscale. Next, we obtain the depth values by performing stereo matching on the extracted SURF keypoints. Then, the keypoints are triangulated to generate the associated 3D landmarks. The VO 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.

After data association, Random Sample Consensus (RANSAC) [10] is applied to the matched landmarks to reject outliers, where the resulting inliers are passed to the pose estimator. 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 drops below a certain threshold, a new vertex is added to the pose graph.

#### Repeat Phase

In the repeat phase, the robot autonomously repeats the taught path while localizing against the built visual map. The task is to perform metric localization on the robot with respect to the 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 moves, and localization corrects the pose estimate using sensor measurements.

Live images are captured with the stereo camera, then rectified and converted to grayscale. After SURF features are extracted, the keypoints are triangulated to find 3D landmarks, and are matched to existing 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 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 of VO, RANSAC is used for outlier rejection, and the pose, $T_{lm}$, 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 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 [11].

### 2.3.2 Multi-Experience Localization

When relying on classical handcrafted features such as SURF [4], feature matching tends to fail once appearance change between the live images and the map images become too large. Multi-Experience Localization tackles drastic appearance change by using intermediate experiences to bridge the appearance gap as shown in Figure 2.4. The data for each repeat run is added to the pose graph map to build a spatio-temporal pose graph (SPTG), where the temporal edges record the relative transforms between vertices on the teach run (i.e., privileged run), and the spatial edges represent the relative pose transforms between vertices from each repeat runs with the teach run. These poses are obtained with localization during repeats. Finally, the temporal relative transformations between adjacent vertices that are obtained with VO are also recorded for each experience.

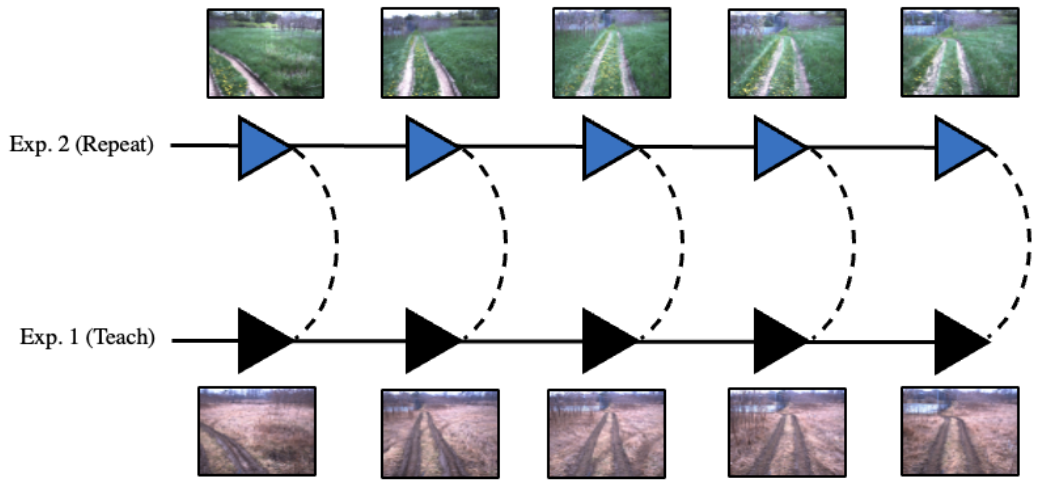


Figure 2.3: **Single-experience localization**. The user teaches a path by manually driving the robot and build a local visual map. The path is repeated autonomously during the repeat phase by directly localizing against the visual map.

During MEL, 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 selects a subset of the stored experiences that are most relevant (i.e., similar to the current conditions) to localize. However, the downside of MEL is the fact that experiences must be collected continually as the environment changes. Moreover, MEL can not easily generalize to a new path as it would require collecting and storing a new set of intermediate experiences to capture environmental changes.

## 2.4 UTIAS Datasets

In this thesis, we make use of two publicly available outdoor datasets that were collected at the University of Toronto Institute of Aerospace Studies (UTIAS) back in 2017. The data were gathered during autonomous path following with the Clearpath Grizzly robot using multi-experience VT&R, which uses a Bumblebee XB3 stereo camera with 24 cm baseline and 16 Hz frame rate as the sensor. The robot has two

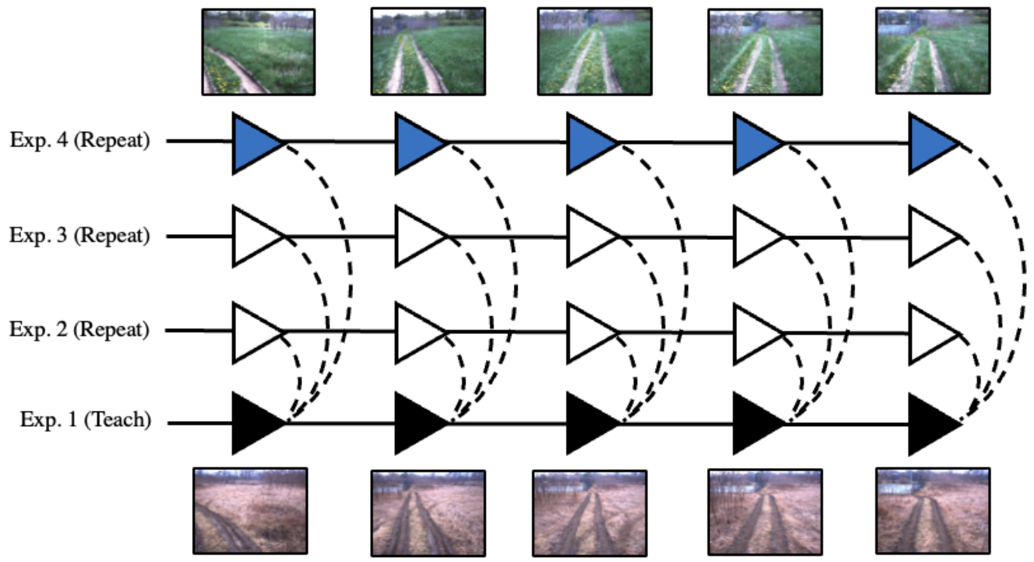


Figure 2.4: **Multi-experience localization** relies on intermediate experiences to localize a live frame against the mapped teach run by bridging the large appearance gap.

forward-facing and two backward-facing LED lights that can be used while driving in low-light environments. The VT&R data are stored in a pose graph structure, where the sensor data are stored as vertices and relative poses as edges in a spatio-temporal pose graph. In this section, we provide details on each dataset.

## 2.4.1 UTIAS In-the-Dark

The data in the UTIAS In-The-Dark dataset is collected using multi-experience VT&R across all lighting conditions over a 31-hour window. The total path is approximately 250 metres long and was collected around the Mars Dome building. The dataset contains 39 experiences with a total of 72,666 stereo image pairs. The number of stereo image pairs for each experience varies between 862 and 4042.

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. The aerial view of the path is shown in Figure 2.6a.

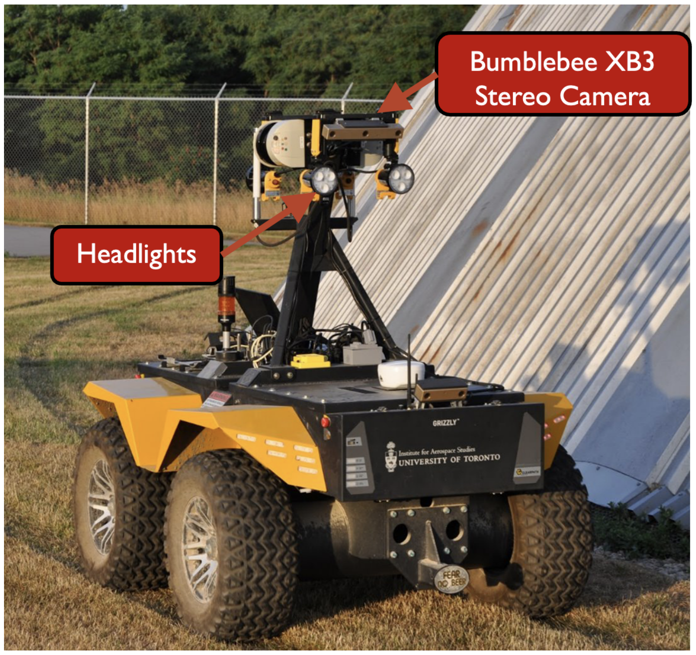
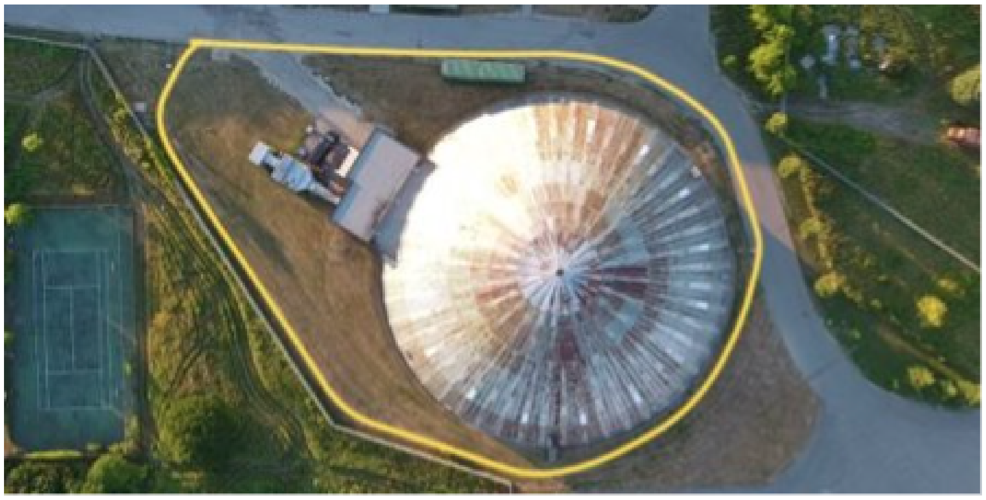


Figure 2.5: **The Clearpath Grizzly Robot** with a forward facing Bumblebee XB3 stereo camera and LED headlights.

### 2.4.2 UTIAS Multiseason

The UTIAS Multiseason dataset consists of 136 total runs of the path, which were collected over 17 weeks in 2017, capturing changes in lighting as well as in weather. The total path is approximately 165 metres, and is mainly off-road in an unstructured environment with no permanent structures and dense vegetation. The dataset consists of experiences collected with varying snow accumulation, snow melting, muddy conditions, and green grass. The overhead view of the path is shown in Figure 2.6b


(a) UTIAS-In-the-Dark training paths.

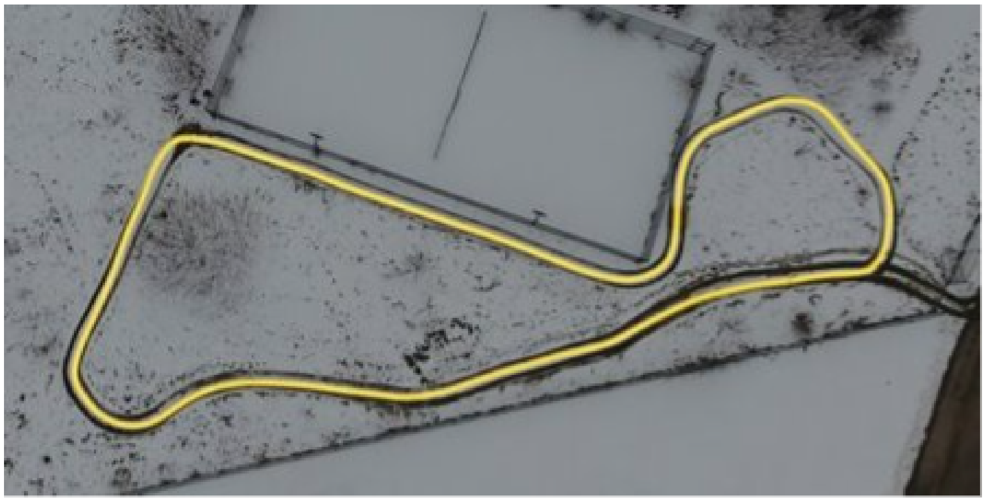

(b) UTIAS-Multiseason training paths.

Figure 2.6: **Training paths of the two UTIAS datasets**.

# Chapter 3

# Self-Supervised Feature Learning for Long-Term Localization

## 3.1 Introduction

Due to recent developments in the field of deep learning, many recent works use neural networks to directly predict relative poses [12–14] or absolute poses [15] from images for localization. Instead of directly learning poses from images, deep-learned interest-point detectors and descriptors [16–25] have gained popularity since they produce more accurate results than direct pose regression when combined with a classical pose estimator. In addition, Gridseth and Barfoot [16] have proven the effectiveness of deep-learned features in the VT&R framework under different illumination conditions. However, these feature learning networks typically require training with accurate image correspondences that are extracted from known scene geometry or ground-truth camera poses. Although some methods simplify the problem by collecting data using a stationary camera [18], applying known homographic adaptation [19, 23, 24], or rendering synthetic training data [19, 22], the generalizability on unseen data is limited.

In this chapter, we propose a novel self-supervised feature learning framework for long-term visual localization that does not require any ground-truth labels while preserving reasonable generalizability on unseen data. We identify that one of the main challenges is to find accurate whole-image correspondences in unstructured outdoor environment without ground-truth data. Since the ultimate goal is to learn features in a self-supervised manner, we do not want to presuppose the existence of local features. Especially, we want to avoid using deep-learned features that are pretrained on other labeled datasets with known image correspondences. Hence, we adopt a sequence-based visual place recognition algorithm (i.e., SeqSLAM [26]) that works directly with whole-image descriptors to generate coarse image alignment, which can be used for sampling training image pairs for feature learning pipeline. Our overall pipeline consists of two stages: place recognition and feature learning. During the first stage, we generate training samples by performing sequence-based visual place recognition on data collected under different seasons and lighting conditions. During the second stage, we train a feature learning network using the sampled training stereo image pairs from the previous stage to predict keypoints with associated descriptors and scores in an Expectation-Maximization loop without any ground-truth pose supervision. Lastly, we integrate the learned features in the VT&R framework for closed-loop localization.

The content in this chapter is published in IEEE Robotics and Automation Letters 2023 [7], and was presented at ICRA 2023.

