

Optimizing PTZNet for Real-World Deployment

Batch-Level Mixing to Compensate for Data Scarcity in Pose and Overlap Regression

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Marco Zeisler, BSc

Matrikelnummer 01628030

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof.Dipl.-Ing.Mag.Dr. Margrit Gelautz

Mitwirkung: Dr. Maximilian Bachl

Dipl.Ing. Thomas Köppel, BSc

Wien, 30. November 2025

Marco Zeisler

Margrit Gelautz



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Optimizing PTZNet for Real-World Deployment

Batch-Level Mixing to Compensate for Data Scarcity in Pose and Overlap Regression

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Marco Zeisler, BSc

Registration Number 01628030

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof.Dipl.-Ing.Mag.Dr. Margrit Gelautz

Assistance: Dr. Maximilian Bachl

Dipl.Ing. Thomas Köppel, BSc

Vienna, November 30, 2025

Marco Zeisler

Margrit Gelautz



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Marco Zeisler, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 30. November 2025

Marco Zeisler



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

Coming from a background in Software Engineering and Internet Computing, Deep Learning was initially not on my radar. However, towards the end of my studies, I realized that this was my final chance to deep-dive into the subject. I wanted to use the freedom of this thesis to finally gain experience in this field, which today, driven by the AI boom, is a very prominent research area in the daily life of 2025.

For the opportunity to do so, I would like to thank Kapsch TrafficCom AG for providing resources and for the support of competent colleagues. My thanks go to my previous boss, Bernhard Kunz, as well as my current one, Edwin Frühwirth, for making it possible for me to write this thesis with a partial allocation of working hours.

I thank my two colleagues, Maximilian Bachl and Thomas Köppel, for their comprehensive supervision on the company side. Although both supported me throughout the entire thesis, my special thanks go to Max for his valuable ideas, feedback, and our brainstorming during the practical part. I thank Thomas for sharpening the theoretical part through many in-depth and high-quality reviews, as well as for constantly motivating me and keeping my spirits high. Thomas, you helped me conquer this rollercoaster.

Last but not least, I would like to thank Professor Gelautz, who made this thesis and cooperation possible in the first place. I have learned a lot under her supervision and deeply appreciate her kind and professional guidance. Especially during the stressful final phase, she reviewed the work in great detail and provided me with highly appreciated feedback.

On a personal level, I thank my family above all, especially Mum, Dad, and my grandparents. You stood by me not only financially during the early days of my studies but also with constant moral support. I also extend my thanks to all my other family members. Finally, I thank Mathias, who always encouraged me during the most exhausting phases, even when I may not have shown my best side, and who always made sure I stayed well-fed.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

In der Stadtplanung und im Verkehrsmanagement überwachen Pan, Tilt and Zoom (PTZ)-Kameras Straßen, Kreuzungen und öffentliche Bereiche. Die präzise Schätzung, wie sich eine einzelne PTZ-Kamera zwischen Bildern bewegt und in welchem Maße sich ihre Fields of View (FoVs) überlappen, ist anspruchsvoll - bedingt durch Szenenvariabilität, Linsenverzerrungen, dynamische Bewegungen sowie durch begrenzte oder fehlende Kameraparameter. Das Sammeln großer, diverser Echtwelt-Datensätze mit präzisen Metadaten ist sowohl kostenintensiv als auch logistisch komplex.

Diese Arbeit untersucht, ob unser siamesisches Deep-Learning-Modell, das PTZNet, in der Lage ist, relative PTZ-Rotationen und FoV-Überlappungen aus Bildpaaren infolge von Kamerabewegungen zuverlässig zu schätzen. Dabei wird hauptsächlich auf virtuell erstellte synthetische Bilder zurückgegriffen, während gleichzeitig die Überbrückung der Synthetic-To-Real (S2R)-Domänenlücke adressiert wird - also die Diskrepanz in der Merkmalsverteilung zwischen realen und synthetischen Daten.

PTZNet schätzt gleichzeitig die Kamerarotation und die FoV-Überlappung. Die Genauigkeit wird anhand des 3D-Rotationsfehlers, der sphärischen Intersection over Union (IoU) für die Überlappung sowie eines kombinierten relativen Fehlers zur gemeinsamen Genauigkeitsbewertung gemessen. Ein zentrales Element ist das Batch-Level Mixing (BLM)-Verfahren, das synthetische und begrenzt verfügbare Echtwelt-Bilder strategisch innerhalb von Trainingsbatches integriert, um die Generalisierbarkeit auf reale Szenen zu verbessern.

Unsere Experimente zeigen, dass die Mischung synthetischer Bilder mit einer kleinen Menge realer Daten deutlich geringere Rotations- und Überlappungsfehler erzielt als das Training auf nur einer der beiden Domänen. Dieser Vorteil bleibt über verschiedene Mengen realer Daten hinweg bestehen: Wenn die Erhebung von Echtwelt-Bildern kostspielig ist, kann ein primär auf synthetischen Bildern basierendes Training, ergänzt durch eine gezielte kleine Menge realer Daten, den Mangel an großer Mengen von Echtwelt-Daten effektiv ausgleichen und die Inferenz verbessern.

Die Forschungsbeiträge dieser Arbeit umfassen (1) die Analyse der S2R-Domänenlücke für synthetisch erzeugte PTZ-Daten und deren Einfluss auf die Inferenz, (2) die Entwicklung einer Batch-Level Mixing (BLM) Trainingsstrategie, die, verglichen mit einem Dataset-Level Mixing (DLM), robuster gegen Domänenverschiebungen ist, (3) den quantitativen

Nachweis, dass synthetische Bilder die Schätzung relativer PTZ-Rotationen und FoV-Überlappungen verbessern, wenn reale Daten knapp sind, sowie (4) die Einführung von PTZNet - einer siamesischen Architektur für die gemeinsame Regression von PTZ-Rotation und FoV-Überlappung, trainiert auf gemischten synthetischen und realen Daten.

Abstract

In urban planning and traffic management, Pan, Tilt and Zoom (PTZ) cameras monitor streets, intersections, and public areas. Precisely estimating how a single PTZ camera moves between images, and how much their Fields of View (FoVs) overlap, is challenging due to scene variability, lens distortion, dynamic motion, and limited or unavailable camera parameters. Collecting large, diverse real-world datasets with precise metadata is costly and logistically complex.

This thesis investigates whether our Siamese deep learning model, PTZNet, can infer relative PTZ rotations and FoV overlap with competitive performance caused by camera movement between image pairs, while relying primarily on virtually created synthetic images and mitigating the Synthetic-To-Real (S2R) domain gap, which is the discrepancy in feature distribution between real-world and synthetic data.

PTZNet jointly regresses camera rotation and FoV overlap. Performance is measured with a 3D rotation error, the spherical Intersection over Union (IoU) for overlap, and a combined relative error capturing joint accuracy. A key element is the Batch-Level Mixing (BLM) method, which strategically integrates synthetic and limited real-world images within training batches to improve generalization to real-world scenes.

Our experiments show that mixing synthetic images with a small amount of real-world data yields substantially lower errors in predicting rotation and overlap than training on either domain alone. This improvement persists across varying real-data amounts: When real-world image collection is costly, training primarily on synthetic images and tuning with a small targeted set of real-world samples effectively compensates for limited real-world data and improves inference.

The main contributions of this work include (1) an analysis of the S2R domain gap for virtually created PTZ data and its effect on inference, (2) the development of the Batch-Level Mixing (BLM) data training strategy that outperforms Dataset-Level Mixing (DLM) in robustness to domain shift, (3) quantitative evidence that synthetic images improve relative PTZ rotation and overlap prediction accuracy when real-world images are scarce, and (4) the introduction of PTZNet, a Siamese architecture for joint PTZ rotation and FoV-overlap regression trained on mix of synthetic and real-world data.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Motivation	2
1.2 Problem Statement	2
1.3 Methodology	4
1.4 Thesis Structure	5
2 Background and Key Concepts	7
2.1 Machine Learning: From Fundamentals to Deep Neural Networks	8
2.2 PTZ Camera Displacement: Mathematical Foundations	15
3 Related Work	27
3.1 Pose and Overlap Regression	27
3.2 Deep Domain Adaptation	34
3.3 Domain Adaptation Regression (DAR)	39
3.4 Synthetic-To-Real (S2R) Domain Adaptation	43
3.5 Summary and Research Gaps	48
3.6 Contributions	50
4 Method	53
4.1 PTZNet: Siamese Network Design for PTZ Estimation	54
4.2 Training and Evaluation Metrics	55
4.3 Real-World and Synthetic Data Mixing Strategies	61
4.4 Quantitative Matrix Analysis	63
5 Datasets	67
5.1 Dataset Motivation	67
5.2 Synthetic Dataset	68
5.3 Real-World Dataset	75
	xiii

6	Implementation	81
6.1	PTZNet Network Architecture	82
6.2	Training Pipeline	83
6.3	Optimization	84
6.4	Data Preprocessing	85
6.5	Configurable Dataset Integration	88
7	Evaluation and Results	89
7.1	Experimental Setup	90
7.2	Results	92
7.3	Matrix Analysis	98
8	Conclusion and Future Work	107
8.1	Hypotheses Validation	108
8.2	Contributions	109
8.3	Limitations	110
8.4	Future Work	111
8.5	Summary	112
	Overview of Generative AI Tools Used	113
	Grammarly	113
	DeepL Translator	113
	ChatGPT 4o (Writing Assistance)	113
	ChatGPT 4o (Coding Support)	114
	Perplexity AI	114
	Scope and Attribution of AI Assistance	114
	List of Figures	115
	List of Tables	121
	List of Algorithms	123
	Acronyms	125
	Bibliography	129

Introduction

In urban planning and traffic management, Pan, Tilt and Zoom (PTZ) camera systems are widely used to monitor streets, intersections, and public spaces. These cameras provide rich information on the movement and patterns of vehicles, pedestrians, and other objects, making them essential for tasks such as multi-camera tracking, scene understanding, and automated monitoring, even under challenging conditions such as fog, rain, or nighttime operation (see Figure 1.1).

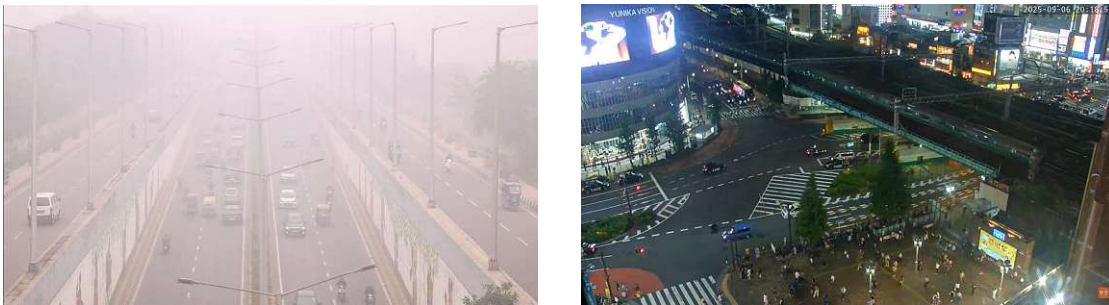


Figure 1.1: Left: a camera observing a foggy highway in northern India. Right: a camera surveying a crosswalk in Shinjuku, Tokyo, Japan. These images illustrate diverse traffic monitoring locations. Images from [New25, Sky25].

Compared to static cameras, PTZ models offer flexibility by adjusting their frustum through panning, tilting, and zooming along the camera coordinate axes (see Figure 1.2). This flexibility enables dynamic coverage of the environment but also introduces challenges when estimating relative displacement and the overlap between cameras' Fields of View (FoVs).

Estimating relative rotation and overlap from images of a single PTZ camera is difficult due to scene variability, lens distortions, dynamic motion, and limited or unavailable intrinsic



Figure 1.2: Left: front view of a PTZ camera mounted on a pole. Right: back view, also showing parts of the camera’s FoV. Images from [Gmb25, Tea23].

parameters. Collecting large, diverse real-world datasets that include the necessary camera metadata is expensive, logistically complex, and often requires repeated calibration under varying lighting, weather, and environmental conditions [CSVV20, YLZ21, ISB22, AK23].

1.1 Motivation

These challenges motivate research to leverage virtually created synthetic data, which can be generated efficiently in large quantities in controllable environments with fully accurate labels cost-effectively [ZRK⁺20, GFW⁺25]. The central question is whether relative PTZ displacement and overlap can be reliably regressed from image pairs using primarily virtually created synthetic training data, supplemented with a limited number of real-world samples. This requires careful consideration of the Synthetic-To-Real (S2R) domain gap, referring to differences in appearance, lighting, textures, and sensor noise between synthetic and real-world images, as well as the development of strategies to mitigate its impact during training.