## 3.2 Related Work

Traditionally, hand-crafted features have been commonly used for visual localization [3–5], but suffer from low repeatability on images under dramatic appearance changes [27]. Experience-based visual navigation systems [6,28,29] attempt to bridge

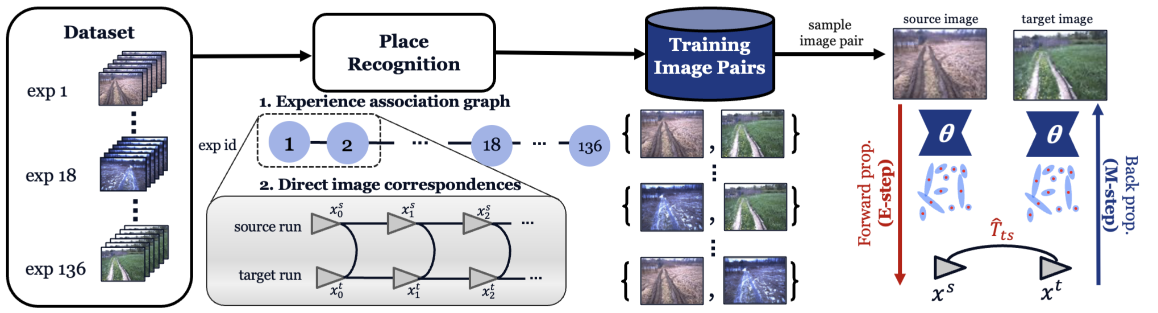


Figure 3.1: **Overall pipeline**. The dataset consists of multiple sequences (i.e., experiences) of image data organized by their collection times. During the place-recognition stage, we directly match neighbouring experiences using a place-recognition algorithm, and find indirect image correspondences by constructing an experience-association graph. We sample image pairs using the predicted image correspondences for the subsequent feature-learning stage. During the E-step of feature learning, the network predicts keypoints from the source and target images, which can be used to compute the relative transform. During the M-step, we use the estimated relative transform as a supervisory signal to optimize the network weights.

significant appearance changes by storing multiple appearances of the same location (i.e., experiences) and choosing the most relevant experiences for feature matching during online operation. However, experience-based methods have multiple downsides; These methods require storing a growing number of experiences to capture continuous environmental change, and can not generalize well to unseen environment.

Inspired by recent deep learning advances, there has been a wide range of deep-learning-based approaches for pose estimation to address the deficiencies of hand-crafted features while removing the reliance on intermediate experiences. Some methods tackle camera localization by training a CNN that directly learns relative poses [12–14] or absolute poses [15] from images. However, learning pose directly from image data can struggle with accuracy [30] as it is not trivial to encode the 3D scene geometry in a CNN model.

Other works focus on only learning the visual features or descriptors [1, 16–25], which can be integrated with a classical pose estimator for localization. For instance, Gridseth and Barfoot [16] and Sun et al. [24] have proven the effectiveness of deep-learned features against illumination changes by integrating with the VT&R pipeline for long-term visual localization. However, training networks for visual localization generally requires accurate image correspondences extracted from ground-truth camera poses or known scene geometry. Although Kasper et al. [25] proposed an unsupervised transform consistency loss to train features, it still requires GPS to generate training image pairs, ensuring that the query and reference images have sufficient visual overlap. Alternatively, Verdie et al. [18] use a stationary camera with a fixed transform to the world frame to observe the same scene under different illumination conditions, whereas other methods obtain image correspondences from either using synthetically rendered data [19, 22] or applying known transformations (e.g., colour shift, homographic transform) to real-world images [19, 23, 24]. However, these approaches either limit the diversity of the training data, or struggle with sim2real domain gap, which greatly limits the generalizability of the learned model on unseen data.

In this chapter, we propose to find image correspondences using a visual place-recognition algorithm without any ground-truth labels. We will discuss some relevant place-recognition approaches in the rest of the section. Since our goal is to sample training data for feature learning in a self-supervised manner, we restrict our discussion to hand-crafted algorithms that do not involve any learning or human annotations.

Lowry et al. [31] provide a thorough overview of existing methods for place recognition. Traditional bag-of-words-based FAB-MAP2 [32] shows good performance, but tends to fail in the presence of dramatic appearance changes [33] due to the limited repeatability of local hand-crafted keypoints [3, 4] in changing environments [27]. In order to improve the performance of place recognition under appearance changes, various approaches have been developed to exploit the sequential nature of the data [26, 29, 34]. SeqSLAM [26] is one of the most recognized sequence-based algorithms that achieves significant performance improvements over FAB-MAP2 under appearance changes. SeqSLAM finds image correspondences by constructing a pairwise similarity matrix between the local query image sequence and a database image sequence without the need for keypoint extraction. Hansen et al. [35] further improve the overall flexibility of the sequence alignment procedure by using a hidden Markov model (HMM) to formulate a graph-based sequence search method in the similarity matrix. In order to match image sequences under severe seasonal changes more efficiently, Naseer et al. [36] formulate sequence matching as a minimum cost flow problem in an offline data-association graph. Vysotska et al. [37] extended this idea to build a data-association graph online using a pretrained CNN for feature extraction, and later proposed a hashing-based relocalization strategy [38] to improve data association for flexible trajectories. Additionally, Neubert et al. [39] detect similarities within the database to avoid unnecessary image matching between query and database images.

For our purposes, we adopt SeqSLAM aided by Visual Odometry (VO) to find direct image correspondences between temporally neighbouring experiences, and indirectly link further experiences by building an offline experience-association graph.

## 3.3 Methodology

We propose a two-stage pipeline as shown in Figure 3.1, which includes: 1) place recognition and 2) self-supervised feature learning. During the place-recognition stage, we build a locally connected experience-association graph by matching neighbouring experiences with gradual perceptual changes, where we can sample image pairs from direct or indirect image correspondences between experiences using a graph-search algorithm. Our fully differentiable feature-learning pipeline takes a pair of source and target stereo images generated from the previous stage, and estimates

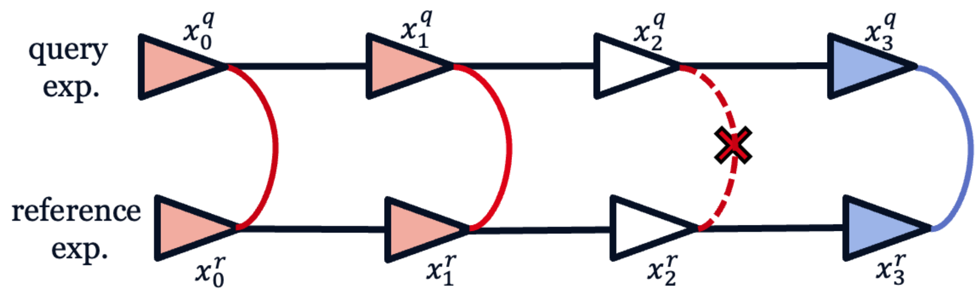


Figure 3.2: **Direct image correspondences**. The black horizontal edges represent the relative transforms between frames (known from VO). The red vertical edges represent candidate matches suggested by SeqSLAM. The red solid edges are validated by VO, whereas the dotted edges are rejected by VO. After the candidate match is rejected, VO can also potentially find a new match in the reference sequence that is closest to the query image, which is represented by a blue edge.

the relative pose, $\mathbf{T}_{ts} \in SE(3)$, between their corresponding frames. During the feature-learning stage, we train a neural network in an Expectation-Maximization loop. In the E-Step, the network predicts sparse local keypoints with associated descriptors and scores for both input images. We find point correspondences between the predicted keypoints from the source and target images, which are used in a differentiable pose estimator to estimate the relative pose change. In the M-step, we construct a self-supervised keypoint loss using the estimated relative keypoint loss as a supervisory signal to optimize the network weights.

## 3.3.1 Place Recognition

We define the set of training data $\mathcal{D} = (s_0, ..., s_M)$ as a temporally ordered set of $M$ experiences, where each experience is defined as a sequence of $N$ consecutive images $s_i = (x_0^i, ..., x_N^i)$. The ultimate goal is to generate direct or indirect image correspondences between any two experiences in the dataset $\mathcal{D}$. However, matching images of the same place under appearance change is a non-trivial task, as the appearance of the same place is likely to change substantially due to different illumination conditions and seasonal changes. A naive approach is to directly match images between experiences using proximity provided by Visual Odometry (VO). However, this results in poor image correspondences for longer routes due to drift errors in VO. Alternatively, we can exploit the sequential nature of the data and use SeqSLAM [26] to match different image sequences using patch-normalized image representation, but face two main problems:

a) SeqSLAM is capable of matching experiences with gradual appearance changes but struggles with significant appearance gap.

b) The raw SeqSLAM matching results may contain outliers and discontinuities.

To mitigate these issues, we combine SeqSLAM with VO to improve the direct matching results, and construct an experience-association graph to extract indirect image correspondences between experiences. To sufficiently limit scope, we make the following assumptions:

a) All experiences roughly follow the same trajectory with the same starting and ending positions.

b) All experiences are temporally ordered by their collection times. Nearby experiences are collected within a relatively short time frame, thus having similar appearances with each other compared to further experiences.

### Direct Image Correspondences

According to the second assumption, we assume the temporally neighbouring experiences are collected within a shorter time frame, such that the appearance changes are small enough for us to find direct image correspondences using SeqSLAM and VO. For each query experience $s_i$, it is directly matched with $k$ previous experiences $\{s_{i-k}, ..., s_{i-1}\}$, which are denoted as the reference experiences.

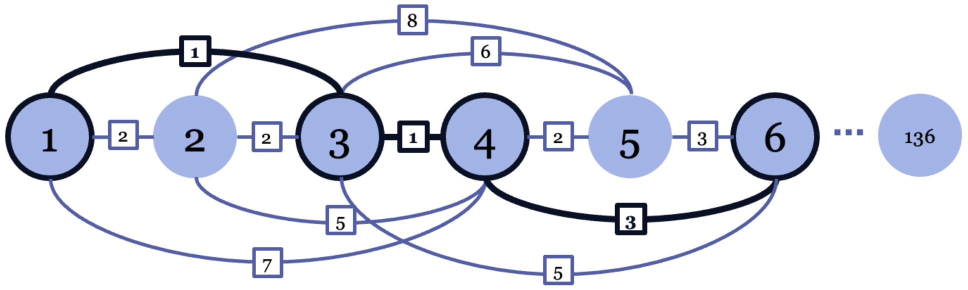


Figure 3.3: **Experience-Association Graph**. Each vertex represents an experience, which is locally connected to 3 previous neighbours. and each edge represents direct image correspondences. The assigned cost term on each edge reflects the quality of the match. To compute indirect image correspondences between experience 1 and 6, the minimum cost path is indicated in black.

As shown in Figure 3.8, SeqSLAM constructs a difference matrix by computing image-by-image dissimilarity scores between the two sequences using full-image descriptors, then finds raw image matches through the full matching matrix with the smallest sum of dissimilarity scores. We further validate the matching results using VO to reject outliers as illustrated in Figure 3.2.

For each image in the query experience, we use the raw SeqSLAM matching results to suggest a candidate match in the reference experience, then validate the match using VO. Assuming we have successfully matched and validated the image pair $\{q_0, r_0\}$ using SeqSLAM and VO, we consider the image pair as the last validated match $m$, then proceed to the next image in the query experience, $q_1$. After retrieving the candidate match $r_1$ from the reference experience using SeqSLAM, we use VO to estimate the relative transform between the last validated image and the current image in the query and reference experiences, namely $\hat{\mathbf{T}}_{q_1,q_0}$ and $\hat{\mathbf{T}}_{r_1,r_0}$. Since the last validated image pair $m = \{q_0, r_0\}$ are two images of the same place, we can approximate the relative transform $\hat{\mathbf{T}}_{r_0,q_0}$ with an identity matrix $\mathbf{I}$. As a result, the relative transform between the query and reference images can be estimated as:

$$\hat{\mathbf{T}}_{r_1,q_1} = \hat{\mathbf{T}}_{r_1,r_0}\hat{\mathbf{T}}_{r_0,q_0}\hat{\mathbf{T}}_{q_1,q_0}^{-1} \approx \hat{\mathbf{T}}_{r_1,r_0}\hat{\mathbf{T}}_{q_1,q_0}^{-1} = \begin{bmatrix} \mathbf{C}_{r_1,q_1} & \mathbf{r}_{r_1,q_1} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.1}$$

The absolute distance between $q_1$ and $r_1$ can be computed as:

$$d_{r_1,q_1} = \left\| \mathbf{r}_{r_1,q_1} \right\|_2^2 . \tag{3.2}$$

We compare the distance $d_{r_1,q_1}$ with a pre-specified threshold value $e$ to check if the candidate match is consistent with VO. We discuss two emerging cases as follows:

(a) $d_{r_1,q_1} \leq e$. The candidate match suggested by SeqSLAM is validated by VO. Hence, we update the last validated match $m$ to be $\{q_1, r_1\}$ when processing the next image in the query sequence.

(b) $d_{r_1,q_1} > e$. The candidate match suggested by SeqSLAM is rejected by VO. As a replacement, we retrieve a new image $r_j$ from the reference experience such that the distance between $r_j$ and $q_1$ is minimized. Note that we do not update the last validated match $m$ in this case.

Since the last validated match $m$ is frequently updated, we usually only need to compute VO for a relatively short time frame starting from the last validated match. Hence, the matching results are less likely to be affected by VO drift. The detailed procedure of direct image correspondences is shown in Algorithm 1

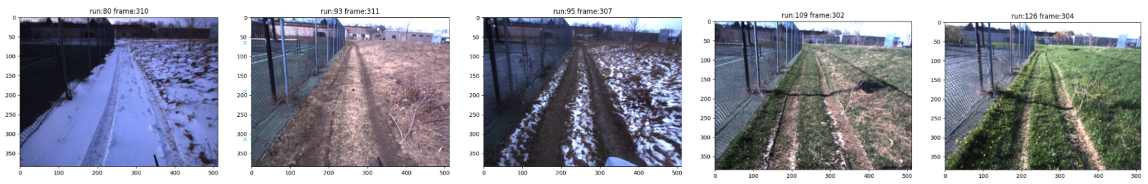
### Indirect Image Correspondences

According to the second assumption, relatively similar experiences are temporally closer together, whereas dissimilar experiences are further apart. Hence, we can find image correspondences between further experiences indirectly using bridging experiences. After finding direct image correspondences between neighbouring experiences, we can build a locally connected graph $G = \{V, E\}$, where $V$ is a set of vertices representing all experiences in the dataset, and $E$ is a set of edges representing direct image correspondences between the vertices. Each vertex is connected to $k$ previous vertices, and Figure 3.3 shows an example when $k = 3$.

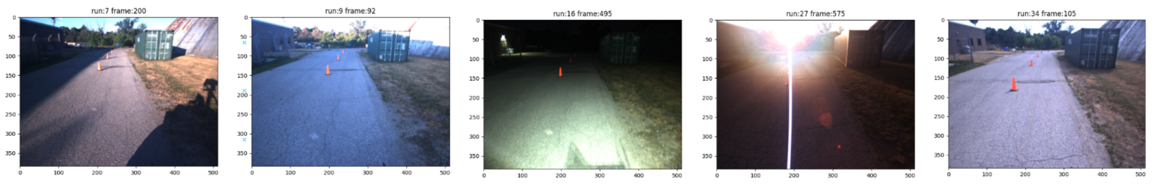
We assign a cost term to each edge in $E$ to indicate the quality of the match. The cost term is defined as the number of frames rejected by VO out of the total number frames in the source sequence, where a higher cost represents a poorer match and a lower cost represents a better match. To generate image correspondences between two experiences that are not directly connected, we can find the minimum-cost path in the graph using a graph-search algorithm to indirectly extract image correspondences. The detailed procedure of indirect image correspondences is shown in Algorithm 2

**Algorithm 1:** Direct Image Correspondences

```
Input: $s_q$, $s_r$ // query run and reference run
Parameters: $e$ // threshold
Output: $m$ // frame matches
$M_{raw} = SeqSLAM(s_q, s_r)$ ;                // compute raw matches using SeqSLAM
prevValidatedMatch = $[q_0, r_0]$ ;
$M = \{\}$;
// loop through each frame in the query sequence
for $q_i$ in $s_q$ do
    // retrieve the candidate matched frame in the reference experience
       using SeqSLAM
    $r_i = M_{raw}[q_i]$;
    $d_{q_i r_i} = getDistFromVO(q_i, r_i,$ prevValidatedMatch$)$;
    if $d < e$ then
        // validate the matched frame using VO
        prevValidatedMatch = $[q_i, r_i]$;
    else
        // correct the matched reference frame using VO
        $r_i = getClosestFrameFromVO(q_i, r_i,$ prevValidatedMatch$)$ ;
    end
    M$[q_i] = r_i$;
end
return $M$
```



(a) Sample frame correspondences generated by seqSLAM for Multiseason dataset across different runs under seasonal changes.



(b) Sample frame correspondences generated by seqSLAM for In-the-Dark dataset across different runs under lighting changes.

Figure 3.4: **Sample image correspondences generated by seqSLAM.**

**Algorithm 2:** Indirect Image Correspondences

```
Input: totalRuns[:] ;                    // a list of experiences with length N
Output: $G = \{V, E\}$ ;                                              // graph
totalRuns.sort();
$G$ = Graph();
for $i \leftarrow 1$ to $N$ do
    $s_i \leftarrow$ totalRuns[$i$];
    $G.addVertex(s_i)$;
    // loop through k nearest neighbors
    for $j \leftarrow i$ to $i + k$ do
        // compute direct image correspondences and return the quality of the
           match
        $c_{ij} = DirectImageCorrespondences(s_i, s_j)$;
        $G.addVertex(s_j)$;
        $G.addEdge(s_i, s_j, c_{ij})$;
    end
    return $G$
end
```

### 3.3.2 Feature Learning

#### Network Architecture

Our network is adapted from the architecture presented by [1, 16], which is a U-Net style convolutional encoder-multi-decoder architecture to output keypoints, descriptors, and scores based on visual inputs as shown in Figure 3.5. The encoder is a VGG16 network [40] pretrained on the ImageNet dataset [41], truncated after the *conv_5_3* layer.

The keypoint-locations decoder predicts the sub-pixel locations of each sparse 2D keypoint, $\mathbf{q} = [u_l, v_l]^T$, in the left stereo image. To achieve this, we equally divide the image into equally sized $16 \times 16$ square cells, with each generating a single candidate keypoint. We then apply a spatial softmax on each cell and take a weighted average of the pixel coordinates to return the sub-pixel keypoint locations.

In addition, the network computes the scores by applying a sigmoid function to the output of the second decoder branch. The scores $s$ are mapped to $[0, 1]$, which predict how useful a keypoint is for pose estimation. The network assigns all static structure in the scene a score of 1, and all noise, moving objects and empty regions a score of 0.

Finally, we generate dense descriptors for each pixel by resizing and concatenating the output of each encoder layer, which results in a descriptor vector, $\mathbf{d} \in \mathbb{R}^{960}$. The descriptors aim to uniquely identify real-world locations under keypoints so that we can relate points by comparing descriptor similarity.

#### E-Step: Pose Estimation

After generating $N$ keypoint predictions for the source image, we need to perform data association between the keypoints in the source and target images by performing a dense search for optimum keypoint locations in the target image as shown in

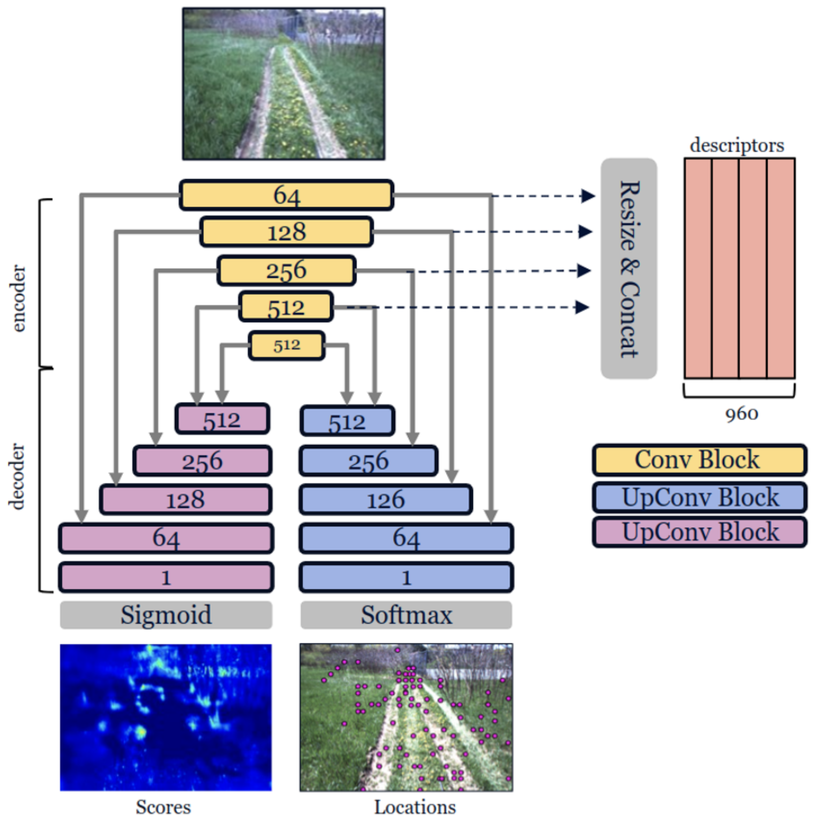


Figure 3.5: **Feature Network Architecture**. It takes an image as input, and outputs keypoint locations, and scores. Descriptors are generated for all pixels by resizing and concatenating the output feature maps of each encoder layer.

Algorithm 3. For each keypoint in the source image, we compute a matched point in the target image by taking the weighted sum of all image coordinates in the target image as depicted in Figure 3.6. The matched point in the target image is computed by:

$$\hat{\mathbf{q}}_t^i = \sum_{j=1}^{M} \sigma(\tau f_{\text{zncc}}(\mathbf{d}_s^i, \mathbf{d}_t^j))\mathbf{q}_t^j, \tag{3.3}$$

where M is the total number of pixels in the target image, $f_{\text{zncc}}(\cdot)$ computes the zero-normalized cross correlation (ZNCC) between the descriptors, which is equivalent to cosine distance for real matrices. $\sigma(\cdot)$ takes the temperature-weighted softmax with $\tau$ as the temperature. Finally, we find the descriptor, $\hat{\mathbf{d}}_t^i$, and score, $\hat{s}_t^i$, for each computed target keypoint. Given a set of keypoint locations we can extract keypoint

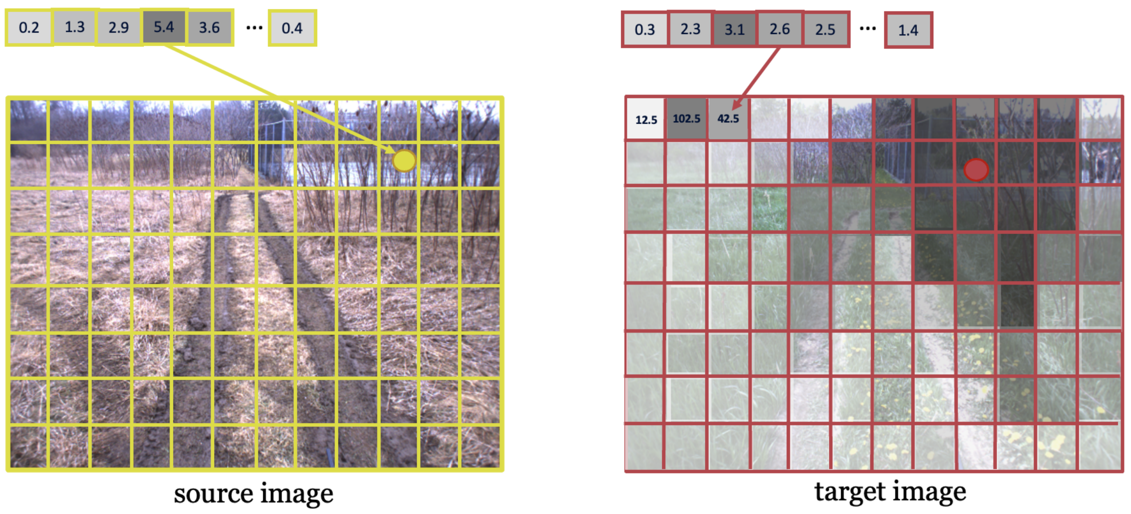


Figure 3.6: **Keypoint matching between the source and target images**. Descriptors of each pixel in the source and target images are compared using ZNCC, where the resulting value reflects how well descriptors match. For each keypoint in the source image, we compute a matched point in the target image by taking the weighted sum over all image coordinates in the target image.

descriptors and scores using a sampling function, @, such that $D@q$ takes a bilinear interpolation of dense feature map $D$ at coordinates $q$.

For the matched 2D keypoints, we compute their corresponding 3D coordinates using an inverse stereo camera model. The camera model, $\mathbf{g}(\cdot)$, maps a 3D point, $\mathbf{p} = [x\ y\ z]^T$, in the camera frame to a left stereo image coordinate, $\mathbf{q}$, as follows:

$$\mathbf{y} = \begin{bmatrix} u_l \\ v_l \\ d \end{bmatrix} = \begin{bmatrix} \mathbf{q} \\ d \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 0 & f_u b \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{3.4}$$

where $f_u$ and $f_v$ are the horizontal and vertical focal lengths in pixels, $c_u$ and $c_v$ are the camera's horizontal and vertical optical center coordinates in pixels, $d = u_l - u_r$ is the disparity obtained from stereo matching, and $b$ is the baseline in metres. We use the inverse stereo camera model to get each keypoint's 3D coordinates:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{g}^{-1}(\mathbf{y}) = \frac{b}{d} \begin{bmatrix} u_l - c_u \\ \frac{f_u}{f_v}(v_l - c_v) \\ f_u \end{bmatrix}. \tag{3.5}$$

Given the corresponding 2D keypoints $\{\mathbf{q}_s^i, \hat{\mathbf{q}}_t^i\}$ from the source and target images, we use (3.5) to compute their 3D coordinates, $\{\mathbf{p}_s^i, \hat{\mathbf{p}}_t^i\}$. In addition, their descriptors and scores are: $\{\mathbf{d}_s^i, \hat{\mathbf{d}}_t^i\}$, and $\{s_s^i, \hat{s}_t^i\}$.

We then perform RANSAC to reject outliers prior to pose estimation to find features that are geometrically consistent with each other. This step is essential to provide a reasonable guess of the relative pose in the beginning of training to help the algorithm converge.

Given the inlier 3D keypoints, we can estimate the relative pose from the source to the target by minimizing the following cost using Singular Value Decomposition (SVD) as described in Chapter 2.2 :

$$J = \sum_{i=1}^{N} w^i \left\| (\mathbf{C}_{ts}\mathbf{p}_s^i + \mathbf{r}_t^{st}) - \hat{\mathbf{p}}_t^i \right\|_2^2, \tag{3.6}$$

where the weight for a matched point pair is a combination of the learned point scores and how well the descriptors match:

$$w^i = \frac{1}{2}(f_{\text{zncc}}(\mathbf{d}_s^i, \hat{\mathbf{d}}_t^i) + 1) s_s^i \hat{s}_t^i. \tag{3.7}$$

Finally, we obtain the relative transform between the source and target images as:

$$\hat{\mathbf{T}}_{ts} = \begin{bmatrix} \mathbf{C}_{ts} & \mathbf{r}_t^{st} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.8}$$

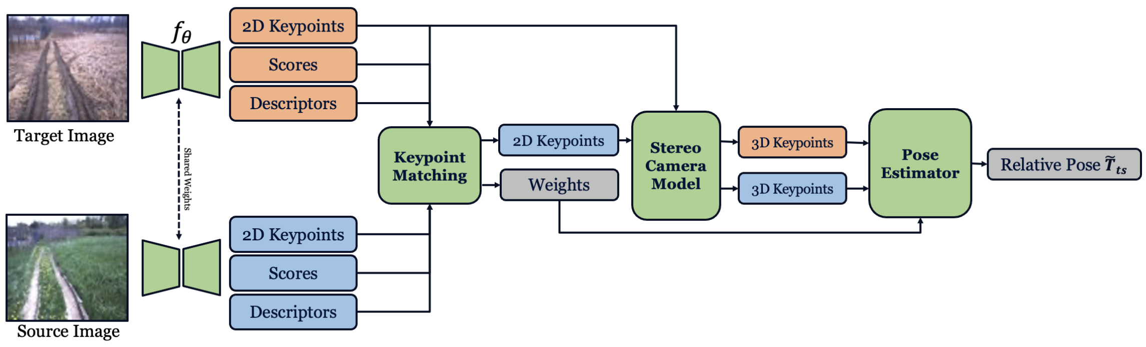


Figure 3.7: **Feature Pipeline**. We pass a source image and a target image to a neural network that predicts keypoints, descriptors, and scores. The learned keypoints are matched between the source and target image frames. We use the stereo camera model to find the corresponding 3D coordinates of the match keypoints, which are passed to a differentiable pose estimator to estimate the relative pose between the source and target images.

The overall procedure is illustrated in Figure 3.7.

**Algorithm 3:** Differentiable Point Matching [1]

```
Input: $\mathbf{q_s}$                                        // source keypoint pixel locations
       $\mathbf{D_s}, \mathbf{D_t}$                           // source and target descriptor maps
       $\mathbf{S_s}, \mathbf{S_t}$                           // source and target score maps
Parameters:
       $\mathrm{T}$                                           // descriptor cosine distance softmax temperature
       $\mathbf{X}$                                           // pixel locations map
Output: $\mathbf{q_d}$                                        // target keypoint locations
       $\mathbf{w}$                                           // point match weights
for $i \leftarrow 1$ to $n$ do
    $\mathbf{d_{si}} \leftarrow \ell_2(\mathbf{D_s}@\mathbf{q_{si}})$          // extract and normalize source point descriptors
    $\mathbf{C_i} \leftarrow \mathbf{d_{si}} \odot \mathbf{D_d}$              // pixelwise cosine distance to target descriptor map
    $\mathbf{S} \leftarrow \sigma(T\mathbf{C_i})$                              // apply temperature weighted softmax
    $\mathbf{q_{di}} \leftarrow \mathbf{S} \odot \mathbf{X}$                  // extract target point pixel coordinates
    $\mathbf{d_{di}} \leftarrow \ell_2(\mathbf{D_d}@\mathbf{q_{di}})$          // extract and normalize target point descriptor
    $s_{si} \leftarrow \mathbf{S_s}@\mathbf{q_{si}},\ s_{di} \leftarrow \mathbf{S_d}@\mathbf{q_{di}}$   // extract source and target scores
    $w_i \leftarrow \frac{1}{2}(\mathbf{d_{si}} \odot \mathbf{d_{di}} + 1)s_{si}s_{di}$   // compute weight for point match
```

**M-Step: Feature Optimization**

Barnes and Posner [1] and Gridseth and Barfoot [16] use the ground-truth transform between the source and target images to construct a supervised pose loss. Since

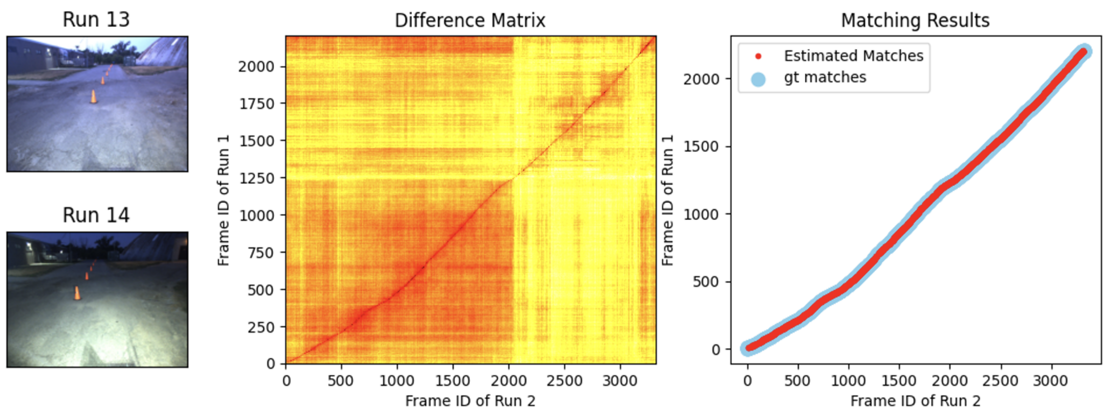


Figure 3.8: **Frame-level topological matching results** between two neighbouring runs, run 13 and run 14, in the UTIAS-In-the-Dark dataset. The raw difference matrix generated by SeqSLAM algorithm and the processed matching results is shown.

we do not rely on any ground-truth data, we directly use the estimated transform as a supervisory signal to construct a keypoint loss without any ground-truth pose information. We transform the predicted source keypoints $\mathbf{p}_s^i$ to the target frame using the estimated relative transform $\hat{\mathbf{T}}_{ts}$, and define a keypoint loss as follows:

$$\mathcal{L} = \sum_{i=1}^{N} ||\hat{\mathbf{T}}_{ts}\mathbf{p}_s^i - \hat{\mathbf{p}}_t^i||_2^2. \tag{3.9}$$

## 3.4 Experiments

### 3.4.1 Datasets

For training, we use the same datasets as described in Chapter 2.4 to test localization for a route-following problem, which is similar to [16]. Unlike [16], we do not rely on the spatio-temporal pose graph to extract ground-truth relative transforms between vertices for sampling image pairs or for training.