This work addresses these challenges by analyzing the domain gap of virtually created synthetic PTZ data, proposing a training framework that strategically combines synthetic and real-world images via Batch-Level Mixing (BLM), and evaluating how synthetic data enhances inference performance under varying amounts of real-world samples. Experiments are conducted using PTZNet, a Siamese deep learning model developed for this diploma thesis, for joint regression of relative PTZ rotation and overlap, providing a foundation to answer the research questions and test the hypotheses formulated in this thesis.

1.2 Problem Statement

This thesis addresses two co-dependent challenges for PTZ camera-based relative displacement estimation:

1.2.1 Problems

- (P1) The domain gap, i.e., the feature distribution discrepancy that negatively affects neural network generalization [WD18, FVRA20], also concerns synthetic PTZ data, where it degrades rotation and overlap regression on real images. Addressing this gap requires systematically evaluating domain adaptation strategies, including the optimal balance and mixing of synthetic and real-world data.
- (P2) For PTZ cameras, gathering huge amounts of diverse real-world images, including the required metadata like camera intrinsics and extrinsics, is expensive and complex. Specialized equipment is required, and cameras have to be remounted in different, varying scenes [CSVV20, YLZ21, ISB22, AK23].

1.2.2 Research Questions

Based on the problems listed above, we formulate the following research questions that will be covered by this work:

- (Q1) Can we use neural networks to derive PTZ displacement and overlap data with competitive performance (Section 3.1.6) by comparing two images from a single PTZ camera?
- (Q2) Can we mainly rely on virtually created synthetic data for training of (Q1)? In particular:
- (Q2.1) How does purely synthetic training data affect inference performance on real-world images, in terms of rotation and overlap prediction?
 - Addresses (P1) by measuring the domain gap when no real data is used.
 - (Q2.2) How do different Real-To-Synthetic (R2S) data batch mixes during training impact prediction accuracy, compared to simply merging the two datasets?
 - Addresses (P1) by finding the optimal mixing strategy to minimize the domain gap.
 - (Q2.3) When varying the absolute number of real-world samples, how much does adding synthetic data help?
 - Addresses (P2) by quantifying how synthetic data can compensate for scarce real PTZ images.

1.2.3 Hypotheses

The research questions above motivate the following hypotheses. We expect that the strategic combination of virtually created synthetic data and limited real-world samples can enable competitive performance (Section 3.1.6) for regressing relative PTZ camera displacement and overlap. Specifically, a methodical training strategy should allow deep

neural networks to mitigate the S2R domain gap and make effective use of synthetic data. The hypotheses below formalize these expectations:

- (H1) A deep neural network trained on a methodically combined dataset of virtually created synthetic and real-world PTZ image pairs can regress the relative rotation and overlap with competitive performance (Section 3.1.6) provided both images originate from the same PTZ camera.
- (H2) Incorporating virtually created synthetic data in the training of a PTZ rotation and overlap regressor, especially via Batch-Level Mixing (BLM) strategies, significantly improves inference performance on real-world data over using synthetic-only or real-world-only data, also if the number of real-world images varies.
 - (H2.1) Integrating synthetic and real-world data for training improves model performance compared to using either synthetic-only or real-world only data, with synthetic-only training yielding the weakest performance.
 - (H2.2) Batch-Level Mixing of synthetic and real-world data leads to better model performance than Dataset-Level Mixing.
 - (H2.3) Incorporating synthetic data enhances model performance even as the amount of real-world data increases.

1.2.4 Contributions

To address the problems, research questions, and test the hypotheses outlined above, this thesis proposes the following contributions:

- (C1) Providing an analysis of the Synthetic-To-Real (S2R) domain gap of virtually created synthetic Pan, Tilt and Zoom (PTZ) camera data, focusing on its effect on inference performance (P1).
- (C2) Proposing a framework that prioritizes Batch-Level Mixing (BLM) strategies over general Dataset-Level Mixing (DLM) to achieve the highest inference accuracy for addressing the S2R domain gap (P1).
- (C3) Demonstrating quantitatively how virtually created synthetic training data can enhance the inference performance of neural networks regressing relative PTZ rotation and overlap, particularly when the number of real-world images is limited and varies (P2).
- (C4) Proposing PTZNet, a relative PTZ pose and overlap regression deep-learning model, trained on a combination of virtually created synthetic and real-world data (Q1).

1.3 Methodology

We continue with explaining the methodology for answering our research questions.

1.3.1 Literature Review

The first step of this thesis is an in-depth review of the State of the Art (SOTA) in camera pose estimation and domain adaptation techniques, with a focus on deep learning and regression tasks, as well as S2R transfer (P1). This review identifies research gaps and motivates the design of PTZNet, a deep learning model for PTZ rotation and overlap estimation trained mainly on virtually created synthetic data (P2). Insights from the review shape the design of PTZNet and the experimental approach.

1.3.2 Definition of Training and Evaluation Metrics

We define metrics required during model training and to evaluate performance in PTZ rotation and overlap estimation. These metrics are designed to capture both accuracy and robustness, enabling systematic comparison across different training configurations and datasets. They are also the baseline of the design of domain adaptation experiments and help quantify generalization to real-world scenarios (P1).

1.3.3 Dataset Creation

To support model training, we create diverse datasets combining extensive synthetic data with a limited number of real-world samples. The synthetic data provides wide coverage of different weather, lighting, and movement scenarios, enabling the model to learn a broad range of PTZ rotations and overlaps. Real-world samples, though few, are crucial for evaluating model performance under real conditions and for testing domain adaptation strategies with varying amounts of real-world data included (P2).

1.3.4 Initial Trials and Hyperparameter Tuning

Initial trials and hyperparameter tuning are performed to establish a robust baseline model before full-scale training. These experiments help identify optimal network parameters, training strategies, and data configurations, ensuring a solid foundation for subsequent domain adaptation studies (Q1).

1.3.5 S2R Domain Adaptation Experiments and Evaluation

We perform experiments to assess the impact of synthetic data and domain adaptation strategies on model performance. Evaluations are designed to systematically compare different training configurations, quantify transferability from synthetic to real data, and identify the most effective approaches for improving generalization (Q2).

1.4 Thesis Structure

This thesis is structured as follows.

Chapter 2 introduces the necessary background on machine learning, deep neural networks, and the mathematical basics of PTZ camera displacement, including 3D rotations, quaternions and FoV overlap.

Chapter 3 reviews prior work in pose regression and domain adaptation, with a focus on S2R adaptation and BLM vs. Dataset-Level Mixing (DLM) strategies. This motivates our contributions (C1) and (C2).

Chapter 4 outlines the overall research methodology, including the design of PTZNet (C4), training and evaluation metrics, and dataset strategies.

Chapter 5 presents the motivation alongside the synthetic and real-world datasets used in the quantitative PTZ study (C3).

Chapter 6 details the implementation of PTZNet, including architecture, training pipeline, and preprocessing.

Chapter 7 reports our results, comparing training strategies and analyzing the impact of synthetic data and BLM/DLM approaches.

Finally, Chapter 8 concludes with findings, contributions (C1)–(C4), and directions for future research.

Background and Key Concepts

As this work will cover a domain adaptation study (Q2), evaluating the effectiveness of mainly using virtually created synthetic data to train a Pan, Tilt and Zoom (PTZ) camera rotation and overlap estimation deep learning model (Q1), we start by introducing the foundational works of machine learning (Section 2.1), covering:

- **Machine Learning** fundamentals (Section 2.1.1)
- Characteristics of **Artificial Neural Networks (ANNs)** and **Deep Learning** (Section 2.1.2)
- Specialized **Deep Learning Architectures** required for our PTZ rotation and overlap model (Section 2.1.3)

This is followed by covering the essential mathematical background for PTZ camera displacement (Section 2.2):

- **3D Rotations** (Section 2.2.2)
- **Quaternion Representation** of 3D Rotations (Section 2.2.3)
- The computation of **PTZ-image overlap** (Section 2.2.4)

This chapter thus prepares the reader for the subsequent chapters of this thesis by introducing the necessary terminology, concepts, and mathematical tools.

2.1 Machine Learning: From Fundamentals to Deep Neural Networks

Artificial intelligence is a broad field focused on the creation of systems capable of performing tasks that typically require human intelligence. Within this field, machine learning has emerged as a core approach, enabling systems to learn from data rather than relying on hand-crafted rules (Figure 2.1). This section introduces the foundational ideas of machine learning, which ultimately lead to deep learning and the neural network architectures used in this work.

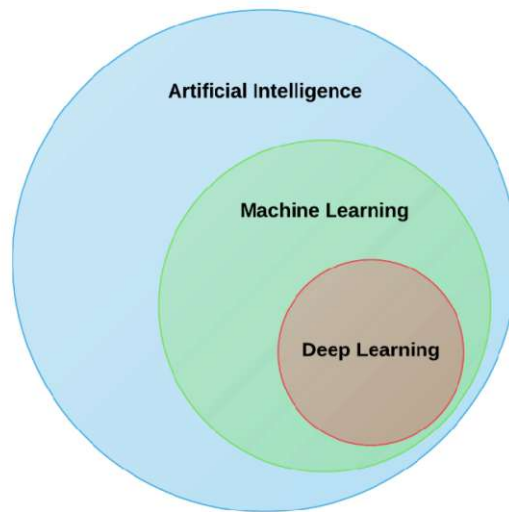


Figure 2.1: Hierarchical relationship of artificial intelligence, machine learning, and deep learning. Image from [VSvLD⁺20] page 27.

2.1.1 Machine Learning

Unlike traditional computer programs, which follow explicit, deterministic rules defined by programmers, machine learning systems are based on statistical methods and pattern recognition. They learn from data to make predictions or decisions. Machine learning is commonly used for tasks that are hard to solve using standard programs. Machine learning algorithms process examples and derive statistical information from their features in these tasks. Common tasks solved with machine learning are classification and regression. In classification tasks, the machine learning program assigns an input to one of N categories, either by predicting a single class label $k = f(x)$, where $k \in 1..N$, or by outputting a probability distribution $P(k|x)$ over all possible categories. A very popular example is image classification, where the model predicts the objects present in an input image (e.g., “cat”, “dog”).

In contrast, regression tasks predict numerical values corresponding to an input. The basic

idea is similar to classification, but the output function $y = f(x)$ produces continuous values $y \in \mathbb{R}$ rather than discrete categories. Our PTZ camera rotation and overlap model fits this regression task, as we aim to predict continuous rotation and overlap values that describe the displacement of two input images [GBC16].

Machine learning, a subfield of artificial intelligence, is commonly categorized into three foundational types: supervised, unsupervised, and reinforcement learning. Each is distinguished by how it processes information and adapts to patterns [GBC16].

In unsupervised learning, algorithms are exposed to raw, unlabeled data. Their primary goal is to discover underlying structures, like clusters or associations, without any supervision about what those patterns should be. This approach is often used for grouping similar data points or reducing the dimensionality of complex datasets [GBC16].

Supervised learning, on the other hand, involves learning from examples where each input is paired with a target label or value. The system processes such labeled examples and infers relationships between inputs and outputs. It can effectively learn to predict labels or values for new, unseen data. Example tasks are image classification or regression [GBC16].

In contrast, a reinforcement learning algorithm is often called an agent, actively interacting with its environment. Instead of learning from a static dataset, such an agent receives feedback through rewards or penalties as it makes decisions. It improves over time while adapting its strategy to maximize cumulative rewards, much like learning through trial and error [GBC16].

2.1.2 Artificial Neural Networks (ANNs) and Deep Learning

One of the most widely used approaches in modern machine learning is the use of ANNs. These models form the foundation of deep learning, the category our relative PTZ rotation and overlap model falls into, enabling the representation of complex, non-linear functions through layered architectures.

Neuron & Neural Networks

ANNs were originally inspired by the science of how brains learn, with the term “Artificial Neural Network” reflecting their conceptual roots in biological neurons and their interconnections, with the foundational works being from McCulloch and Pitts in 1943 [MP43]. While biological neurons fire when the sum of their inputs exceeds a threshold, modern ANNs often use activation functions like Rectified Linear Unit (ReLU) [NH10] (Figure 2.3), which activate only when the input is positive, mimicking this thresholding behavior. However, the ReLU function $\max(0, x)$ is a mathematical abstraction and does not fully capture the complexity of biological neuron firing. Thus, an ANN consists of multiple interconnected artificial neurons (or perceptrons [Ros58], Figure 2.2), each receiving inputs x_0, \dots, x_n with associated weights w_{ij} . These inputs are summed, shifted by a bias term θ_j , and passed through an activation function φ to produce the neuron’s

2. BACKGROUND AND KEY CONCEPTS

output o_j . Connecting and chaining multiple neurons in parallel and/or subsequent order results in an Artificial Neural Network [GBC16].

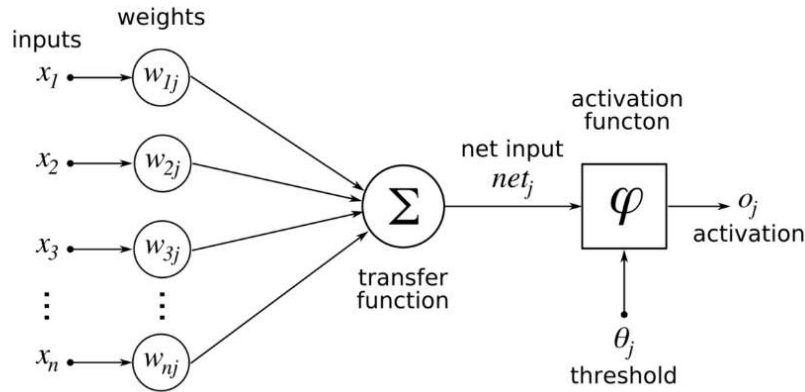


Figure 2.2: Schema of a single ANN neuron, where weighted inputs x_i with weights w_{ij} are linearly summed by the transfer function Σ into net_j , then with the bias θ_j passed through the non-linear activation function φ to produce the output o_j . Image from [i2t19].

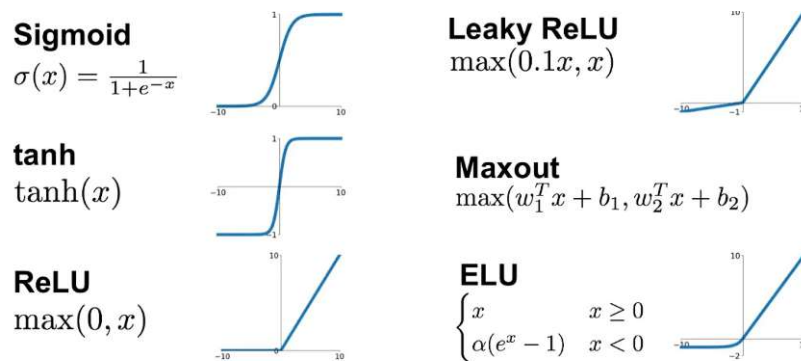


Figure 2.3: Comparison of common ANN activation functions, showing their mathematical definitions and response graphs, including sigmoid, hyperbolic tangent, ReLU, leaky ReLU, maxout function, and Exponential Linear Unit (ELU). Image from [Jad18] page 1.

These neurons are typically organized into layers (Figure 2.4): an input layer i that receives the raw data, one or more hidden layers $h_{1..n}$ that transform the input through learned intermediate representations, and an output o layer that produces the final prediction. The term hidden layer refers to the fact that these layers are not directly observable in either the input or output. They are internal to the network. The depth (i.e., number of hidden layers) and width (i.e., number of neurons per layer) of the network determine its capacity to learn complex functions. When a network contains multiple hidden layers, it is referred to as a Deep Neural Network (DNN), enabling it to model

intricate and hierarchical patterns in data [GBC16].

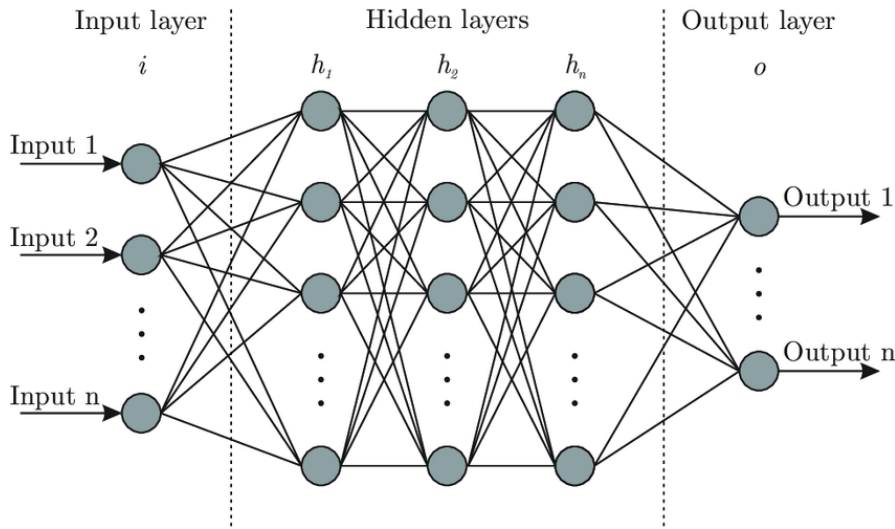


Figure 2.4: Illustration of a deep feedforward ANN architecture, with an input layer of n units, multiple fully connected hidden layers h_1, h_2, \dots, h_L , and an output layer of o units. Image from [Lav25].

Activation Function & Nonlinearity

Activation functions (Figure 2.3) such as sigmoid, tanh, and ReLU are essential for introducing non-linearity into such a network. Without nonlinearity, an ANN would be limited to representing only linear relationships, regardless of its depth or complexity. Non-linear activation functions enable ANNs to learn and model the complex, non-linear relationships found in real-world data [GBC16].

Loss Function

To enable an ANN to make its predictions, it must first be trained. Training refers to the process by which the network learns to map inputs to desired outputs. To facilitate this, we need a way to quantify how well the predicted output o_{pred} matches the expected target o_{target} . This is achieved using a *loss* function (Equation 2.1), which measures the discrepancy between the prediction and the ground truth [GBC16].

$$loss = f(o_{pred}, o_{target}) \quad (2.1)$$

The loss is computed after a forward pass, in which the input is propagated through the network to produce o_{pred} . This value acts as a feedback signal. The higher the loss, the less accurate the prediction. During training, the network aims to minimize the loss by adjusting its weights. This is performed through backpropagation and an optimization algorithm such as gradient descent, which uses the loss to compute how

each weight should change. Common loss functions include the Mean Squared Error (MSE) (Equation 2.2) and the Mean Absolute Error (MAE) (Equation 2.3) [GBC16].

$$MSE(o_{pred}, o_{target}) = \frac{1}{n} \sum_{i=1}^n (o_{pred}^{(i)} - o_{target}^{(i)})^2 \quad (2.2)$$

$$MAE(o_{pred}, o_{target}) = \frac{1}{n} \sum_{i=1}^n |o_{pred}^{(i)} - o_{target}^{(i)}| \quad (2.3)$$

Backpropagation

Backpropagation is the algorithm used to compute the gradient of the loss concerning each network parameter by applying the chain rule of calculus in a reverse-mode pass through the layers. After the loss is evaluated at the output, the error signal is propagated backwards. At each layer, the partial derivative of the loss for that layer’s pre-activation is computed, and then used to obtain derivatives for the weights and biases. This efficient procedure ensures that every parameter update is informed by its precise contribution to the overall error. Without backpropagation, training deep architectures would be computationally unfeasible [GBC16].

Gradient Descent / Optimization

Gradient descent (Figure 2.5) is an iterative procedure for adjusting each weight and bias to minimize the network’s loss. At each step, we compute the gradient, which is the vector of partial derivatives of the loss concerning every parameter, since this vector points towards the steepest increase in loss. Moving in the opposite direction, therefore, produces the fastest decrease in error. In its simplest form, stochastic gradient descent, these gradients are evaluated on a small randomly chosen subset of the training data called a batch, and each parameter is updated by subtracting a fraction of its gradient determined by the learning rate from its current value. This form of stochastic sampling not only accelerates each update, but it also introduces sufficient noise to help the model avoid shallow local minima and explore diverse regions of the loss landscape [GBC16].

Modern optimizers like Momentum [Qia99], RMSProp [HSS12], and Adam [KB14] build on the basic idea of gradient descent by introducing mechanisms that help with convergence and stability. For example, they may keep track of a running average of past gradients (as a kind of velocity) to dampen oscillations in steep directions, or adapt the step size for each parameter based on recent gradient magnitudes. These improvements often lead to faster training and better generalization in DNNs, while also reducing the need for manual learning rate tuning.

2.1.3 Specialized ANN Architectures

As these optimization strategies are generally applicable to many types of deep neural networks, they are especially beneficial when training more structured architectures

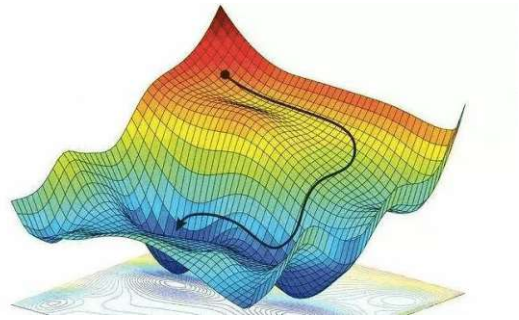


Figure 2.5: Gradient descent trajectory on a two-dimensional cost surface, showing iterative steps from the initial parameter estimate to a local minimum. Image from [Xia19].

relevant for computer vision tasks. One prominent example is the Convolutional Neural Network (CNN).

Convolutional Neural Network (CNN)

CNNs (Figure 2.6) are a type of DNNs particularly well-suited for processing grid-like input data, such as (PTZ camera) images. They consist of a series of convolutional layers that apply small, learnable filters, also called kernels, sliding over the input (b) to extract local spatial features. This enables parameter sharing and helps the model learn hierarchical representations. Early layers capture basic patterns like edges and textures, while deeper layers recognize more complex structures such as shapes or objects [LBBH98].

Pooling layers (c) are typically positioned between convolutions to downsample the feature maps, reducing computational load and providing translational invariance. After the final convolution and pooling operations, the resulting feature maps are flattened and passed into a multilayer perceptron (d), or fully connected layer, which produces the final output. Non-linear activation functions are applied throughout to enable the network to model complex relationships in the data [LBBH98, ZF14].

Siamese Neural Network (SNN)

SNNs build on the CNN architecture by using two or more identical sub-networks (Figure 2.7), also named their backbone, that share weights and parameters. The primary goal of an SNN is to learn a similarity function between inputs, which we can use to compare two images of a PTZ camera to regress their relative displacement and overlap. Each branch of the network processes one input and maps it into a shared embedding space, where a distance or similarity function compares the resulting vectors. This setup enables the model to assess how similar or different two inputs are [KZS15].

The concept was first introduced by Bromley and LeCun [BGL⁺93], who used two time-delay neural networks with a cosine similarity function to distinguish between genuine

2. BACKGROUND AND KEY CONCEPTS

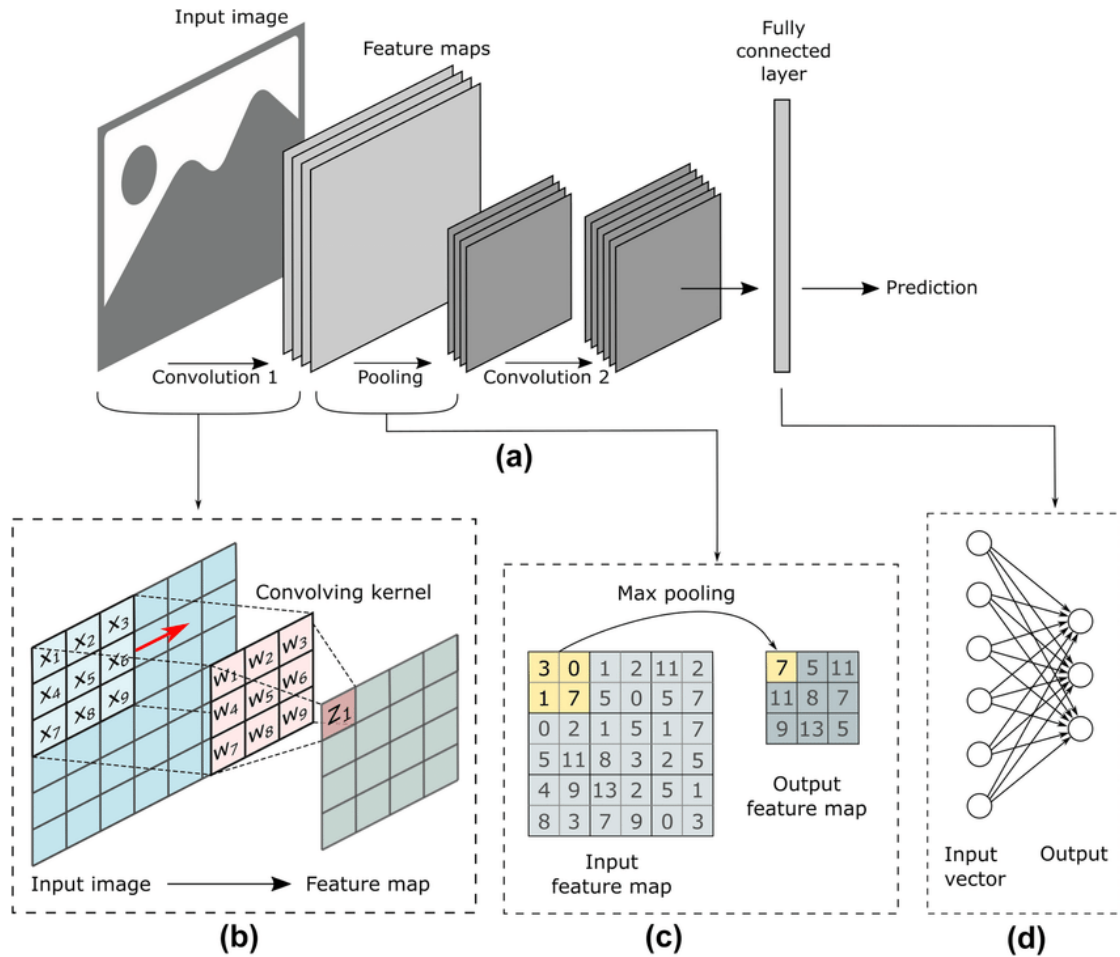


Figure 2.6: CNN architecture (a) where an input image is processed by convolutional layers with an intermediate pooling step to produce feature maps, followed by a fully connected layer for prediction. The convolution applies a sliding kernel (b) over the input to extract local features, max pooling (c) down-samples each feature map, and the flattened feature vector is fed into a multilayer perceptron (d) to generate the final output. Image from [AMS⁺22] page 3721.