We use data from two different paths for training, which are included in the In-the-Dark and Multiseason datasets respectively. The In-The-Dark dataset contains 39 runs of a path collected at the campus of UTIAS in summer 2016. The same

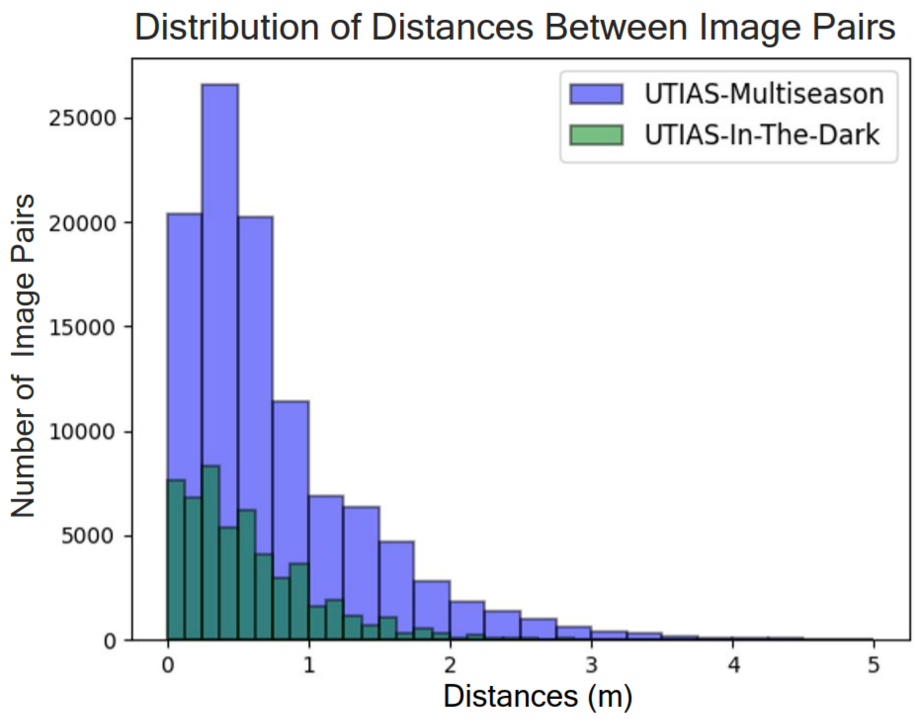


Figure 3.9: **Distribution of distances between sampled image pairs** of Multiseason and In-the-Dark datasets.

path was repeated hourly for 30 hours, which captures incremental lighting changes throughout the day. The Multiseason dataset contains 136 runs of a path in an area with more vegetation. The path was consistently repeated from January until May 2017, which captures diverse seasonal and weather conditions throughout the year. Both datasets are publicly available.

### 3.4.2 Training Data Generation using Topological Localization

We generate direct and indirect image correspondences across different sequences in the training datasets as described in Section 4.3, which are used to sample 100,000 training image pairs and 20,000 validation image pairs. To evaluate the accuracy of the sampled training data, we extract the relative transform between the source and target image from the spatio-temporal graph for all sampled image pairs, then

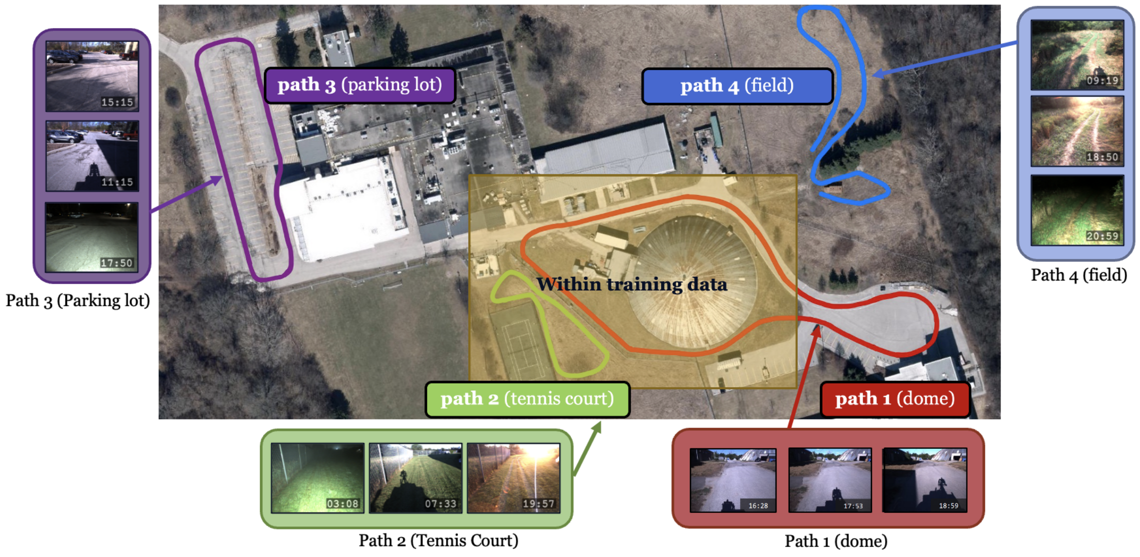


Figure 3.10: **Training and testing paths**. The training paths from the datasets are highlighted by the yellow region. Path 1 (tennis court) and part of path 2 (dome)is collected in the same region as the training data, whereas the bottom right portion of path 2 (dome), path 3 (parking lot) and path 4 (field) are not in training data.

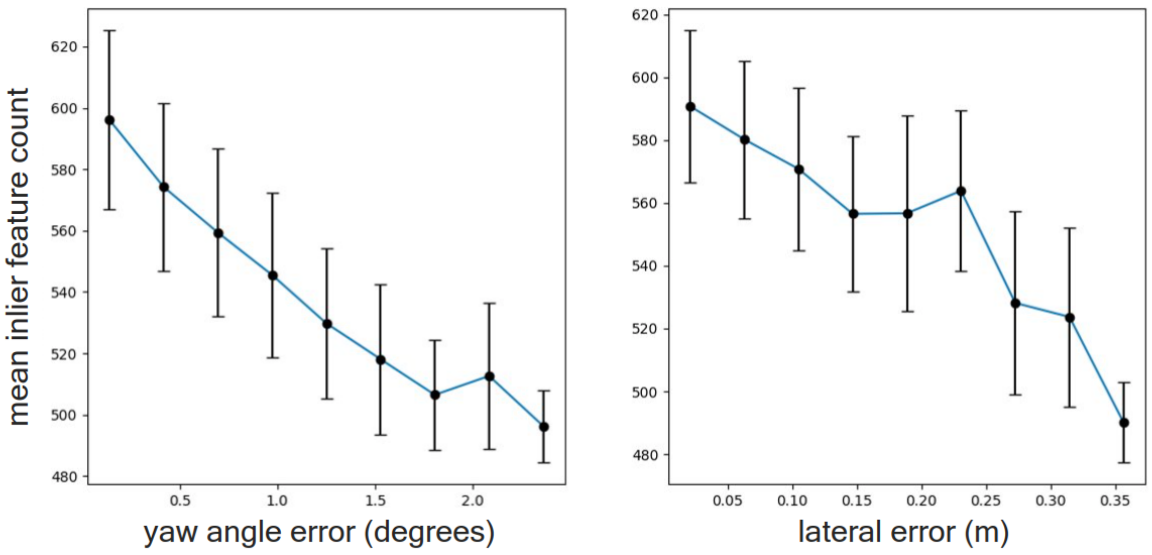
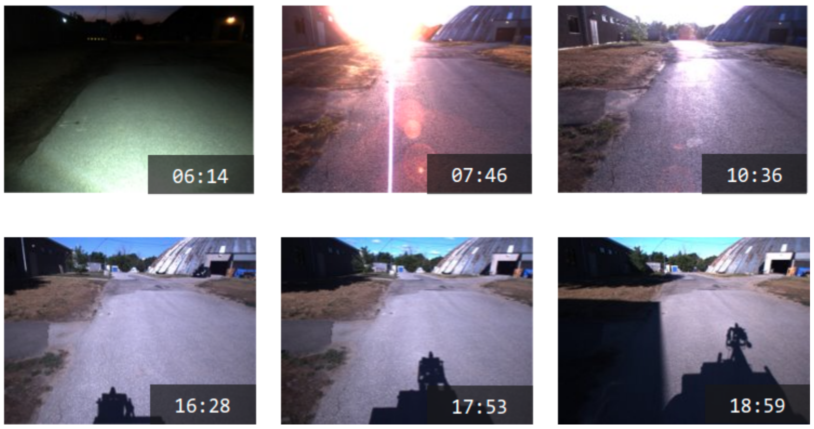
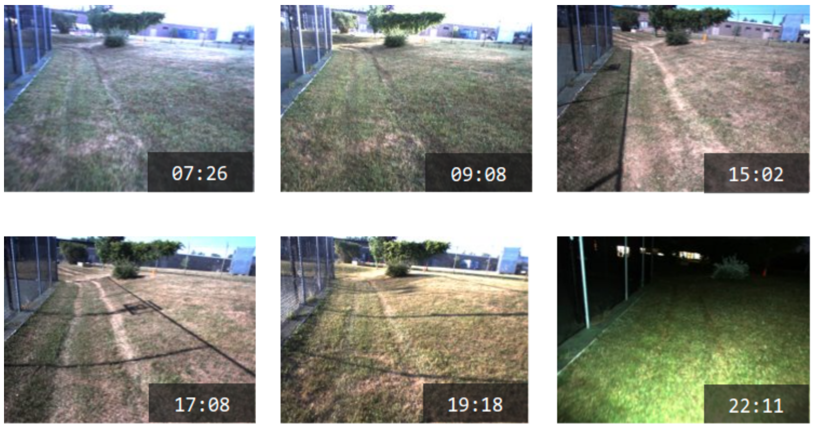


Figure 3.11: **Median feature inlier count** (with $1\sigma$ bound) for different lateral and yaw angle errors of the UTIAS-In-the-Dark test set. The median feature inlier count is highly correlated with the path-following errors, which serves as a reasonable proxy for localization performance in VT&R.

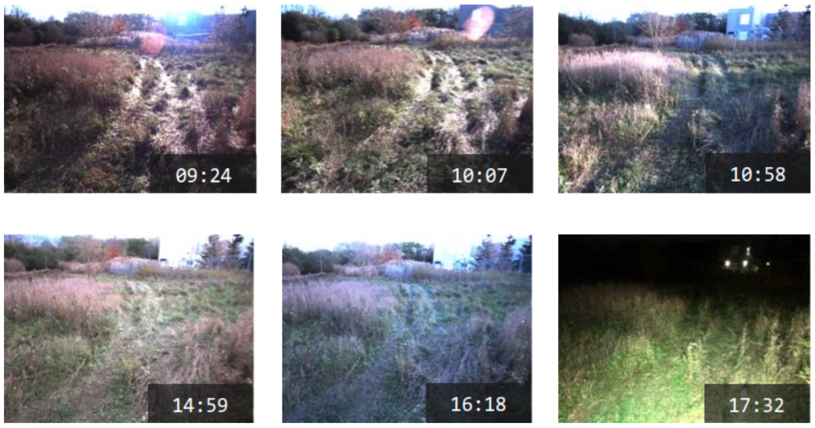
plot the distribution of the absolute distances in Figure 3.9. As can be seen from the histogram, majority of the image pairs are within 3 metres of each other, which



(a) Selected camera views during closed-loop experiment for path 1 (dome).



(b) Selected camera views during closed-loop experiment for path 2 (tennis court).



(c) Selected camera views during offline experiment for path 4 (field).

Figure 3.12: **Selected camera views** from closed-loop and offline experiments labeled with collection time (hh:mm).

Table 3.1: **Comparison of the path-following errors and the median number of inliers** for SuperPoint [19], D2-Net [17], Supervised U-Net [16], and our self-supervised method. We construct 6 teach-repeat pairs under different lighting changes for evaluation from the UTIAS-In-the-Dark test set, where we list the experience IDs as well as their collection time for each pair. We report the lateral errors $\Delta y$ in meters, and the yaw angle errors $\Delta\theta$ in degrees.

| | **2-28** | | | **4-17** | | | **11-4** | | |
|---|---|---|---|---|---|---|---|---|---|
| | 09:42 - 06:44 | | | 12:29 - 21:58 | | | 20:44 - 12:29 | | |
| | $\Delta y$ | $\Delta\theta$ | inliers | $\Delta y$ | $\Delta\theta$ | inliers | $\Delta y$ | $\Delta\theta$ | inliers |
| SuperPoint [19] | 0.18 | 0.78 | 60 | 0.14 | 1.2 | 36 | 0.22 | 1.53 | 72 |
| D2-Net [17] | 0.16 | 1.02 | 133 | 0.29 | 3.68 | 72 | 0.05 | 0.21 | 278 |
| Sup. U-Net [16] | **0.02** | **0.25** | **606** | **0.03** | **0.47** | **538** | **0.03** | 0.20 | 617 |
| Self Sup. U-Net (Our) | 0.03 | 0.34 | 579 | 0.06 | 0.50 | 536 | **0.03** | **0.19** | **619** |

| | **16-17** | | | **17-23** | | | **28-35** | | |
|---|---|---|---|---|---|---|---|---|---|
| | 21:48 - 21:58 | | | 21:58 - 05:34 | | | 06:44 - 11:26 | | |
| | $\Delta y$ | $\Delta\theta$ | inliers | $\Delta y$ | $\Delta\theta$ | inliers | $\Delta y$ | $\Delta\theta$ | inliers |
| SuperPoint [19] | 0.11 | 0.61 | 41 | 0.20 | 0.89 | 51 | 0.16 | 1.31 | 72 |
| D2-Net [17] | 0.02 | 0.33 | 459 | 0.09 | 1.29 | 129 | 0.18 | 1.55 | 122 |
| Sup. U-Net [16] | **0.02** | **0.31** | **614** | **0.03** | **0.35** | **550** | **0.03** | 0.29 | **588** |
| Self Sup. U-Net (Our) | **0.02** | 0.32 | 611 | 0.04 | 0.38 | 539 | 0.04 | **0.27** | 582 |

generally means sufficient overlapping visual field for training.