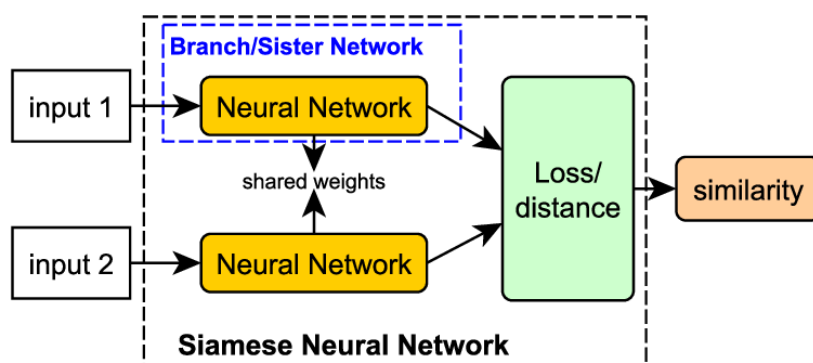


Figure 2.7: Siamese Neural Network (SNN) architecture with two identical CNN-based backbones sharing weights and parameters. Each branch encodes one input into a common embedding space, and a distance metric compares their embeddings to learn a similarity function. Image from [ZFCB⁺22] page 82.

and forged signatures. Koch et al. [KZS15] later extended this idea by using convolutional layers for one-shot image classification. They trained the model with a contrastive loss, which encourages embeddings of similar examples to be close together and those of dissimilar examples to be further apart.

Today, SNNs are widely used in tasks such as face verification, image retrieval, and domain adaptation, where quantifying cross-domain feature similarity is crucial. Common backbone architectures include convolutional models such as ResNet [HZRS16]. A specific application is Relative Pose Regression (RPR), where an SNN compares two images of the same scene to estimate the relative camera pose in terms of translation and rotation (see Section 3.1.4 for more details). Given these capabilities, SNNs appear to be a suitable tool for our domain adaptation study and will be used to address (Q1), which investigates whether ANNs can estimate PTZ displacement with competitive performance (Section 3.1.6) and measure the overlap between two related images.

2.2 PTZ Camera Displacement: Mathematical Foundations

To train and evaluate our neural network for domain adaptation, we require metadata describing PTZ camera displacement and image overlap. This section introduces the mathematical foundations necessary to compute this information.

2.2.1 Notation and Conventions

- **Scalars:** Denoted by italic letters $x \in \mathbb{R}$. The italic letter $q \in \mathbb{H}$ is reserved exclusively for quaternions.

- **Vectors:** For $n \in \mathbb{N}$, vectors are denoted by bold lowercase letters $\mathbf{v} \in \mathbb{R}^n$. Bold $\mathbf{q} \in \mathbb{R}^4$ is reserved explicitly for the vector form of a quaternion.
- **Matrices:** For $n, m \in \mathbb{N}$, matrices are denoted by uppercase bold letters $\mathbf{R} \in \mathbb{R}^{n \times m}$.
- **Angles:** Angles $\alpha, \beta, \gamma, \dots$ are in radians unless specified in degrees, e.g., α° .
- **Trigonometric functions:** $\sin(\cdot)$ and $\cos(\cdot)$ denote the sine and cosine functions, respectively.

2.2.2 Introducing Rotations in the Special Orthogonal Group in 3D - SO(3)

To address (Q1), whether neural networks can estimate PTZ displacement and overlap with competitive performance (Section 3.1.6), we first need to formalize how such displacement is measured. In the case of PTZ cameras, this mainly involves relative rotation. We begin with a brief overview of orientation-preserving 3D rotations within the Special Orthogonal Group SO(3), as defined in Equation 2.4

$$\text{SO}(3) = \{ \mathbf{R} \in \mathbb{R}^{3 \times 3} : \mathbf{R}^\top \mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \}, \quad (2.4)$$

where \mathbf{I} is the 3×3 identity matrix, and $\det(\mathbf{R})$ denotes the determinant of \mathbf{R} [Tri09].

In what follows, we work in the standard Cartesian coordinate system of \mathbb{R}^3 with the orthonormal basis, as defined via unit vectors \mathbf{e}_x , \mathbf{e}_y , and \mathbf{e}_z in Equation 2.5

$$\mathbf{e}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad (2.5)$$

corresponding to the X -, Y -, and Z -axes, respectively. Note that uppercase letters X , Y , Z denote axis labels used in Euler-angle rotations and are distinct from the unit vectors \mathbf{e}_x , \mathbf{e}_y , \mathbf{e}_z .

Rotations are described using Euler angles. We decompose the transformation into three sequential rotations about the coordinate axes \mathbf{e}_x , \mathbf{e}_y , \mathbf{e}_z (i.e., the X -, Y -, and Z -axes), as depicted in Figure 2.8. The Euler angles encode the rotation parameters and are represented by the vector $\mathbf{r}_e \in \mathbb{R}^3$, defined in Equation 2.6

$$\mathbf{r}_e = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}, \quad (2.6)$$

where the angles α , β and γ denote rotations about the X -, Y -, and Z -axes, respectively.

Consider a directional vector represented by the unit vector $\mathbf{v} \in \mathbb{R}^3$, which corresponds to the center of a PTZ camera's Field of View (FoV) in the camera coordinate system. To achieve a desired viewing direction $\mathbf{v}' \in \mathbb{R}^3$, the relative orientation between \mathbf{v} and

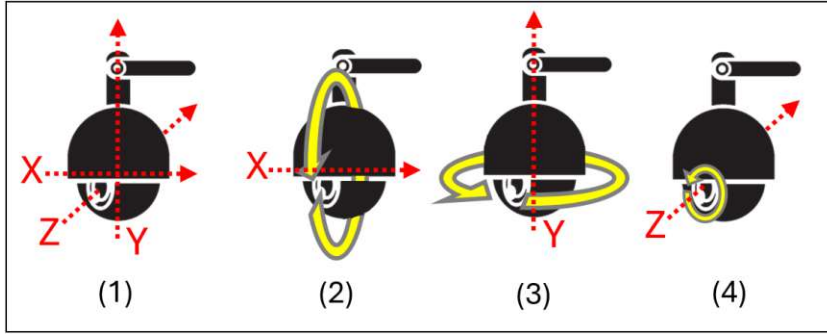


Figure 2.8: Illustration of the theoretical mechanical rotation capabilities of a PTZ camera. (1) The camera's internal coordinate system, with the Y-axis serving as the up vector. (2) Tilt rotation about the X-axis. (3) Pan rotation about the Y-axis. (4) Roll rotation about the optical Z-axis, a degree of freedom not supported by all camera models. Roll should not be confused with zooming, which modifies the camera's intrinsic parameters.

\mathbf{v}' can be described by a rotation matrix $\mathbf{R} \in \text{SO}(3)$ constructed from Euler angles, as defined in Equation 2.8. In this representation, the overall rotation is expressed as a sequence of rotations about the X-, Y-, and Z-axes, following the classical Euler-angle convention. Applying this rotation to the vector results in Equation 2.7 [Sla99]

$$\mathbf{v}' = \mathbf{R}\mathbf{v}, \quad (2.7)$$

so that \mathbf{v}' corresponds to the resulting orientation.

$$\begin{aligned} \mathbf{R}_x(\alpha) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix} \\ \mathbf{R}_y(\beta) &= \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \\ \mathbf{R}_z(\gamma) &= \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \quad (2.8)$$

The rotation order fundamentally affects the final orientation because matrix multiplication is not commutative, as is the case with 3D rotations. For matrices \mathbf{R}_x and \mathbf{R}_y , the multiplication $\mathbf{R}_x\mathbf{R}_y$ means that a vector \mathbf{v} is first rotated by \mathbf{R}_y and then by \mathbf{R}_x . Thus, $\mathbf{R}_x\mathbf{R}_y \neq \mathbf{R}_y\mathbf{R}_x$, and it is necessary to specify both the order and axes of rotation, as shown in Equation 2.9 [Sla99].

$$\mathbf{v}' = \mathbf{R}_x(\mathbf{R}_y\mathbf{v}) \neq \mathbf{R}_y(\mathbf{R}_x\mathbf{v}) = \mathbf{v}'' \quad (2.9)$$

To achieve reproducible rotations from \mathbf{v} to \mathbf{v}' , it is necessary to specify both the rotation sequence and the reference frame convention [Sla99]. For extrinsic rotations (about fixed global axes, i.e., as a mechanical PTZ camera would move relative to the scene), an XYZ sequence corresponds to Equation 2.10 [Sla99]

$$\mathbf{v}' = \mathbf{R}_z(\gamma) \mathbf{R}_y(\beta) \mathbf{R}_x(\alpha) \mathbf{v}, \quad (2.10)$$

whereas, for intrinsic rotations \mathbf{v}'_i (about the moving body axes), the same XYZ sequence is applied in reverse order (Equation 2.11) [Sla99]

$$\mathbf{v}'_i = \mathbf{R}_x(\alpha) \mathbf{R}_y(\beta) \mathbf{R}_z(\gamma) \mathbf{v}. \quad (2.11)$$

The final orientation depends on both the chosen rotation sequence and whether the rotations are intrinsic or extrinsic, highlighting the importance of specifying the convention [Sla99].

Thus, while Euler angles provide an intuitive means of representing three-dimensional rotations, they are associated with several limitations. Specifically, Euler angles are prone to [Sla99]:

- **Gimbal lock:** This occurs when two of the three rotational axes become aligned, losing one degree of freedom. This can lead to inaccurate rotational calculations or difficulties in controlling motion.
- **Ambiguity:** A single physical orientation may be represented by multiple distinct sets of Euler angles, complicating the interpretation and consistency of the rotation description.
- **Discontinuities:** Near its range boundaries, small orientation changes can lead to abrupt jumps in the Euler angle values. These discontinuities hinder smooth transitions in applications that require continuous rotational updates.

In contrast, rotation matrices offer a robust and unambiguous representation of rotations [Sla99]. However, they require a larger number of parameters, which can be computationally expensive in terms of storage and processing, especially in high-dimensional applications.

2.2.3 Quaternions in the Special Orthogonal Group in 3D - SO(3)

Rotations in 3D can be intuitively understood as a turn by an angle θ around a fixed axis, which we represent by the vector $\mathbf{v} \in \mathbb{R}^3$ defined in Equation 2.12.

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad (2.12)$$

This axis–angle perspective describes any orientation change as a single, unified transformation rather than a sequence of rotations around the individual coordinate axes. While this view does not apply to representations such as Euler angles, which describe orientation using multiple sequential rotations about the X -, Y -, and Z -axes, it forms the basis for the unit quaternion representation.

Quaternions offer a compact, efficient, and numerically stable alternative that is especially beneficial for applications like PTZ camera control, where smooth transitions between orientations are critical [Voi21]. In camera pose estimation tasks, quaternions are commonly used to describe camera rotation [DWS⁺19, YLZ20, RMVH22, LYS⁺23]. We introduce the basics of quaternions as important tool for this thesis and our definition of relative rotation, especially for our train and evaluation metrics definition in Section 4.2.

A quaternion $q \in \mathbb{H}$ [Vic01, Ebe02] can be written in the classical Hamilton form (Equation 2.13)

$$q = w + xi + yj + zk, \quad (2.13)$$

where i, j, k are the quaternion imaginary units.

For reference, the same quaternion $q \in \mathbb{H}$ can also be written as a combination of its scalar $w \in \mathbb{R}$ and vector parts $\mathbf{v} \in \mathbb{R}^3$ as $\mathbf{q} \in \mathbb{R}^4$ (Equation 2.14) [Vic01, Ebe02], being

$$\mathbf{q} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}, \quad \text{vector}(\mathbf{q}) = \text{vector}(q) = \mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \text{scalar}(\mathbf{q}) = \text{scalar}(q) = w \quad (2.14)$$

where $w, x, y, z \in \mathbb{R}$.

The conjugate of a quaternion [Vic01, Ebe02] (Equation 2.15) is denoted by $\bar{\mathbf{q}} \in \mathbb{R}^4$ (or equivalently $\bar{q} \in \mathbb{H}$) and is obtained by negating the vector part. Intuitively, the conjugate is similar to taking a vector and “mirroring” it, which is useful for inverting rotations.

$$\bar{\mathbf{q}} = \begin{bmatrix} w \\ -\mathbf{v} \end{bmatrix} = \begin{bmatrix} w \\ -x \\ -y \\ -z \end{bmatrix}, \quad (2.15)$$

$$\bar{q} = w - xi - yj - zk,$$

$$\bar{\bar{\mathbf{q}}} \equiv \mathbf{q}.$$

The product of two quaternions $q_1, q_2 \in \mathbb{H}$ is defined via polynomial multiplication using the relations of i, j, k (Equation 2.16) [Vic01, Ebe02]:

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1, \\ ij = k = -ji, \quad jk = i = -kj, \quad ki = j = -ik. \end{aligned} \quad (2.16)$$

Expanding this multiplication results in [Vic01, Ebe02]:

$$\begin{aligned}
 q_1 q_2 = & \left(w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \right) \\
 & + \left(w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2 \right) i \\
 & + \left(w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2 \right) j \\
 & + \left(w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2 \right) k.
 \end{aligned} \tag{2.17}$$

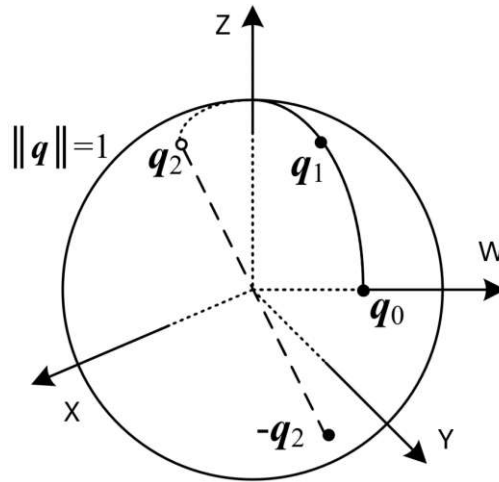


Figure 2.9: Schematic representation of unit quaternions $\mathbf{q}_{0..2}$ located on S^3 , with their axes X, Y, Z and fourth dimension W . The antipodal $-\mathbf{q}_2$ of \mathbf{q}_2 shows the double-cover property [Tri09], as both quaternions represent the same rotation in $SO(3)$. Image source: [Lei22].

If the quaternion $\mathbf{q} \in \mathbb{R}^4$ satisfies Equation 2.18, it is called a unit quaternion (Figure 2.9) [Ebe02, Tri09]. This condition guarantees that the quaternion has a unit norm, essential for preserving distances during rotation. Consequently, all unit quaternions lie on the 3-sphere S^3 [Ebe02, Tri09].

$$\|\mathbf{q}\|^2 = w^2 + x^2 + y^2 + z^2 = 1 \implies \|\mathbf{q}\| = 1 \implies \mathbf{q} \in S^3, \tag{2.18}$$

where $\|\mathbf{q}\|$ denotes the Euclidean norm of the quaternion. Unit quaternions also satisfy $q\bar{q} = 1$, which is an equivalent definition using quaternion multiplication (Equation 2.17) [Ebe02, Tri09]. This ensures that rotations preserve lengths and orientations [Ebe02, Tri09].

Any rotation in 3D can be described by an angle θ about a unit axis $\mathbf{v} \in \mathbb{R}^3$. In the standard quaternion representation, the half-angle is denoted $\phi = \theta/2$ [Vic01]. The vector part can be written in Hamilton form as $xi + yj + zk$, with (x, y, z) as the unit

axis components, or, equivalently, as the column vector $\mathbf{v} = [x, y, z]^T$ [Ebe02]. The unit quaternion for this rotation [Vic01, Ebe02] is

$$\mathbf{q} = \begin{bmatrix} \cos(\phi) \\ x \sin(\phi) \\ y \sin(\phi) \\ z \sin(\phi) \end{bmatrix}, \quad x^2 + y^2 + z^2 = 1, \quad (2.19)$$

so that the scalar part $w = \cos(\theta/2)$ and $\|\mathbf{v}\| = \sin(\theta/2)$ [Vic01, Ebe02]. Because of the half-angle representation, both \mathbf{q} and $-\mathbf{q}$ represent the same rotation, the so-called double-cover property $S^3 \rightarrow \text{SO}(3)$ [Tri09].

It is important to note that the double-cover property [Tri09] of unit quaternions maps each rotation in $\text{SO}(3)$ to two antipodal points on S^3 : if $q \in S^3$ represents a rotation, then $-q$ represents the same rotation in 3D space (Figure 2.9).

To rotate a vector $\mathbf{v} \in \mathbb{R}^3$ via $q \in \mathbb{H}$, \mathbf{v} is first treated as a “pure quaternion” $\mathbf{v}_{pq} \in \mathbb{R}^4$ with zero scalar part (Equation 2.20) [Vic01]:

$$\mathbf{v}_{pq} = \begin{bmatrix} 0 \\ v_x \\ v_y \\ v_z \end{bmatrix} \equiv q_v = 0 + v_x i + v_y j + v_z k. \quad (2.20)$$

The rotation is performed using quaternion multiplication (Equation 2.21) [Vic01]:

$$q'_v = q q_v q^{-1}. \quad (2.21)$$

The rotated 3D vector $\mathbf{v}' \in \mathbb{R}^3$ (Equation 2.22) corresponds to the vector part of q'_v [Vic01]:

$$\mathbf{v}' = \text{vector}(q'_v), \quad \text{scalar}(q'_v) = 0. \quad (2.22)$$

To compute the relative rotation $q_{rel} \in \mathbb{H}$ between two unit quaternions $q_1, q_2 \in \mathbb{H}$ (Equation 2.23) [Vic01], we multiply one by the inverse (conjugate) of the other:

$$q_{rel} = q_1 \bar{q}_2, \quad (2.23)$$

where q_{rel} represents the rotation aligning the orientation q_2 to q_1 . Since q_1 and q_2 are unit quaternions, q_{rel} is also a unit quaternion (Equation 2.18) [Vic01].

For two unit quaternions $q_1, q_2 \in \mathbb{H}$, their dot product is (Equation 2.24) [Vic01]

$$q_1 \cdot q_2 = w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2, \quad |q_1 \cdot q_2| = \cos\left(\frac{\theta}{2}\right), \quad (2.24)$$

where θ is the angular difference between the corresponding rotations. The absolute value accounts for the double-cover property, since q and $-q$ represent the same rotation [Tri09]. This provides a natural measure of rotational similarity: it equals 1 when the

rotations are identical and 0 when the corresponding rotations in $SO(3)$ differ by 180° . Intuitively, such rotations can be thought of as being “as far apart as possible” in $SO(3)$, corresponding to quaternions that are orthogonal vectors on S^3 [Ebe02, Tri09].

This section reviewed the mathematical foundations necessary for both training and evaluating models that operate on PTZ orientations. Specifically, the properties of quaternions within $SO(3)$ enable us to define and compute rotational differences in a well-defined manner, which we then use to formulate training and evaluation metrics in Section 4.2 (Q1).

2.2.4 Intersection over Union on the Unit Sphere Surface

A key baseline for addressing (Q1) is the definition and quantification of overlap between two PTZ camera FoVs. Here, overlap refers to the spatial intersection of their FoVs, measured using the Intersection over Union (IoU) metric on the unit sphere. This chapter introduces the IoU as a quantitative measure of FoV overlap, explaining its computation and its importance for evaluating how closely predicted camera positions match the targets. The IoU metric is central to this thesis, especially in Section 4.2, where it serves as the main measure of overlap for model supervision during training and evaluation. Additionally, using the IoU makes it straightforward to enforce a minimum overlap threshold during training, ensuring that images with little or no overlap are excluded from the training process. The IoU overlap between two PTZ camera FoVs is calculated with the pipeline described in Algorithm 2.1. To detail the underlying calculations, we start by defining the general IoU metric for two sets A and B by Equation 2.25 [RTG⁺19].

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \quad (2.25)$$

This metric is commonly used for object detectors but is generally applied on 2D surfaces and does not work with areas on a 3D sphere without modifications or usage of specialized libraries. To overcome this issue, we leverage the Spherical Geometry library [Sph25a] in our Algorithm 2.1.

To compute each camera’s actual FoV position on the unit sphere (Figure 2.10), we temporarily convert from spherical (*latitude, longitude*) to Cartesian coordinates (x, y, z). This simplifies rotation, which is otherwise non-trivial on a curved surface due to the need for spherical trigonometry or complex coordinate transformations. In Cartesian space, 3D rotation matrices, as mentioned in Section 2.2.2, can be applied directly and efficiently. In Figure 2.10 we show an example using a rectangular FoV defined by its angular width 40° and height 30° , initially centered around $(0^\circ, 0^\circ)$ on the sphere with spherical coordinates (a). Its four corner points are converted to Cartesian coordinates (x, y, z) (b). Using the camera’s tilt, pan, and roll information, we apply 3D rotations to these points (c). The rotated corners are then projected back to spherical coordinates, yielding the actual FoV orientation on the unit sphere (d). This procedure is performed independently for both FoVs.

Algorithm 2.1: Union Sphere IoU Calculation for Camera FoV Overlap

```

1 Function calculateSphericalIoU (
  rotation1, fovWidth1, fovHeight1,
  rotation2, fovWidth2, fovHeight2):
  // Compute spherical corners of each camera FOV
2  corners1 ← getSphericalFov (rotation1, fovWidth1, fovHeight1);
3  corners2 ← getSphericalFov (rotation2, fovWidth2, fovHeight2);
  // Create spherical polygons from corner points
4  polygon1 ← SphericalPolygon.from_lonlat(corners1);
5  polygon2 ← SphericalPolygon.from_lonlat(corners2);
  // Calculate intersection and union regions
6  intersectionArea ← polygon1.intersection(polygon2).area;
7  unionArea ← polygon1.union(polygon2).area;
  // Compute IoU ratio
8  iou3d ←  $\frac{\text{intersectionArea}}{\text{unionArea}}$ ;
9  return iou3d

10 Function getSphericalFov (rotation, fovWidth, fovHeight):
  // Convert FOV dimensions from degrees to radians
11  fovWidthRad ← toRadians(fovWidth);
12  fovHeightRad ← toRadians(fovHeight);
  // Define corner points as local spherical coordinates
  // (latitude and longitude)
13  heightHalve ←  $\frac{\text{fovHeightRad}}{2}$ ;
14  widthHalve ←  $\frac{\text{fovWidthRad}}{2}$ ;
15  corners ← [
    (heightHalve, widthHalve), (heightHalve, -widthHalve),
    (-heightHalve, -widthHalve), (-heightHalve, widthHalve)
  ]
  // Convert spherical to Cartesian coordinates
16  pointsToRotate ← [];
17  foreach (lat, lon) in corners do
18  | x ← cos(lat) × cos(lon);
19  | y ← cos(lat) × sin(lon);
20  | z ← sin(lat);
21  | Append (x, y, z) to pointsToRotate;
22  end
  // Apply rotation to Cartesian points
23  rotatedPoints ← rotation.apply(pointsToRotate);
  // Convert rotated Cartesian points
  // back to spherical coordinates
24  rotatedFovCorners ← [];
25  foreach (x, y, z) in rotatedPoints do
26  | lat ← arcsin(z);
27  | lon ← arctan2(y, x);
28  | Append (toDegrees(lon), toDegrees(lat)) to rotatedFovCorners;
29  end
30  return rotatedFovCorners;