## 3.4.3 Training and Inference

We use the sampled image pairs to train the network. During training and inference, we discard outlier keypoints using RANSAC. The network is trained using the Adam optimizer with a learning rate of $10^{-5}$ on an NVIDIA Tesla V100 DGXS GPU. The encoder is a VGG16 network [40] pretrained on the ImageNet dataset [41], truncated after the *conv*_5_3 layer. The network is finetuned end-to-end on the UTIAS Multiseason and In-the-Dark datasets for 50 epochs.

After training the network, we integrate the learned keypoint detector and descriptor with the VT&R system to extract features for visual localization. While the user manually drives the robot to teach a path, VT&R creates a spatio-temporal pose graph that stores relative poses between keyframes. The path is autonomously repeated by computing VO and localizing keyframes. VT&R relies on SURF for VO, and utilizes the learned features along with a sparse descriptor matcher for localization.

We conduct three experiments to verify the localization performance of the learned features. We compare our methods to other existing learned features under lighting changes by calculating the path-following errors using the UTIAS-In-the-Dark test set.

In addition, we perform an offline localization ablation study using pre-collected rosbags [16] for paths 3 and 4. Lastly, we perform closed-loop experiments on three paths (path 1, path 2, and path 3) under different lighting conditions as shown in Figure 2.5. We set the maximum speed of the robot to 0.6 m/s. Selected camera views for the localization experiments are shown in Figure 3.12. For these two experiments, it was often not feasible to collect accurate RTK GPS ground truth as we drove in area where tree cover blocks the GPS signals. Hence, we report the median number of inliers in feature matching to reflect the performance of localization. We plot the median number of inliers against the path-following errors in both lateral and yaw directions using the UTIAS-In-the-Dark test set in Figure 3.11, which indicates that the median number of inliers is highly correlated with the path-following performance, showing that a higher inlier count is beneficial for accurate and robust localization.

### 3.4.4 Offline Localization: Learned Features Comparison

We compared the localization performance of our method to its supervised counterpart (Sup. U-Net) [16], SuperPoint [19] and D2-Net [17] in PyTorch. The pretrained networks provided by the authors are used to extract the keypoints, descriptors, and scores. We compute the path-following errors between six teach-repeat pairs sampled from the UTIAS-In-the-Dark test set, then report the lateral errors, yaw angle errors, and the median number of inliers in Table 4.1, where the ground truth poses are obtained from Multi-Experience Localization during data collection. Our method is competitive with its supervised counterpart, and outperforms SuperPoint and D2-Net in all metrics for all six localization experiments.

### 3.4.5 Offline Localization: Ablation Study

We perform offline experiments on paths 3 and 4 using pre-collected rosbag files, and present the results in Figure 3.13. Each path is played back 7 times from morning to nighttime with 100% localization success rate. In the same plot, we compare with different methods. In all experiments, SURF results in a significantly lower number of feature inliers, and fails to localize at the beginning of three different repeats. This shows that the learned features are superior in terms of handling illumination changes.

When trained on UTIAS datasets only (i.e., not pretrained on ImageNet), we see a slight performance drop in the self-supervised method when compared to the supervised counterpart. Although SeqSLAM matching is not perfect, using SeqSLAM matching to generate training pairs results in a competitive performance compared to using a ground-truth image matches. This suggests the effectiveness of using a place-recognition algorithm to find coarse image correspondences for detailed metric feature learning.

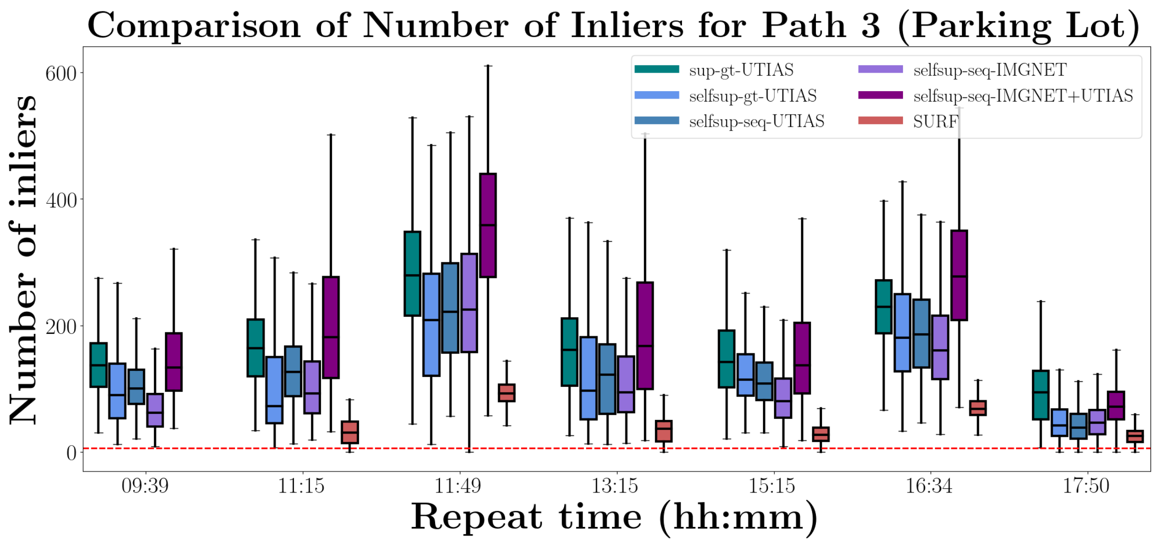
When trained on ImageNet only, the self-supervised method does not perform well compared to other methods, and failed to localize in two runs for path 4. However, the best performance for the self-supervised method is achieved when pretrained on ImageNet and finetuned on UTIAS datasets, which even outperforms the supervised method (i.e., not pretrained on ImageNet). It shows that by training on UTIAS datasets, it allows the model to generalize better to unseen data.

### 3.4.6 Online VT&R: Closed-Loop Experiments

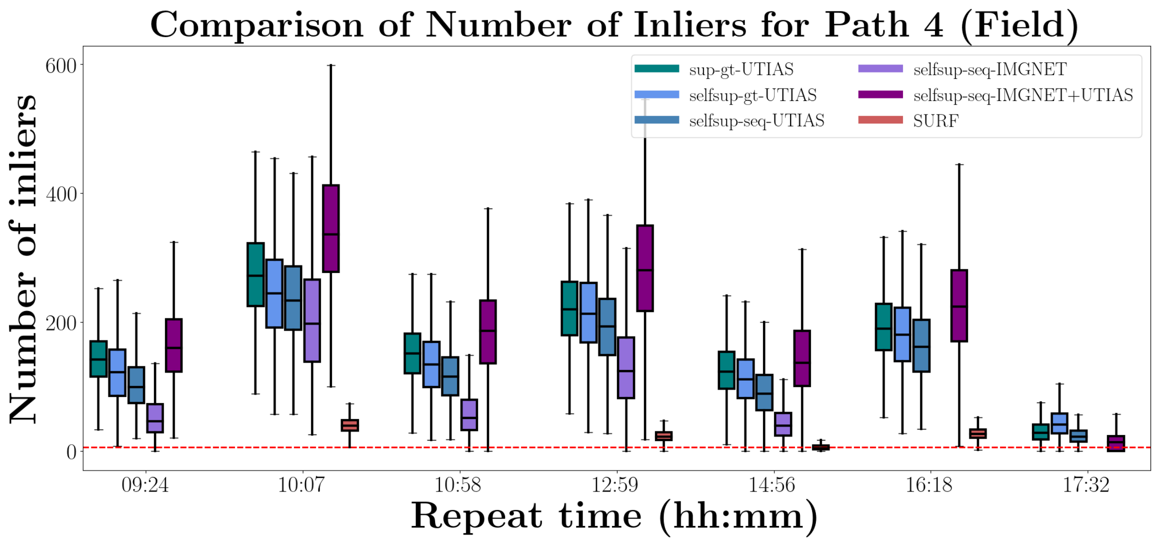
We perform closed-loop experiments using online VT&R on path 1, 2, and 3 from Figure 3.10 with 100% success rate. Path 1 (dome) and path 2 (tennis court) is repeated 18 and 20 times respectively, from 6 am to 9 pm, and from July 26 to August 10 in 2022. Path 3 (parking lot) is repeated 17 times from 7 am to 10 pm in mid-August in 2022. The feature inliers for each repeat is shown in Figure 3.15. For every repeat, we obtained sufficient feature inliers for localization. In general, we get the highest number of inliers in the middle of the day or when the weather is cloudy, since there are no shadows on the ground. The number of inliers decreases at dusk or dawn, and is at the lowest during the night. In addition, we test on paths that are not within the training data to evaluate the generalizability of the network. Although part of path 1 and path 3 are outside of the training data, we still obtained sufficient number of feature inliers, which shows that the learned features can generalize well to unseen regions. We have included the sample views of the VT&R live demo on two selected paths in Figure 3.16.

## 3.5 Conclusion

In this chapter, we have shown that we can generate image correspondences for image-pair sampling and perform self-supervised feature learning without any ground-truth pose information. We validated the effectiveness of learned features in the VT&R framework on unseen paths under various lighting conditions. In addition, our pipeline can be readily deployed on self-collected image sequences without additional ground truth data. Limitations of this work are the assumptions that we made to generate image correspondences. In the future, we intend to further relax these constraints and match sequences of image data that might not start or end at the same place. In addition, we plan to run closed-loop experiments over a longer period of time to test the robustness of the learned features against seasonal change.

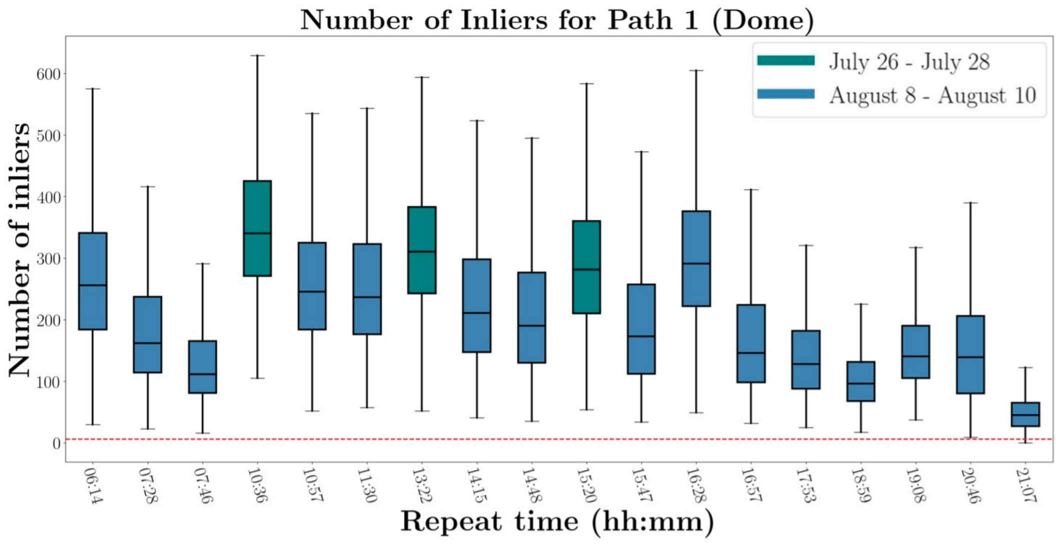


(a) Plot of feature inliers for path 3 (parking lot).

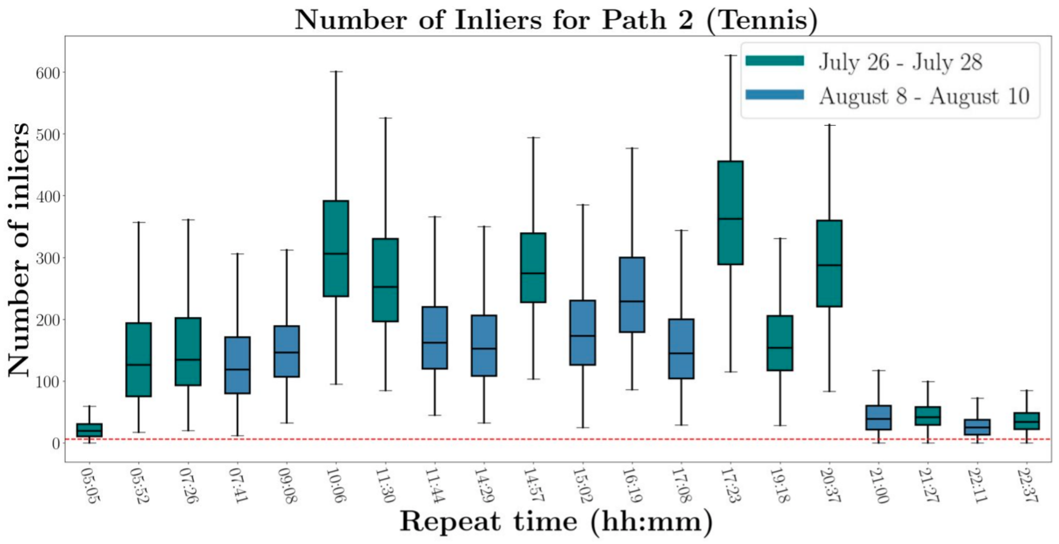


(b) Plot of feature inliers for path 4 (field).

Figure 3.13: **The mean number of inliers for each repeat of the offline experiments** using pre-collected rosbag files for paths 3 (a) and path 4 (b), where '*sup/selfsup*' indicates if the model is trained using supervised or self-supervised losses, '*gt/seq*' indicates if the image pairs are generated using ground-truth poses or SeqSLAM-based algorithm, and '*IMGNET/UTIAS*' represents the datasets that the model is pretrained or finetuned on. A missing entry indicates failure in localization. The learned features outperform SURF in all experiments. Under the exact same settings (i.e., trained on UTIAS datasets only), the self-supervised method achieves competitive results in comparison with the supervised method in all experiments. In addition, the performance increases for the self-supervised method when pretrained on ImageNet and finetuned on UTIAS datasets.