```

2. BACKGROUND AND KEY CONCEPTS

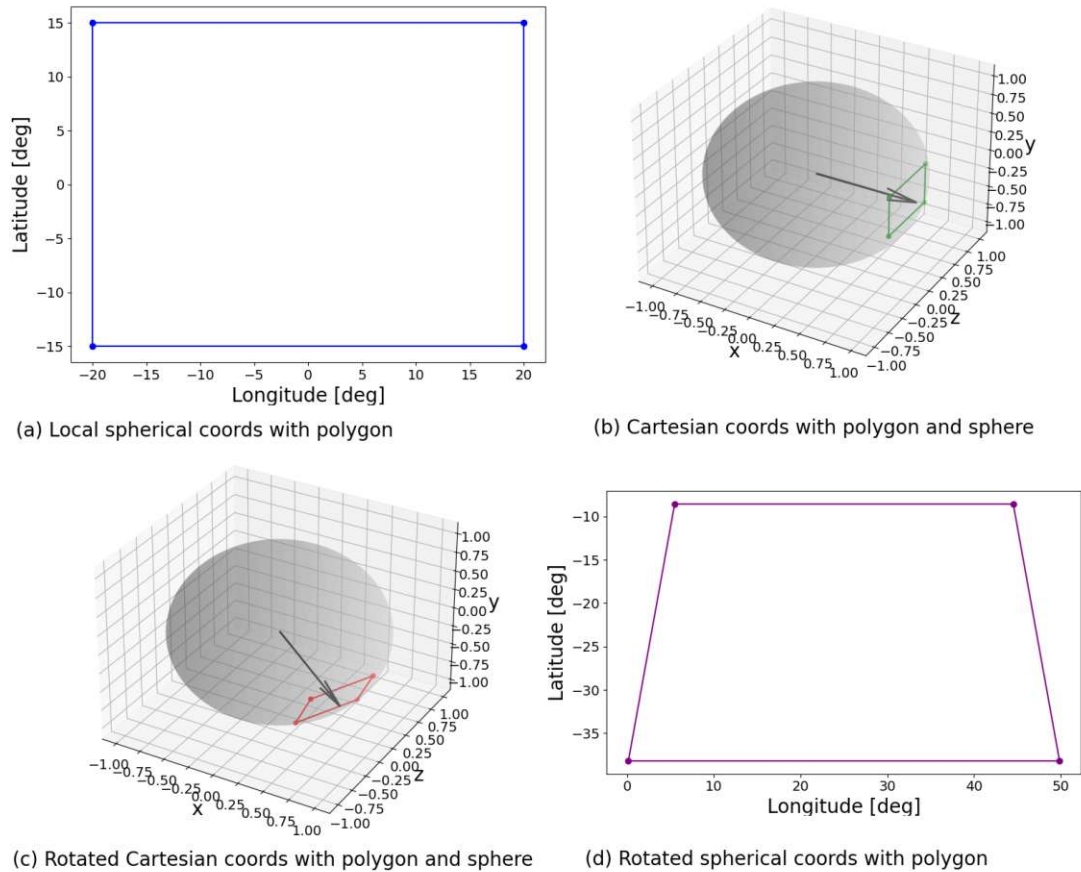


Figure 2.10: Step-by-step transformation of a rectangular FoV from its default position on the sphere to its actual orientation using PTZ parameters. (a) shows the initial spherical coordinates, (b) visualizes the polygon in Cartesian space, (c) applies rotation based on PTZ angles, and (d) maps the result back to spherical coordinates.

The resulting spherical coordinates of the corners serve as inputs to the `Spherical Polygon` of the Spherical Geometry library, which computes geodesic polygons on the unit sphere. The library calculates the steradian area A of these spherical polygons using a generalization of Girard's Theorem (Equation 2.26), where the area equals the sum of internal angles θ , with n being the number of vertices [Sph25b]. A steradian is the unit used to measure a solid angle Ω in 3D (Figure 2.11), just like a radian measures angles in 2D. One steradian corresponds to a patch of surface area equal to 1 on the unit sphere, which has a total surface area A of 4π . A full sphere thus covers 4π steradians.

$$A = \theta - (n - 2)\pi \quad (2.26)$$

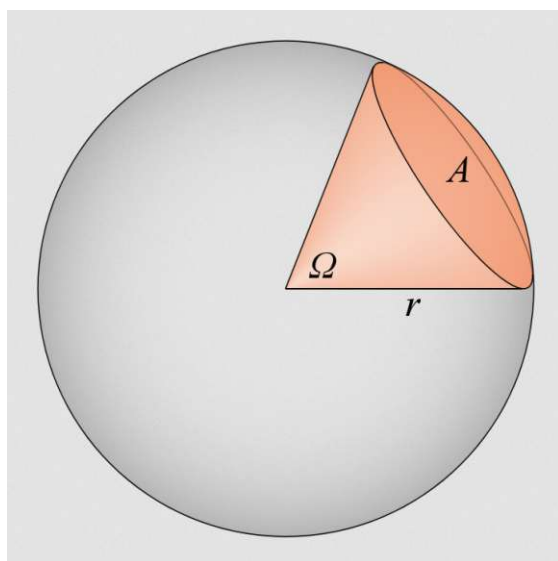


Figure 2.11: A solid angle Ω (measured in steradians) formed by a cone on a unit sphere, covering a curved surface area A . Image source: [Met25].

Following this, the common IoU calculation (Equation 2.25) is performed, with the library handling the computation of the geodesic area and any potential edge cases. This approach ensures the spherical IoU is invariant to rotation or projection effects.

To demonstrate the practical application of this metric across different PTZ camera alignments, Figure 2.12 visualizes three distinct overlap scenarios on the unit sphere. In each case, the blue and red polygons represent two independent FoVs that have been transformed according to the pipeline described above. The intersection area (green) of these blue and red FoVs corresponds to the IoU of approximately 30% on Sphere 1 (partial overlap) and 80% on Sphere 2 (high overlap). The FoV on Sphere 3 is non-overlapping and thus results in an IoU of 0%.

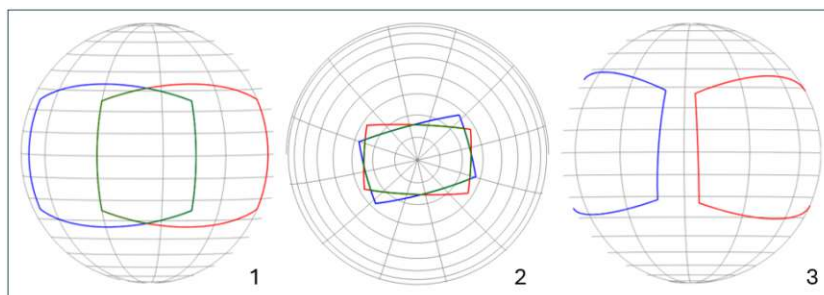


Figure 2.12: Unit spheres visualising relative PTZ camera FoVs (blue and red), projected radially outward from each sphere's center. The intersection areas (green) represent the spherical IoU across three distinct scenarios: partial overlap (Sphere 1), high overlap (Sphere 2), and no overlap (Sphere 3).

Related Work

Estimating camera pose and image overlap is an important topic in computer vision. However, as highlighted in (P2), collecting large amounts of real-world Pan, Tilt and Zoom (PTZ) camera data with accurate metadata is both expensive and technically challenging. At the same time, relying on synthetic data introduces a domain gap (P1) that often leads to reduced model performance when moving from Synthetic-To-Real (S2R) scenarios. To address these issues, recent work has focused on domain adaptation techniques that help models trained on synthetic data generalize better to real images, with particular attention to both deep learning and regression-based approaches.

This related work section provides a comprehensive review of:

- Existing methodologies for **PTZ Pose and Overlap regression** (Section 3.1)
- **Deep Domain Adaptation** techniques (Section 3.2)
- **Domain Adaptation for Regression Problems** (Section 3.3)
- The **Synthetic-To-Real (S2R)** context (Section 3.4)

The results of this investigation (Section 3.5) are briefly recapitulated and then categorized regarding their strengths and limitations (Table 3.1) as well as relevance to our problems (P1) and (P2) (Table 3.2).

3.1 Pose and Overlap Regression

Accurate estimation of camera pose and image overlap is a fundamental problem in computer vision, with applications ranging from robotics to surveillance. For Pan, Tilt and Zoom (PTZ) cameras, this task is particularly challenging due to their dynamic

nature and the complexities involved in collecting large-scale, diverse real-world datasets with precise metadata (P2).

To address these challenges, this chapter provides an overview of existing approaches to camera pose regression (Figure 3.1), which lays the foundation for investigating domain adaptation strategies aimed at closing the Synthetic-To-Real (S2R) gap. We organize this discussion into four distinct categories:

- Key-Point Detection and Feature Matching (KDFM)
- Scene Coordinate Regression (SCR)
- Absolute Pose Regression (APR)
- Relative Pose Regression (RPR)

These represent important approaches for camera pose estimation. Additionally, this chapter creates a baseline for evaluating the applicability of neural networks in deriving PTZ displacement and overlap data (Q1) with competitive performance (Section 3.1.6) and summarizes research on leveraging synthetic data to address such regression problems (Q2).

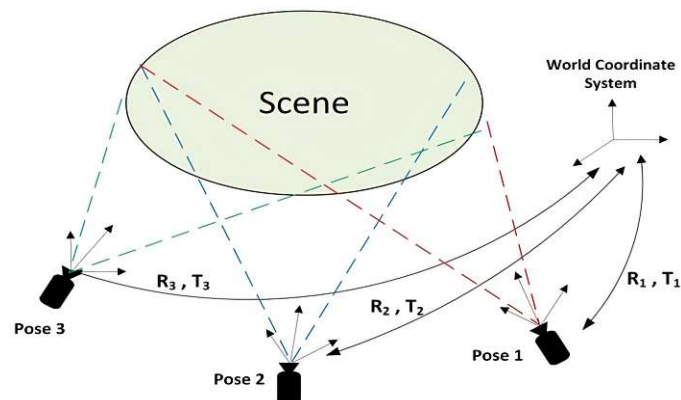


Figure 3.1: Multi-view camera pose estimation with camera rotation R and translation T . Image source: [HK18] page 94.

3.1.1 Key-Point Detection and Feature Matching (KDFM)

A classical computer vision approach for computing the transformation between two images is KDFM (see Figure 3.2). Feature detection abstracts the image information into its features. Ideally, those detection techniques should be robust to rotation, scale, illumination, noise, and affine transformations. After the detection, features are described, matched, and a transformation matrix is estimated. Although many techniques have been

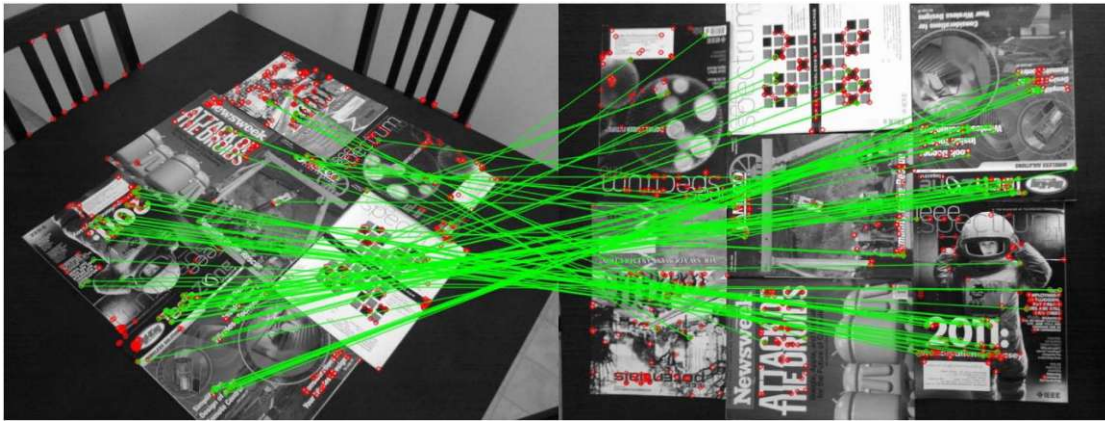


Figure 3.2: Keyframe matching identifies distinctive features in both images (red dots), and feature matching connects corresponding points across the images (green lines), enabling accurate alignment and comparison. Image source: [RRKB11] page 1

proposed, a universal solution for different sensors and image conditions is considered a challenge [KPS17, LXW21].

Proposed detectors and matchers are for example Scale-Invariant Feature Transform (SIFT) [Low04], Speeded Up Robust Features (SURF) [BTVG06] and Oriented FAST and rotated BRIEF (ORB) [RRKB11], but their performance can degrade depending on the extent of distortion, compression, blur, rotation, scale, and viewpoint changes [BKK21, LXW21].

Liu et al. [LXW21] conducted an extensive comparison between such hand-crafted technologies compared to deep learning-based ones and summarized that deep learning-based tools can outperform hand-crafted ones generally. Still, they depend on a high amount of training data for various scenes, which might not always be available, especially in our PTZ camera context (P2).

3.1.2 Scene Coordinate Regression (SCR)

SCR is a technique used for estimating the camera pose from a single RGB(-D) image relative to a known scene or environment.

In 2013, Shotton et al. [SGZ⁺13] were among the first to use machine learning for SCR. They used regression forests to predict correspondences between any image pixel and points in the scene's 3D world coordinate frame. The camera pose was then optimized using Random Sample Consensus (RANSAC) [FB81].