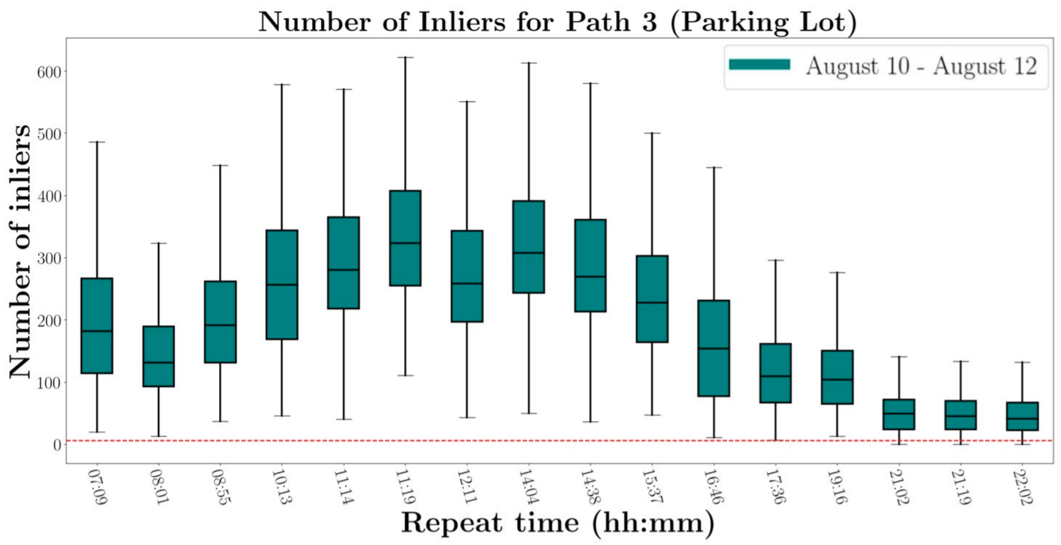


(a) Plot of feature inliers for path 1 (tennis court).



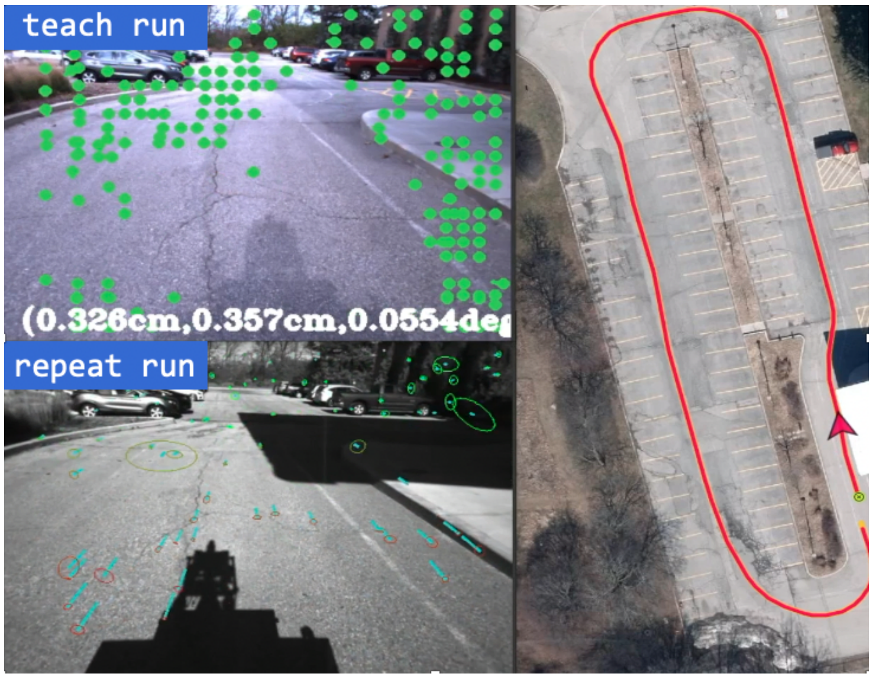
(b) Plot of feature inliers for path 2 (dome).

Figure 3.14: **The mean number of inliers for each repeat of the closed loop experiments** for path 1 (a), path 2 (b), and path 3 (c).

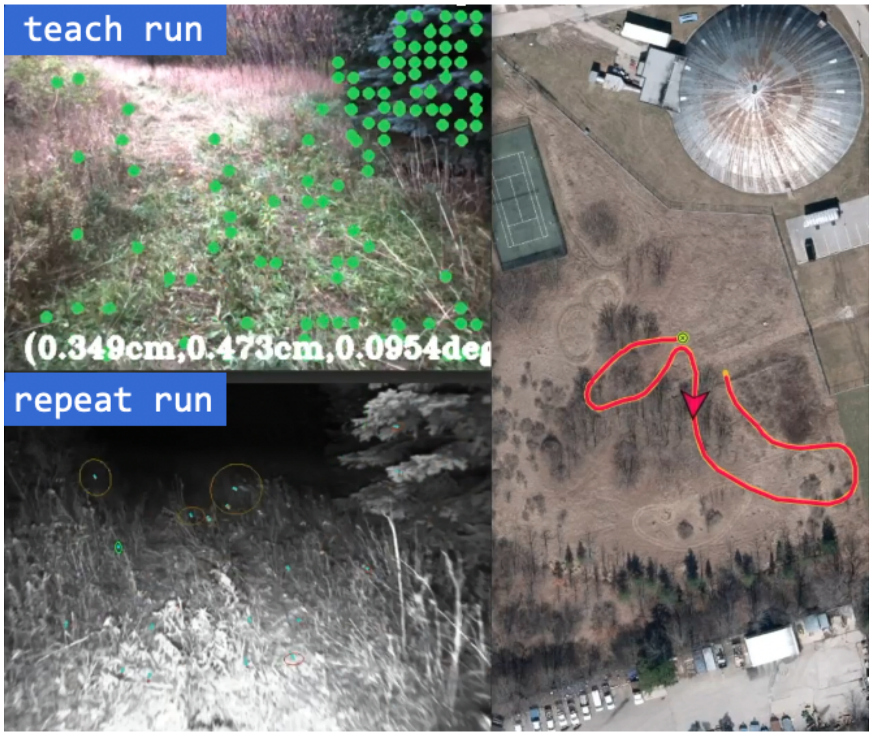


(a) Plot of feature inliers for path 3 (parking lot).

Figure 3.15: **The mean number of inliers for each repeat of the closed loop experiments** for path 1 (a), path 2 (b), and path 3 (c).



(a) **VT&R live demo on path 3 (parking lot)**



(b) **VT&R live demo on path 4 (field)**

Figure 3.16: In VT&R, the user teaches a path by driving the robot manually as shown on the right side, where the taught path is repeated autonomously. We visualize the images from the taught path in the top left corner, and the images from the live repeat run in the bottom left corner. The inliers of the learned features are directly overlaid on top of the teach images.

# Chapter 4

# Image Transformation using Neural Style Transfer

## 4.1 Introduction

Previously, we have discussed using deep learning to improve sparse visual features for long-term localization. While many existing approaches have attempted to improve the feature detectors and descriptors, other recent works [42–46] proposed to modify the input images using a deep image transformation to reduce the appearance gap prior to image matching. Inspired by the recent neural style-transfer techniques, the goal of [45, 46] is to use a style-transfer network that transforms the query images to resemble reference images to improve the performance of day-night feature matching results for long-term localization tasks. However, these methods optimize the transform network from scratch for each considered day and night image pair, which is not feasible for real-time deployment.

In this chapter, we attempt to answer the following question: *Is it better to learn features, image transformations, or both?* We propose an end-to-end differentiable pipeline that incorporates an image transformation network and a feature learning

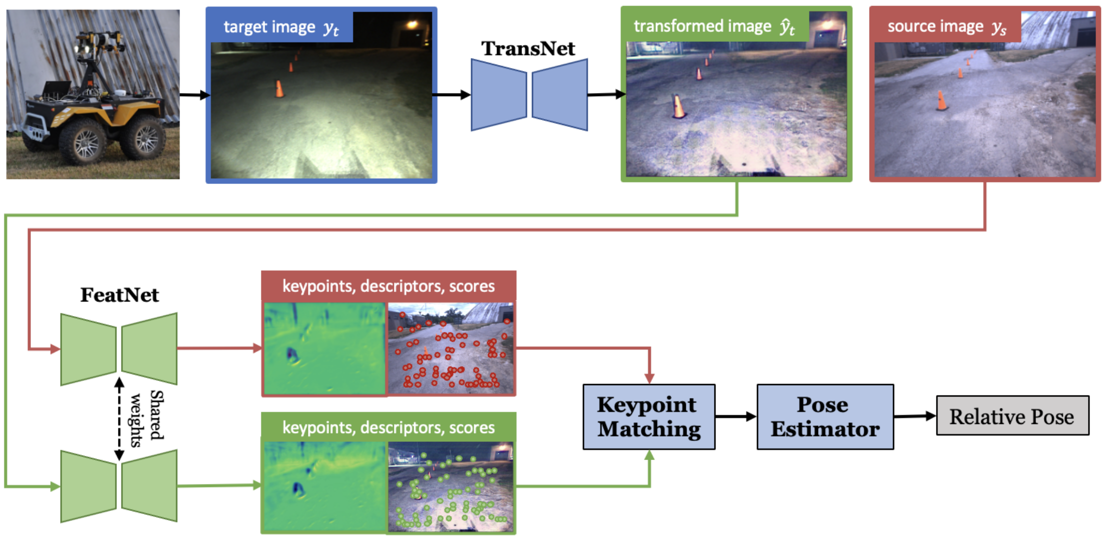


Figure 4.1: **High-level Overview**. TransNet transforms the target night images to day-like conditions. FeatNet predicts keypoints, descriptors, and scores on the source image and transformed target image that can be used in classical pose estimation to localize a robot despite large appearance change.

network as shown in Figure 4.1. The image transformation network transforms night-time images to resemble the style of daytime images, and the transformed images are used for feature learning. We compare our proposed method with using only the learned features or image transformation alone, and show that the combination of both components outperforms each individual part.

The content of this work is submitted to IROS 2023 [8].

## 4.2 Related Work

We have discussed many existing works in Chapter 3 that focus on directly learning poses or keypoints from input images. Instead of improving the invariance of feature detectors and descriptors under significant appearance changes, another direction is to modify the input images prior to feature extraction to reduce the appearance gap. Recent work in [43] takes inspiration from physics-based colour constancy models and learns a nonlinear transformation from RGB to grayscale colourspace. Other

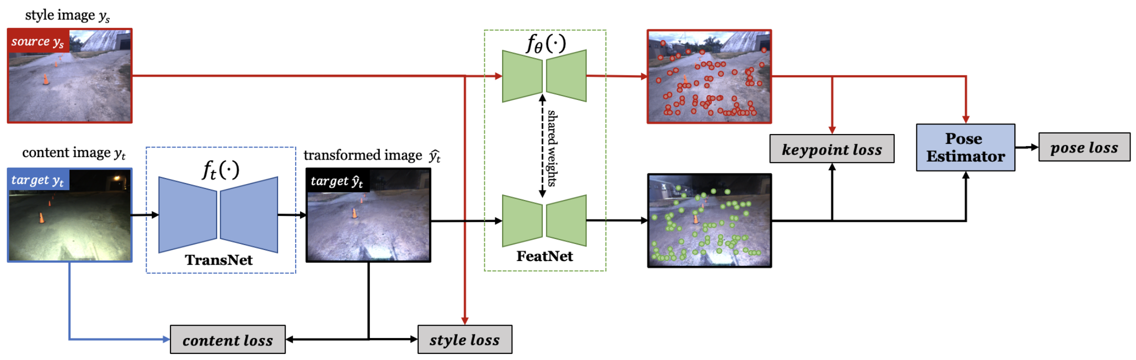


Figure 4.2: **Detailed Overall Pipeline**. Given a pair of day and night images to be matched, the objective of the TransNet is to transform the night images to a day-like condition by optimizing the standard content and style loss using a fixed loss network pretrained for image classification. FeatNet takes the original style images (i.e., source images) and the transformed images (i.e., target images) as input, then outputs the keypoints location, scores, and associated descriptors for each image. The keypoints are matched in a differentiable way to generate relative poses.

approaches attempt to transform the challenging query image to resemble the reference images. Generative adversarial networks (GAN) can be useful for such image transformations. ToDayGAN [47] uses an unsupervised image-to-image transformation to improve the localization performance of night query images. Query images are transformed by a network to day-like conditions, where the network is trained on aligned day and night image pairs. The features are extracted from the transformed image to match against the reference image. Porav et al. [42] proposed to use a cycle-consistency GAN with the addition of a descriptor-specific loss to help generate images that are optimized for matching without requiring aligned image pairs during pre-training. However, GAN-based methods cannot perfectly preserve the texture information that is beneficial for local-feature extraction, hence leading to limited performance gain in day-night image matching.

Another direction is to utilize neural style transfer [48] to perform image-to-image transformation. However, the traditional content and style perceptual losses used for style transfer mainly aim to reconstruct visually pleasing results, which might not result in images that are explicitly optimized for local feature extraction. Local low-level statistics of these transformed images can be drastically different compared to natural images, making it challenging to find local feature matches. To address this, [45, 46] adopt style transfer and extend the traditional content and style loss with an explicit feature-matching loss to ensure the resulting image transformation improves the feature matching performance. However, these methods utilize a fixed pretrained feature-detection-and-description network that is not optimized during training. In addition, these methods optimize the image transform network from scratch for each considered day and night image pair, which is not feasible for potential real-time deployment.

## 4.3 Methodology

The proposed end-to-end differentiable pipeline consists of two components: image transformation and feature learning. The image transformation network (i.e., TransNet) transforms the night images to day-like condition. The feature-learning network (i.e., FeatNet) takes the transformed day-like images as input to compute the keypoint and pose losses. The overall architecture is shown in Figure 4.2.

### 4.3.1 Image Transformation

Given a pair of source and target images, the objective is to transform the target image taken under adverse conditions to be more similar to the source image in order to improve the matching performance. In this work, we specifically focus on matching night-to-day image pairs by transforming night images to day-like conditions. We adopt style transfer to solve this matching problem, where the content image corresponds to the target image (e.g., night), and the style image corresponds to the source image (e.g., day). The image transformation component consists of two networks: an image transformation network and a fixed loss network.

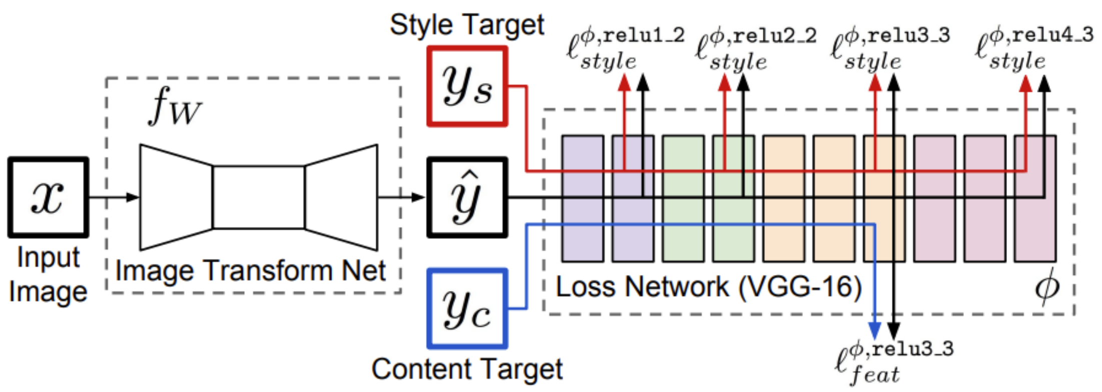

Style Target
$y_s$
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
$f_W$
$x$
$\hat{y}$
Input Image
Image Transform Net
$y_c$
Loss Network (VGG-16)
$\phi$
Content Target
$\ell_{feat}^{\phi,relu3\_3}$

Figure 4.3: **Content and Style Loss Overview.** We train an image transformation network to transform input images. We use a fixed loss network pretrained for image classification to define content loss and style loss, which measure perceptual differences in content and style between images respectively. The loss network remains fixed during the training process.

We denote our image transformation network to be $f_t(\cdot)$, which is composed of one encoder, five residual blocks, and one decoder similar to [45,46,48]. The residual block is able to retain the original content information and learn additional illumination-style information of day images with shortcut connections.

The style images are taken during daytime, denoted as $y_s$, whereas the content images, $y_t$, are taken during nighttime. The source-and-target image pair is represented by $(y_s, y_t)$. Our goal is to transform $y_t$ using $f_t(\cdot)$ to reconstruct a new image $\hat{y}_t = f_t(y_t)$, which resembles the day-like appearance style of $y_s$.

Similar to the perceptual losses defined in [48, 49], we make use of a fixed loss network, $\phi$, to define two perceptual losses that measure the differences in content and style between images. The fixed loss network is a 16-layer VGG network [40] pretrained on the ImageNet dataset [41] as shown in Figure 4.3, where $\phi_j(y)$ denotes the $H_j \times W_j \times C_j$ feature map at layer $j$, for the input image, $y \in \mathbb{R}^{H \times W \times 3}$. For image transformation, the content image, $y_t$, is the input image and the output image, $\hat{y}_t$, should combine the content of the target image, $y_t$, with the style of the source image, $y_s$.

### Content Loss

The objective of the content loss is to enforce the similarity between the original content image, $y_t$, and the transformed image, $\hat{y}_t$, on a higher conceptual level rather than on a per pixel basis. The content loss is the squared normalized Euclidean distance between feature representations defined as

$$\mathcal{L}_{\text{content}}(\hat{y}_t, y_t) = \frac{1}{H_j W_j C_j} \left\| \phi_j(\hat{y}_t) - \phi_j(y_t) \right\|_2^2, \tag{4.1}$$

where $j$ refers to the output from layer $relu3\_3$ of the VGG-16 loss network, $\phi$. Finding an image $\hat{y}_t$ that minimizes the content loss for early layers tends to produce images that are visually indistinguishable from $y_t$. As we reconstruct from higher layers, image feature and overall spatial structure are preserved but color, texture, and exact shape are not. Using a content loss for training our image transformation networks encourages the output image $\hat{y}_t$ to be perceptually similar to the target image $y_t$, but does not force them to match exactly.

### Style Loss

The style representation of an image can be captured with the Gram matrix of CNN features [49]. The Gram matrix at layer $j$ can be defined as

$$G_j(x) = \phi'_j(x)^T \phi'_j(x) \in \mathbb{R}^{C_j \times C_j}, \tag{4.2}$$

where $\phi'_j(y) \in \mathbb{R}^{H_j W_j \times C_j}$ is the 2D matrix representation of the $j$th feature map obtained with a reshaping operation from $\phi_j(y)$.

Then, the style loss between the transformed image, $\hat{y}_t$, and the style image, $y_s$, is defined as

$$\mathcal{L}_{\text{style}}(\hat{y}_t, y_s) = \sum_{j \in S} \left\| \frac{G_j(\hat{x}) - G_j(y_s)}{H_j W_j C_j} \right\|_2^2, \tag{4.3}$$

where $S = \{relu1\_2, relu2\_2, relu3\_3, relu4\_3\}$ is the considered set of VGG-16 layers. Generating an image $\hat{y}_t$ that minimizes the style loss preserves stylistic features from the target image, but does not preserve its spatial structure. Reconstructing from higher layers transfers larger-scale structure from the target image.

## 4.3.2 Feature Learning

Similar to Chapter 3, our feature-learning network, $f_\theta(\cdot)$, is adapted from the architecture presented by [1, 16, 50], which is a U-Net style convolutional encoder-multi-decoder architecture to output keypoints, descriptors, and scores based on image inputs. The encoder is a VGG16 network [40] pretrained on the ImageNet dataset [41], truncated after the $conv\_5\_3$ layer.

The keypoint-location decoder predicts the sub-pixel locations of each sparse 2D keypoint, $\mathbf{q} = [u_l, v_l]^T$, in the left stereo image. To achieve this, we equally divide the image into $16 \times 16$ square cells, with each generating a single candidate keypoint. We then apply a spatial softmax on each cell and take a weighted average of the pixel coordinates to return the sub-pixel keypoint locations.

In addition, the network computes the scores by applying a spatial softmax function to the output of the second decoder branch. The scores, $s$, are mapped to $[0, 1]$, which predicts how useful a keypoint is for pose estimation. Finally, we generate dense descriptors for each pixel by resizing and concatenating the output of each encoder layer, which results in a descriptor vector, $\mathbf{d} \in \mathbb{R}^{960}$.

### Pose Estimation

After generating $N$ keypoint predictions for the source image, we need to perform data association between the keypoints in the source and target images by performing a dense search for optimum keypoint locations in the target image. For each keypoint in the source image, we compute a matched point in the target image by taking the weighted sum of all image coordinates in the target image as Equation 3.7. Then, we find the descriptor, $\hat{\mathbf{d}}_t^i$, and score, $\hat{s}_t^i$, for each computed target keypoint using bilinear interpolation.

Given the corresponding 2D keypoints $\{\mathbf{q}_s^i, \hat{\mathbf{q}}_t^i\}$ from the source and target images, we use Equation 3.5 to compute their 3D coordinates, $\{\mathbf{p}_s^i, \hat{\mathbf{p}}_t^i\}$. In addition, their descriptors and scores are $\{\mathbf{d}_s^i, \hat{\mathbf{d}}_t^i\}$ and $\{s_s^i, \hat{s}_t^i\}$.

We then use the ground-truth pose information to reject outliers, resulting in features that are geometrically consistent with each other.

Given the inlier 3D keypoints, we can estimate the relative pose from the source to the target by minimizing the following cost using Singular Value Decomposition (SVD) as described in Chapter 2.2:

$$J = \sum_{i=1}^{N} w^i \left\| (\mathbf{C}_{ts}\mathbf{p}_s^i + \mathbf{r}_t^{st}) - \hat{\mathbf{p}}_t^i \right\|_2^2, \tag{4.4}$$

where the weight for a matched point pair is a combination of the learned point scores and how well the descriptors match:

$$w^i = \frac{1}{2}(f_{\text{zncc}}(\mathbf{d}_s^i, \hat{\mathbf{d}}_t^i) + 1)s_s^i \hat{s}_t^i. \tag{4.5}$$

Finally, we obtain the estimated relative transform between the source and target images as

$$\hat{\mathbf{T}}_{ts} = \begin{bmatrix} \hat{\mathbf{C}}_{ts} & \hat{\mathbf{r}}_t^{st} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{4.6}$$

### Keypoint Loss

Similar to Chapter 3, we transform the predicted source keypoints, $\mathbf{p}_s^i$, to the target frame using the ground-truth relative transform, $\mathbf{T}_{ts}$, and define a keypoint loss as follows:

$$\mathcal{L}_{\text{keypoint}} = \sum_{i=1}^{N} ||\mathbf{T}_{ts}\mathbf{p}_s^i - \hat{\mathbf{p}}_t^i||_2^2. \tag{4.7}$$

**Pose Loss**

Unlike Chapter 3, we compoute a pose loss given the ground-truth pose and the estimated pose, which is defined as a weighted sum of the translational and rotational errors:

$$\mathcal{L}_{\text{pose}} = \left\|\mathbf{r}_t^{st} - \hat{\mathbf{r}}_t^{st}\right\|_2^2 + \lambda \left\|\mathbf{C}_{ts}\hat{\mathbf{C}}_{ts}^T - \mathbf{1}\right\|_2^2, \tag{4.8}$$

where $\lambda$ is used to balance the rotational and translational errors.

The total loss is the weighted sum of all four losses (i.e., content loss, style loss, pose loss, and keypoint loss), which is defined as:

$$\mathcal{L} = \lambda_1\mathcal{L}_{\text{style}} + \lambda_2\mathcal{L}_{\text{content}} + \lambda_3\mathcal{L}_{\text{pose}} + \lambda_4\mathcal{L}_{\text{keypoint}}, \tag{4.9}$$

where the weights, $\lambda_i$, are determined empirically to balance the influence of the four loss terms.

## 4.4 Experiments

We conducted various experiments on real-world long-term vision datasets to validate and compare the effectiveness of feature learning and style transfer with selected baselines on localization tasks. We organize the experiments into the following three main categories for comprehensive analysis. In Section 4.4.3, we conducted four experiments on different combinations of FeatNet and TransNet with different training schemes to reveal the importance of our proposed joint training approach. In Section 4.4.4, we compare the learned features against the classical SURF [4] features to evaluate the effectiveness of our proposed approach in overcoming the limitations

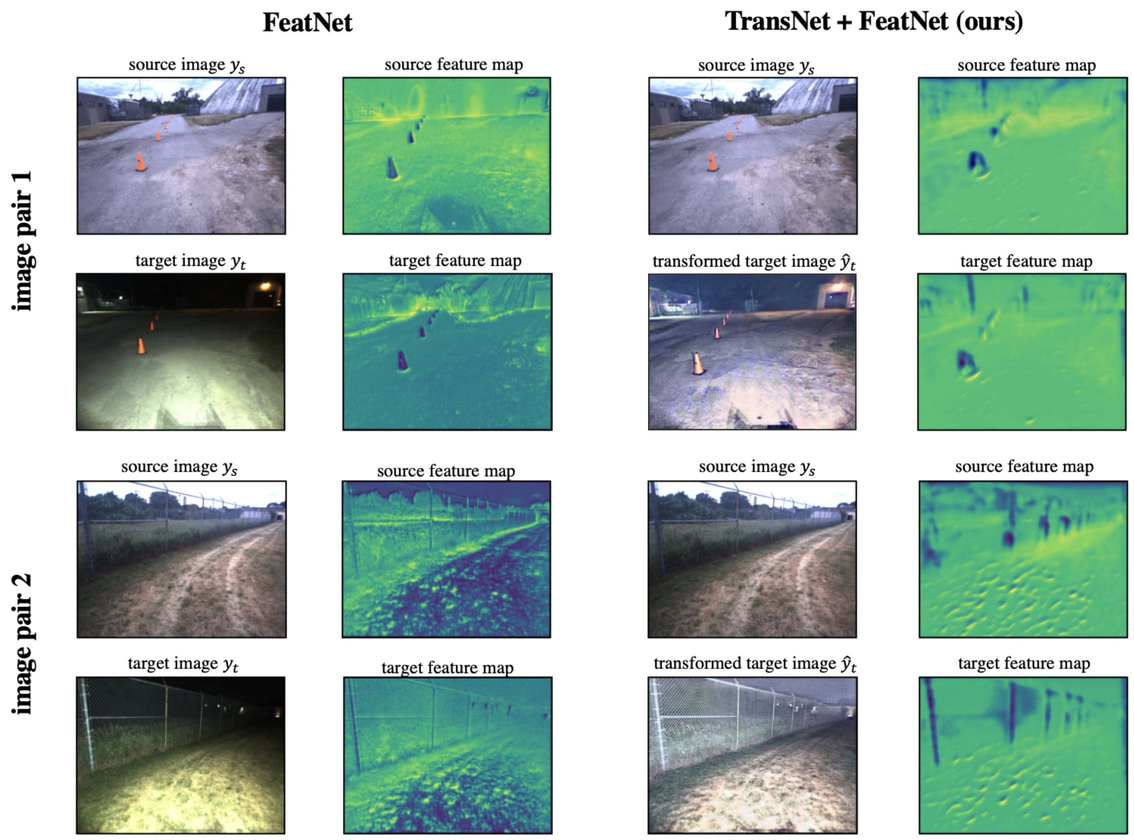


Figure 4.4: **Qualitative Results.** Visualization of the original source and target image pairs, the transformed target images and their feature maps. The feature maps are the dense feature detector values output from the FeatNet decoder, prior to the spatial softmax layer.

of traditional hand-crafted features. In Section 4.4.5, we compare TransNet with the colourspace transformation proposed in [43]. It aims to show the benefits of image transformation network brought to feature learning over the existing works, specifically in handling appearance changed caused by illumination variations.

## 4.4.1 Datasets

### UTIAS Dataset

For training, we use the public dataset, UTIAS-In-the-Dark, as described in Chapter 2.4. The training data was collected using a Clearpath Grizzly robot with a Bumblebee XB3 camera. The robot autonomously repeats a path across drastic lighting changes using Multi-Experience VT&R [6]. VT&R stores stereo image keyframes as vertices in a spatio-temporal pose graph, where each edge contains the relative pose between a repeat vertex and the teach vertex. The dataset contains 39 runs of a path collected at the campus of UTIAS in summer 2016. The same path was repeated hourly for 30 hours, which captures incremental lighting changes throughout the day and night. The path for the In-The-Dark datasets can be visualized in Figure 3.10. To test for day-night image matching, we select one run that is collected during daytime as the reference sequence, and 9 runs in nighttime as the query sequences. We sample 20,000 image pairs for training, and 4,000 non-overlapping image pairs for testing. Each image pair consists of a daytime image as the source image and a nighttime image as the target image.

#### Oxford RobotCar

We also train our network on another publicly available dataset, Oxford RobotCar [51]. However, it is not trivial to generate training image pairs from the Oxford RobotCar dataset because the GPS-based ground-truth is very inaccurate. We adopt the following approach to find the relative poses between images from different sequences. For pairs of images from two different sequences, we accumulate the point cloud captured by the 3D lidar for 20 meters using the visual odometry result provided by the Oxford dataset. The resulting two point clouds are aligned with the global registration followed by ICP alignment using the implementation of Open3D [52].

### 4.4.2 Training and Inference

We use the sampled image pairs to train the pipeline. For each input image pair, we use the daytime image as the style image to train the TransNet. For feature learning, we discard outliers based on keypoint error using the ground-truth pose during training and using RANSAC during inference. The FeatNet encoder is a

VGG16 network [40] pretrained on the ImageNet dataset [41], truncated after the *conv_5_3* layer. Both networks are trained end-to-end using the Adam optimizer with a learning rate of $10^{-5}$ on an NVIDIA Tesla V100 DGXS GPU for 100 epochs. For our experiments, we set $\lambda_1 = 10^{-5}$, $\lambda_2 = 10^{-5}$, $\lambda_3 = 10.0$, $\lambda_4 = 2.0$.

## 4.4.3 Comparison Between TransNet and FeatNet

In this section, we evaluate different combinations of TransNet and FeatNet for metric localization under adverse conditions. Specifically, we compare the performance of ***FeatNet***, ***TransNet***, ***TransNet + FeatNet***, and ***TransNet ← FeatNet***. *FeatNet* is the same approach as mentioned in [16], and is trained on the original day-night image pairs without performing style transfer. *TransNet* is trained with only the perceptual losses proposed in [49], and is tested on image pairs after integrating with a pretrained feature network to compute the localization errors. *TransNet + FeatNet* is our proposed method, where both the TransNet and FeatNet are trained end-to-end on the feature losses and perceptual losses. Lastly, *TransNet ← FeatNet* is similar to the previous method, but follows a two-stage training scheme. During the first stage, FeatNet is trained without image transformation. During the second stage, TransNet is trained with the FeatNet from the first stage, where the FeatNet stays fixed during the second-stage training.