In 2021, Brachmann et al. [BR21] introduced a learning-based framework for visual camera re-localization, Differentiable RANSAC (DSAC)*, based on their previously established DSAC [BKN⁺17] and DSAC++ [BR18], which replaced the traditional

random forest with a fully convolutional neural network. DSAC* is capable of high-precision re-localization from a single RGB or RGB-D image and implements SCR.

However, SCR methods, which predict 3D coordinates of scene key points or pixels, have limitations in generalizing to unseen environments [XWX⁺22] and thus face the issue of a general domain gap (P1).

3.1.3 Absolute Pose Regression (APR)

APR is a computer vision technique to determine the position and orientation of a camera within a known environment or scene, estimating the 6-Degrees of Freedom (DOF) (translation and rotation) camera pose. Many different studies incorporating deep learning have been published.

In 2015, Kendall et al. [KGC15] proposed PoseNet. The model directly regresses the pose from the image without relying on tracking mechanisms. Similar to that is Hourglass Pose, proposed by Melekhov et al. [MYKR17] in 2017. The architecture, named for its hourglass shape, consists of an encoder that encodes the overall context of the input image and a decoder that recovers the fine-grained visual information. In 2019, Yang et al. [YBT⁺19] proposed SANet, designed to extract a scene representation from reference scene images and 3D points. To fuse query image features and scene features, SANet adopts a structure similar to PointNet[QSMG17], which can handle unordered point clouds, to predict the scene coordinate features. Sarlin et al. [SUL⁺21] introduced PixLoc in 2021. It is designed to find the 6-DOF pose using dense features based on classical geometric optimization. The aforementioned method assumes the availability of known sparse or dense 3D structural information.

APR directly regresses the 6-DOF pose parameters from query images using a convolutional backbone, but is like SCR generally limited in its ability to generalize to unseen environments [XWX⁺22] with its domain gap (P1). Additionally, APR is not directly applicable for our task in (Q1), as we aim to estimate the relative and not absolute rotation and overlap.

3.1.4 Relative Pose Regression (RPR)

In contrast to APR, RPR approximates translation and rotation vectors in relation to another camera's system of reference from a pair of images [RMVH22].

Balntas et al. [BLP18] developed RelocNet. They generated a new RGB-D database of entire-image features specifically tailored for camera pose retrieval. Query image and nearest neighbor features are inserted into a Siamese Neural Network (SNN) to estimate the 6-DOF pose difference between the query and the nearest neighbor images. Ding et al. [DWS⁺19] proposed CamNet in 2019. This framework operates on a coarse-to-fine retrieval basis via SNNs. In their approach, image retrieval focuses on learning scene similarities, while pose regression identifies subtle changes in view angles between paired images.

Yang et al. [YLZ20] introduced RCPNet, an end-to-end Convolutional Neural Network (CNN)-based approach for relative camera pose estimation across different urban scenes. They constructed the Tübingen Buildings dataset, consisting of 10,000 drone-captured images across eight urban locations, with absolute poses computed via Structure from Motion (SfM). Over 300,000 image pairs were formed from this dataset using SIFT-based feature matching. The network uses an SNN-architecture built on a ResNet-34 [HZRS16] backbone, predicting both rotation (as a quaternion) and translation between image pairs.

In 2022, Rajendran et al. [RMVH22] presented an end-to-end approach called RelMobNet, which uses an SNN based on MobileNetV3-Large[HSC⁺19]. Adaptive pooling layers are used, allowing the original dimensions and aspect ratio of the input images to be maintained. There were also works proposed that incorporate synthetic data into their training for RPR. Charco et al. [CSVV20, CSVV21] create an urban training data set with the CARLA [DRC⁺17] autonomous driving simulator and use it for transfer learning and re-training on real-world scenarios.

Zhang et al. [ZRK⁺20] proposed DeepPTZ, a dual-SNN for self-calibration of PTZ cameras. The framework estimates intrinsic (focal length, distortion) and extrinsic (rotation angles) parameters between image pairs, leveraging a synthetic dataset stitched from real-world panoramic images. This approach achieves robust calibration without relying on traditional assumptions, bridging the gap between feature-based methods and deep learning solutions.

RPR has the theoretical ability to generalize to unseen scenes (P1), making it a suitable candidate for our research. This allows us to address the question of whether neural networks are theoretically able to estimate PTZ displacement with competitive performance (Section 3.1.6) (Q1) and subsequently to explore the integration of synthetic data for an analytical study to answer (Q2).

3.1.5 Overlap Estimation

Building on the previous discussion of feature matching and pose estimation, this section summarizes the critical problem of overlap estimation between camera FoVs (see Figure 3.3). Efficient camera clustering methods based on overlapping FoVs have been explored for Wireless Multimedia Sensor Networks (WMSNs) [AEBM19]. The proposed method utilizes the Bron-Kerbosch algorithm [BK73] to identify maximal cliques, representing clusters of cameras with significant FoV overlaps, thereby reducing network congestion and redundant data transmission.

Predictive models such as interpretable box embeddings [RGHS⁺20] enable overlap estimation by representing visual relationships geometrically. These embeddings represent images as geometric boxes (axis-aligned hyperrectangles) in a latent space, where the size and overlap of each box directly encode meaningful visual attributes such as scale and shared content. They use the Normalized Surface Overlap (NSO) to measure the amount of features of one image being present within the other. This helps to capture asymmetric

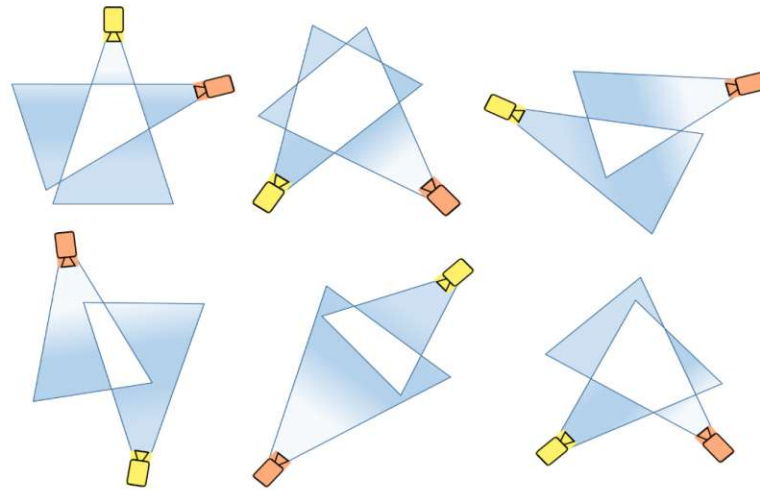


Figure 3.3: Camera FoV intersection examples with various camera mounting locations. Image source: [AEBM19] page 16

and interpretable relationships between images, making it easier to identify image pairs with significant overlap and differences in scale. This approach reduces computational costs while maintaining human interpretability, making it effective for tasks like image localization and retrieval.

Transformer-based architectures, such as Correspondence Transformer (COTR) [JTH⁺21], effectively match correspondences across images by utilizing learned attention mechanisms, which enable the model to focus on the most relevant parts of input data by assigning weights based on importance. This capability can indirectly assist overlap estimation by highlighting overlapping areas as they are most relevant for matching. Likewise, overlap-guided feature matching methods [CHX⁺22] enhance relative pose estimation by explicitly incorporating overlap predictions into the matching process.

Photogrammetry, the science of extracting measurements from images, reconstructs 3D models using overlapping areas. In photogrammetry, field-based methods were proposed to estimate overlaps for intersecting images [DUK22], offering insights into systematic overlap estimation for structured environments. Such method derives overlap fractions as a function of camera position, object geometry, and FoV, enabling practical application with minimal calibration and complexity while providing conservative but reliable estimates.

These advancements highlight the importance of precise overlap estimation of camera FoVs and propose valuable insights in how they can be used to improve feature matching. Overlap estimation is not only relevant as a trainable output parameter, it can also help in guiding the training of a PTZ pose regressor, as we can constrain a lower bound of overlap between two relative images, which is crucial to ensure a minimum amount of shared features between two images (Q1).

3.1.6 Performance Benchmarks and Competitive Standards

To evaluate the feasibility of using neural networks for Pan, Tilt and Zoom (PTZ) displacement (Q1), it is necessary to establish a quantitative definition of **competitive performance** based on existing State of the Art (SOTA) methods. As our approach falls within the domain of Relative Pose Regression (RPR) and focuses on domain adaptation rather than surpassing the absolute SOTA on standard benchmarks, the goal is to achieve a competitive performance against established RPR baselines, despite the scarcity of real-world training data. We start by summarizing the existing performance baselines and will conclude in the next subsection with our definition of competitive performance.

Existing Baselines

In the domain of RPR, reported error rates vary significantly depending on dataset complexity and domain shifts. Under standard conditions, methods typically report median rotation errors ranging from a few degrees to tens of degrees. For instance, RelocNet [BLP18] reports median rotational errors between 4.14° and 13.4° on the 7Scenes [SGZ⁺13] and ScanNet [DCS⁺17] datasets, with the majority of scenes falling into the 5° – 11° range. Similarly, RCPNet [YLZ20] demonstrates high precision with errors ranging from 1.24° to 6.14° on the Cambridge Landmarks [KGC15] and Tuebingen Buildings datasets. RelMobNet [RMVH22] maintains this high-performance baseline using a robust two-stage training approach, achieving median rotation errors between 2.93° and 6.30° on Cambridge Landmarks [KGC15].

However, performance across these methods tends to degrade significantly in cross-dataset evaluations or under severe domain shifts. For example, RCPNet reports error spikes up to 28.6° when tested on unseen scenes [YLZ20]. Similarly, RelMobNet sees rotation errors increase to over 21° when subjected to severe style transfer (e.g., “Starry” style) [RMVH22], highlighting the persistent challenge of domain adaptation.

In contrast to pose estimation, standard benchmarks for explicit overlap regression in PTZ scenarios are notably absent from the literature. The specific complexities of this task, continuous camera motion combined with unstructured outdoor environments, mean that, to the best of our knowledge, no widely accepted error threshold exists. Consequently, performance cannot be measured against a fixed literature value. Instead, effectiveness can be evaluated by the model’s ability to generalize across domains and its improvement over naive single-domain baselines.

Definition of Competitive Performance

Establishing a fair performance standard for this thesis must account for the specific constraints that differentiate this research from standard pose estimation tasks:

1. Reliance on virtually created synthetic data with a significant S2R domain gap (P1).

2. Limited availability of real-world training images (P2).
3. Potential ground truth inaccuracies due to mechanical constraints in PTZ camera systems (P2).

Consequently, in the context of this thesis, particularly regarding Research Question (Q1), we define **competitive performance** as:

1. **Rotation:** Achieving median errors falling within the 5° – 10° range. This demonstrates performance that outperforms degraded cross-domain baselines ($> 20^\circ$) and matches standard benchmarks like RelocNet [BLP18], while approaching the precision of modern SOTA architectures such as RCPNet [YLZ20] and RelMobNet [RMVH22] ($< 5^\circ$).
2. **Overlap:** Demonstrating a significant reduction in error compared to the real-world only domain and the naive Dataset-Level Mixing (DLM) baselines due to missing task-related benchmarks (see Section 4.4).

3.2 Deep Domain Adaptation

Building on foundational works that serve as baselines for addressing our question of whether and how we can regress PTZ rotation and overlap estimation (Q1), we now turn our attention to domain adaptation techniques, particularly deep domain adaptation methods, to address the challenges posed by domain gaps (P1). The scarcity of real-world PTZ training data necessitates the use of synthetic alternatives (P2). However, this reliance on synthetic data intensifies the domain adaptation problem, as synthetic data often poses significant distributional shifts from real-world data. This motivates our investigation into whether synthetic data can be primarily used for training our rotation and overlap regressor network (Q2) while still achieving competitive inference estimations despite these challenges.

This challenge is further complicated by implicit domain shifts occurring in real-world applications, even when synthetic data is not involved. When developing a generalized model, it is generally unrealistic to assume that training and test sets share the same distribution or features in real-world applications. There is always some degree of domain shift between the source and the target introduced by several factors such as illumination changes, pose, or image noise. This is particularly relevant for PTZ camera systems, which often operate in dynamic environments with variable lighting or angles, increasing the domain shift. Incorporating synthetic data amplifies this shift even further. Some form of domain adaptation is thus crucial to improve inference performance by transferring the modeled knowledge of a labeled source domain to an unlabeled target domain. Also, during the training of deep neural networks, it is presumed that the source and target domains share the same distribution, however, this assumption often leads to performance degradation caused by real-world domain shifts [WD18, FVRA20].