For evaluation, we compute the path-following errors between the image pairs sampled from the UTIAS-In-the-Dark test set or from the Oxford RobotCar test set, then report the longitudinal errors, lateral errors and yaw angle errors in Table 4.1 and Table 4.2, respectively.

When trained on the perceptual losses only, the transformed images from *TransNet* are not optimized for keypoint matching, which is detrimental for visual metric localization as shown in Table 4.1 and in Table 4.2: the performance is significantly worse than when not doing style transfer at all (*FeatNet*). When incorporating both of TransNet and FeatNet during training, TransNet is optimized explicitly for feature extraction, which leads to lower localization errors. However, we notice that the order of training both networks has little impact on the performance. Jointly training both networks (i.e., *TransNet + FeatNet*) achieves comparable results as training FeatNet and TransNet sequentially (i.e., *TransNet* $\leftarrow$ *FeatNet*).

We visualize the transformed target images in Figure 4.4, where the nighttime images are transformed into day-like conditions, making it easier for image matching between the day-night image pairs. In addition, we visualize the feature maps output from the FeatNet decoder prior to the spatial softmax layer, which can be interpreted as the detector response map for FeatNet. Adding image transformation results in more similar feature maps compared to using the FeatNet only, which improves the feature matching quality.

### 4.4.4 Comparison Between FeatNet and SURF

In this section, we compare the effectiveness of using learned features against using classical SURF features [4]. Inspired by [42], we also integrate a differentiable SURF feature detector and descriptor pipeline with the TransNet. In addition to the style and content loss, we compute the L1 loss between SURF detector response maps and dense descriptor response maps on the transformed image pairs as described in [42].

From Table 4.1, we can see that using an image transformation network improves the localization results from directly extracting SURF features on the original image pairs. However, using learned features significantly outperforms SURF features.

### 4.4.5 Comparison Between TransNet and Colourspace Transformation

We compare the TransNet with other existing colourspace transformation networks. Clement et al. [43] proposed to find an optimal colourspace transformation that performs nonlinear mapping from RGB to grayscale colourspaces to maximize the number of feature matches for image pairs. We compared our approach to two different formulations of colourspace transformations reported in [43]. Following the same naming convention, we refer to the generalized colour-constancy model as *SumLog* and *SumLog-E*, and refer to the MLP-based model as *MLP-E*.

Since the detectors and matchers used in [43] rely on non-differentiable components, a differentiable proxy network was used to predict the number of feature matches. For fair comparison, we replace their proxy matcher network by our pretrained FeatNet, which directly computes the pose and keypoint losses, to train the colourspace transformation network.

From Table 4.1, we can see that the *MLP-E* transformation outperforms the *SumLog* and *SumLog-E* transformations. However, using neural style transfer achieves consistently better results.

## 4.5 Conclusion

In conclusion, we proposed a fully differentiable pipeline that consists of image transformation and feature learning, which improves the long-term metric localization performance under adverse appearance change. We performed various ablation studies to show the effectiveness of combining image transformation and feature learning in metric localization on real-world datasets. We discovered that the combination of neural style transfer and feature learning leads to the best results, which outperforms each individual component. Adopting neural style transfer prior to feature

Table 4.1: **Comparison of the path-following errors and average number of inliers for UTIAS In-the-Dark**. *FeatNet* is pretrained without the image transformation component, whereas the proposed methods *TransNet + FeatNet* are trained jointly and *TransNet* ← *FeatNet* trains the FeatNet and TransNet sequentially. *TransNet* and *SumLog* are integrated with a pretrained FeatNet during inference time to compute the localization errors. We report the longitudinal errors $\Delta x$ in meters, lateral errors $\Delta y$ in meters, the yaw angle errors $\Delta\theta$ in degrees, and the average number of feature inliers.

| | $\Delta x(m)$ | $\Delta y(m)$ | $\Delta\theta(^\circ)$ | inliers |
|---|---|---|---|---|
| TransNet [49] | 1.52 | 1.38 | 2.9 | 364 |
| FeatNet [16] | 0.07 | 0.05 | 0.47 | 484 |
| **TransNet ←FeatNet** (ours) | **0.019** | **0.014** | 0.25 | 504 |
| **TransNet + FeatNet** (ours) | 0.024 | **0.014** | **0.24** | **505** |
| SURF [4] | 1.24 | 1.52 | 2.84 | 40 |
| TransNet+SURF [42] | 1.09 | 1.20 | 1.82 | 59 |
| SumLog [43] | 0.14 | 0.16 | 1.67 | 463 |
| SumLog-E$^\dagger$ [43] | 0.12 | 0.26 | 1.02 | 474 |
| MLP-E$^\dagger$ [43] | 0.08 | 0.12 | 0.57 | 490 |

matching generates higher-quality matches, which subsequently leads to lower localization errors. For future extensions, we plan to integrate the proposed method into an existing VT&R framework to test for real-time closed-loop deployment in unseen environments.

Table 4.2: **Comparison of the path-following errors and average number of inliers for Oxford RobotCar**. We report the longitudinal errors $\Delta x$ in meters, lateral errors $\Delta y$ in meters, the yaw angle errors $\Delta \theta$ in degrees, and the average number of feature inliers.

| | $\Delta x(m)$ | $\Delta y(m)$ | $\Delta \theta(^\circ)$ | inliers |
|---|---|---|---|---|
| TransNet [49] | 2.59 | 1.49 | 4.9 | 112 |
| FeatNet [16] | 0.28 | 0.32 | 2.7 | 215 |
| **TransNet ←FeatNet** (ours) | **0.19** | **0.14** | 1.44 | **234** |
| **TransNet + FeatNet** (ours) | 0.24 | 0.15 | **1.24** | 229 |

# Chapter 5

# Conclusion

## 5.1 Summary and Contributions

In this thesis, we have discussed and compared different deep-learning-based approaches to improve the accuracy and robustness of visual localization under drastic appearance change.

Chapter 3 presented a novel self-supervised feature learning pipeline, where we can generate image correspondences using place recognition for image-pair sampling and perform self-supervised feature learning without any ground-truth pose information. We validated our methods on the publicly available datasets and showed improved performance when compared to other existing feature extraction methods. In addition, we demonstrated successful closed-loop real-time localization performance when deploying the learned features in the VT&R framework on unseen paths under various lighting conditions.

Chapter 4 presented a novel end-to-end differentiable pipeline that incorporates an image transformation network and a feature learning network that can be jointly optimized. We examined the effectiveness of combining style transfer and feature learning on real-world datasets, and showed that adopting neural style transfer prior to feature matching generates higher-quality matches, which subsequently leads to lower localization errors and more robust localization performance under adverse appearance change.

## 5.2 Future Work

We have demonstrated that we can use learned features and deep learned image transformation for more robust localization performance. However, there are improvements that can be made to our methods. We discuss some of the possible directions for future work in this section.

In Chapter 3, we make assumptions when generating image correspondences using SeqSLAM that all sequences follow the same trajectory that starts and ends at the same location. However, more work can be done to relax the constraints by implementing a more flexible sequence alignment on sequences that only have partial overlaps.

In addition, more work can still be done to improve the viewpoint invariance of the learned features. Currently, we train on data that is collected using Multi-experience localization, which repeats the same trajectory with high-level of accuracy and small offsets. Thus, the sampled training image pairs have very little yaw and lateral viewpoint changes, which is not beneficial for learning viewpoint invariant features in the training phase. To address this, we can explore multiple options to improve the training data. For instance, we can apply random disturbances to the controller of the robot when collecting data using MEL during repeat runs to increase the offset relative to the taught path, thus introducing larger viewpoint changes in the training data. Alternatively, we can collect our own dataset by manually driving the robot so it roughly follows the trajectories without using VT&R, then use the place recognition algorithm described in Chapter 3 to generate training data.

When training the network that produces visual features, we supply one pair of images at a time. As a result, the extracted keypoint locations are not temporally consistent, and the pose estimates computed from the features are less smooth and may contain outliers. However, we can further utilize the prior information from the feature extraction results from the previous frames. For instance, we can take a window of frames as input, and perform feature tracking in addition to feature extraction, which learn features that are temporally consistent across a sequence of frames.

In Chapter 4, the trained neural style transfer deals with illumination changes only. In the future, we can investigate the impact of our method on feature matching across seasonal appearance change.

In addition, the current model can only transform images to one target domain (i.e., day-time condition), and a different model is required to train for a new target domain. Ideally, we only need to learn a single generator model to transform input images to different target domains.

Finally, our networks are trained offline on pre-collected datasets, then deployed on the robot for real-time application. The learning procedure can be further improved for learning-based vision systems to be suitable for the continuous operation of mobile robots. We can adapt to an online learning scheme, where we incrementally improve the networks on-the-fly by training on the new incoming data in sequential order.

In summary, this thesis presented several novel approaches to improve upon vision-based localization pipelines using deep-learning techniques. However, future advances are expected to enable reliable, robust, and data-efficient long-term visual localization.

# Bibliography

[1] D. Barnes and I. Posner, "Under the radar: Learning to predict robust keypoints for odometry estimation and metric localisation in radar," *CoRR*, vol. abs/2001.10789, 2020.

[2] P. Furgale and T. Barfoot, "Visual teach and repeat for long-range rover autonomy," *J. Field Robotics*, vol. 27, pp. 534–560, 09 2010.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," vol. 3951, 07 2006, pp. 404–417.

[5] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," 11 2011, pp. 2564–2571.

[6] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *2016 IEEE/RSJ IROS*, 2016, pp. 1918–1925.

[7] Y. Chen and T. D. Barfoot, "Self-supervised feature learning for long-term metric visual localization," 2022.

[8] F. D. Yuxuan Chen, Binbin Xu and T. D. Barfoot, "What to learn: Features, image transformations, or both?" 2023, submitted for IROS 2023.

[9] T. D. Barfoot, *State Estimation for Robotics.* Cambridge University Press, 2017.

[10] M. Fischler and R. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[11] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design.* Nob Hill Pub., 2009.

[12] V. Balntas, S. Li, and V. Prisacariu, “Relocnet: Continuous metric learning relocalisation using neural nets,” in *The European Conference on Computer Vision (ECCV)*, September 2018.

[13] M. Ding, Z. Wang, J. Sun, J. Shi, and P. Luo, “Camnet: Coarse-to-fine retrieval for camera re-localization,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2871–2880.

[14] Z. Laskar, I. Melekhov, S. Kalia, and J. Kannala, “Camera relocalization by computing pairwise relative poses using convolutional neural network,” 2017.

[15] A. Kendall, M. Grimes, and R. Cipolla, “Convolutional networks for real-time 6-dof camera relocalization,” *CoRR*, vol. abs/1505.07427, 2015.

[16] M. Gridseth and T. D. Barfoot, “Keeping an eye on things: Deep learned features for long-term visual localization,” 2021.

[17] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, “D2-net: A trainable CNN for joint detection and description of local features,” *CoRR*, vol. abs/1905.03561, 2019.

[18] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, “TILDE: A temporally invariant learned detector,” *CoRR*, vol. abs/1411.4568, 2014.

[19] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," *CoRR*, vol. abs/1712.07629, 2017.

[20] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: learned invariant feature transform," *CoRR*, vol. abs/1603.09114, 2016.

[21] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "Lf-net: Learning local features from images," *CoRR*, vol. abs/1805.09662, 2018.

[22] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers, "Gn-net: The gauss-newton loss for deep direct SLAM," *CoRR*, vol. abs/1904.11932, 2019.

[23] J. Revaud, P. Weinzaepfel, C. R. de Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger, "R2D2: repeatable and reliable detector and descriptor," *CoRR*, vol. abs/1906.06195, 2019.

[24] L. Sun, M. Taher, C. Wild, C. Zhao, Y. Zhang, F. Majer, Z. Yan, T. Krajník, T. Prescott, and T. Duckett, "Robust and long-term monocular teach and repeat navigation using a single-experience map," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2635–2642.

[25] M. Kasper, F. Nobre, C. Heckman, and N. Keivan, "Unsupervised metric relocalization using transform consistency loss," *CoRR*, vol. abs/2011.00608, 2020.

[26] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *2012 IEEE ICRA*, 2012, pp. 1643–1649.

[27] C. Valgren and A. J. Lilienthal, "Sift, surf & seasons: Appearance-based long-term localization in outdoor environments," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 149–156, 2010, selected papers from the 2007 European Conference on Mobile Robots (ECMR '07).

[28] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *International Journal of Robotics Research*, vol. 32, pp. 1645–1661, 12 2013.

[29] C. Linegar, W. Churchill, and P. Newman, "Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation," in *2015 IEEE ICRA*, 2015, pp. 90–97.

[30] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixé, "Understanding the limitations of cnn-based absolute camera pose regression," *CoRR*, vol. abs/1903.07504, 2019.

[31] S. Lowry, N. Sünderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, pp. 1–19, 11 2015.

[32] M. Cummins and P. Newman, "Highly scalable appearance-only slam - fab-map 2.0," 06 2009.

[33] P. Neubert, N. Sünderhauf, and P. Protzel, "Appearance change prediction for long-term navigation across seasons," in *2013 European Conference on Mobile Robots*, 2013, pp. 198–203.

[34] M. Milford, "Vision-based place recognition: How low can you go?" *The International Journal of Robotics Research*, vol. 32, pp. 766–789, 06 2013.

[35] P. Hansen and B. Browning, "Visual place recognition using hmm sequence matching," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4549–4555.

[36] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Robust visual robot localization across seasons using network flows," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, Jun. 2014.

[37] O. Vysotska and C. Stachniss, "Lazy data association for image sequences matching under substantial appearance changes," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 213–220, 2016.

[38] ——, "Effective visual place recognition using multi-sequence maps," *IEEE RAL*, vol. 4, no. 2, pp. 1730–1736, 2019.

[39] P. Neubert, S. Schubert, and P. Protzel, "Exploiting intra database similarities for selection of place recognition candidates in changing environments." 2015.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[42] H. Porav, W. P. Maddern, and P. Newman, "Adversarial training for adverse conditions: Robust metric localisation using appearance transfer," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1011–1018, 2018.

[43] L. Clement, M. Gridseth, J. Tomasi, and J. Kelly, "Learning matchable image transformations for long-term metric visual localization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1492–1499, apr 2020.

[44] B. Xu, A. Davison, and S. Leutenegger, "Deep probabilistic feature-metric tracking," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 223 – 230, 2021.

[45] A. Uzpak, A. Djelouah, and S. Schaub-Meyer, "Style transfer for keypoint matching under adverse conditions," ser. 2020 International Conference on 3D Vision (3DV), 2020, pp. 1089–1097.

[46] L. Shi, M. Wang, Y. Yue, and Y. Yang, "Absolute anchor-negative distance based metric learning for day-night feature matching," in *2021 IEEE International Conference on Unmanned Systems (ICUS)*, 2021, pp. 1045–1050.

[47] A. Anoosheh, T. Sattler, R. Timofte, M. Pollefeys, and L. Van Gool, "Night-to-day image translation for retrieval-based localization," 2018.

[48] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423.

[49] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016.

[50] Y. Chen and T. D. Barfoot, "Self-supervised feature learning for long-term metric visual localization," 2022.

[51] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[52] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.