To cover this transfer of knowledge from source to target domain, different domain adaptation techniques evolved, each having strengths and limitations in addressing the domain gap. In this context, domain adaptation is a specialized form of transfer learning and can be categorized into instance-based, feature-based, and deep-domain approaches. While traditional instance-based and feature-based methods built the foundation for domain adaptation, this section will focus on deep-domain techniques that leverage deep neural networks to learn transferable features across domains, as it is of most relevance for this work. [ZQD⁺20, FVRA20]

Here, in Section 3.2, we will start with a thorough review of deep domain adaptation categories proposed by the literature [WD18, FVRA20], namely:

- Discrepancy-based
- Adversarial-based
- Reconstruction-based

In the following sections we will add and discuss categories relevant to our problem (P1), more specifically Domain Adaptation Regression (DAR) (Section 3.3) and S2R domain adaptation. We are discussing their mechanisms and potential applications in the context of regressing PTZ camera pose and overlap trained on real-world and synthetic PTZ data and argue that there is no perfect single method yet.

Deep domain adaptation (Figure 3.4) leverages the power of deep neural networks to learn transferable representations that can bridge the domain gap between source and target domains. These methods often embed domain adaptation into the deep learning pipeline, enabling the extraction of features that are more abstract and invariant to domain-specific variations. [FVRA20]

3.2.1 Discrepancy-based Domain Adaptation

The first unsupervised deep domain adaptation category is discrepancy-based domain adaptation, which achieves adaptation by aligning either marginal distribution $P(X)$, the conditional distribution $P(Y|X)$, or both, with X representing features and Y the labels. Discrepancy-based domain adaptation methods vary in their approach to distribution alignment. Deep Adaptation Networks (DANs) [LCWJ15] focus solely on aligning marginal distributions, embedded into Reproducing Kernel Hilbert Space (RKHS) [Aro50], a mathematical space where functions behave like points in a geometric space. In this setting, similarity can be measured using kernel functions, allowing complex distributions to be compared in a high-dimensional space without explicitly mapping the data there. DAN uses a multi-kernel variant of Maximum Mean Discrepancy (MMD) [ABR⁺12]. Based on DAN, Joint Adaptation Networks (JANs) [LZWJ17] focus on aligning joint distributions of domain-specific layers across domains using a novel MMD adaptation named Joint Maximum Mean Discrepancy (JMMD). JMMD aligns joint

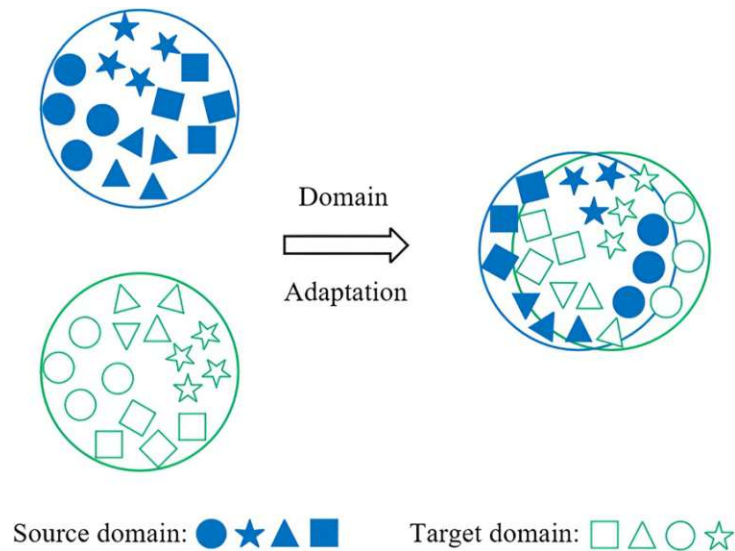


Figure 3.4: Illustration of domain adaptation, showing the initial distribution gap between the source (blue) and target (green) domains, and their alignment through adaptation to enable knowledge transfer. Image source: [HTF⁺23] page 2.

distributions $P(X^1, X^2)$, where X^1 and X^2 are features from different network layers, capturing cross-layer dependencies. Others, such as Deep Transfer Network (DTN) [ZYCW15], combine marginal alignment $P(X)$ with conditional alignment $P(Y|X)$ using pseudo-labels generated during training iteratively via entropy minimization or adversarial training. This dual approach allows for a more comprehensive alignment between source and target domains.

While those methods are effective for classification tasks, they face significant challenges when applied to regression problems such as PTZ camera pose and overlap estimation (P1). These methods are primarily designed for classification tasks and rely on aligning marginal and conditional distributions at a global level, which is insufficient for capturing the precise, continuous relationships required in regression [YRL24]. Moreover, this base category lacks mechanisms to ensure point-wise or geometric consistency. Although techniques such as the Geometric Criterion [WD18] or Domain Interpolation Frameworks [CBG13] can partially address these limitations by embedding spatial relationships or smoothing domain transitions, they still do not fully meet the demands of PTZ pose and overlap estimation, where fine-grained and precise alignment is essential.

3.2.2 Adversarial-based Domain Adaptation

Goodfellow et al. pioneered adversarial training with Generative Adversarial Networks (GANs) [GPAM⁺14] which inspired adversarial-based domain adaptation [GSS15]. In this framework, the feature extractor processes data from both source and target domains, creating shared representations as features. A domain discriminator then attempts

to classify whether these features come from the source or target domain through binary classification. The feature extractor is designed to fool the discriminator by learning features that are indistinguishable between both domains. While Goodfellow introduced the adversarial principle, Ganin et al. [GUA⁺17] specifically adapted this for domain adaptation via gradient flipping in a Gradient Reversal Layer, which basically reverses the gradients from the discriminator during back-propagation and thus trains the feature extractor adversarially. Over time, this adversarial process aligns source and target distributions, creating domain-invariant features that work well for both domains. Adversarial training is also used in hybrid approaches to further enhance techniques like discrepancy-based domain adaptation or reconstruction-based domain adaptation.

Building on this concept, Coupled Generative Adversarial Networks (CoGANs) [LT16] extend adversarial training to multi-domain image generation. They achieve this by learning joint distributions of multi-domain images without requiring paired corresponding images in the training set. The novelty is enforcing a weight-sharing constraint between the generators, which encourages the networks to utilize shared high-level semantic representations across domains. This enables the generation of corresponding image pairs in different domains. A further advancement are Cycle-Consistent Adversarial Networks (CycleGANs) [ZPIE17], which enable unpaired image-to-image translation. CycleGAN learns bidirectional mappings between two domains without the need for paired training examples. It employs a cycle consistency loss, which aims to translate an image to the target domain and back to the source domain while yielding the original image. This cycle consistency loss lead to Cycle-Consistent Adversarial Domain Adaptation (CyCADA) [HTP⁺18] (Figure 3.5), where both pixel and feature level representations are adapted while enforcing semantic consistency.

Another method is Conditional Adversarial Domain Adaptation [LCWJ18], based on Conditional Domain Adversarial Networks (CDANs) [Mir14]. CDAN conditions the domain discriminator on both feature representations and classifier predictions. This approach addresses challenges in multimodal distributions and improves transferability by using multilinear conditioning, which captures complex, multiplicative interactions among multiple features, or entropy-based conditioning, which leverages measures of uncertainty to select more stable, predictable examples, to focus on examples that more easily adapt to diverse scenarios.

Further, Contrastive Unpaired Translation (CUT) [PEZZ20] introduces a contrastive learning approach for unpaired image-to-image translation, replacing the traditional cycle consistency loss of CycleGAN with the PatchNCE loss, incorporating Noise Contrastive Estimation (NCE) [GH10]. This loss encourages the model to learn discriminative features by maximizing the similarity between corresponding patches in the input and output images while minimizing the similarity with the unrelated patches. CUT achieves efficient training and high-quality translations without requiring paired data or cycle consistency by removing the reverse mapping GAN and focusing on patch-level contrastive learning. This method is particularly effective for capturing fine-grained details and preserving important structures during domain adaptation tasks



Figure 3.5: Example of GAN-based domain adaptation via CyCADA. The model transforms an image from GTA V, a computer game, to the real-world Cityscapes [COR⁺16] dataset (top), and performs the reverse transformation from Cityscapes to GTA V style (bottom). Images source: [HTP⁺18] page 6

Summarizing adversarial domain adaptation techniques and reviewing their applicability for solving (P1), they pose potential improvements for the domain adaptation of synthetic PTZ camera data for relative real-world rotation and overlap regression tasks. GANs can reduce the domain gap between synthetic and real-world PTZ data, CycleGANs visually align the images, CDANs cope with diverse scenarios and scenes, and CUT offers a more efficient and flexible approach to unpaired image translation. On the contrary, such adversarial methods focus on global representations and might blur or even change important geometric and spatial features that are needed to perform precise inference of relative rotation and overlap, and thus might degrade the performance, particularly in terms of regression accuracy and robustness for these geometry-sensitive PTZ tasks.

3.2.3 Reconstruction-based Domain Adaptation

This subsection covers unsupervised reconstruction-based domain adaptation and is considered to be pioneered by Ghifary et al. [GKZ⁺16] with their Deep Reconstruction-Classification Network (DRCN). Conceptually, source data is used to train a feature extractor and classifier through labeled data. The same feature extractor is shared and trained on unlabeled target data, where the extracted features are used to train the reconstruction of the target input. This enables the network to learn domain invariant features applicable to both tasks by reconstructing the target data with the classification feature extractor. This helps the model to adapt to the target domain by incorporating structural

information about the target data, even without labels. Boyuan et al. [JCJ20] extend this technique with their Cross-Domain Minimization with Deep Autoencoder (CDMDA) by incorporating adversarial training with an additional discriminator designed to confuse the predicted target labels, making their distribution indistinguishable from a categorical distribution. Shishuai Hu et al. [HLX22] further refine reconstruction-based methods by introducing High-frequency Reconstruction (HFR), which focuses on structural details critical for segmentation tasks while filtering irrelevant low-frequency components. Additionally, they propose Domain-specific Convolution (DSC) to dynamically adapt feature extraction for different domains, improving performance in medical image segmentation.

Reconstruction-based domain adaptation methods offer promising tools for addressing the domain gap in (P1) by learning domain-invariant features, incorporating structural information from the target domain, and dynamically adapting feature extraction across domains. However, they require careful implementation and focus on the challenges of regressing relative rotation and overlap in synthetic PTZ camera data due to differences in texture, lighting, and motion. Additionally, the reconstruction process might emphasize irrelevant features or noise present in either domain.

3.3 Domain Adaptation Regression (DAR)

After summarizing the current SOTA research on general deep domain adaptation techniques, we now focus on two aspects directly tied to PTZ rotation and overlap regression. In this section, we showcase the necessity for specialized methods handling regression tasks and the peculiarities of PTZ rotation and overlap estimation (P1). Subsequently, Section 3.4 addresses Synthetic-To-Real (S2R) adaptation strategies required when including synthetic images to cope with the scarcity of real-world data (P2).

DAR tasks pose distinct technical difficulties compared to classification scenarios, especially when working with PTZ camera systems. Unlike classification models, which rely on clear decision boundaries between categories, regression models work in a continuous space where such distinct separations do not exist. This structural difference causes the need for specialized adaptation approaches for regression-based tasks like PTZ displacement estimation.

The challenge intensifies in PTZ systems due to the growing complexity of keypoint combinations across different camera axes and zoom levels, despite working within finite pixel resolutions. Jiang et al.'s work [JJW⁺21] on DAR demonstrates that conventional adaptation methods struggle with these continuous spaces where traditional feature alignment strategies become less effective.

3.3.1 Techniques

DAR needs specific methods to deal with continuous outputs and the complex variations found in PTZ camera data. The main approaches include:

- Discrepancy-based DAR
- Adversarial-based DAR
- DAR through Multi-Task Learning
- DAR through Multi-Scale Features

Discrepancy-based DAR

Addressing regression challenges requires domain adaptation strategies that can tackle any form of distribution shift, whether due to covariate differences (i.e., unintended variations in data that may influence expected outcomes) or changes in underlying concepts, by effectively reducing discrepancies between domains [CWWL21].

Wu et al. [WHW⁺22] introduce a novel approach specifically designed for DAR through the concept of Distribution-informed Neural Networks (DINO). They define the distribution discrepancy between source and target domains based on the training dynamics of these networks, measuring distribution shifts using MMD in the RKHS induced by the Neural Tangent Kernel (NTK), which characterizes the evolution of infinitely wide neural networks under gradient descent, effectively describing their training dynamics as a linear system. This enables DINO to capture regression-specific subtle distribution discrepancies in continuous output spaces rather than just aligning static feature distributions.

Next, the multi-domain approach for regression is designed to minimize both marginal and conditional shifts, with a particular emphasis on the more difficult conditional shifts, while still handling marginal shifts effectively. Moreover, the Multi-Domain Adaptation for Regression under Conditional shift (DARC) [TNPB23] framework introduces a novel Pairwise Similarity Preserver (PSP) loss that translates differences in target labels into Euclidean distances within a newly constructed feature space, thereby aligning the domains without sacrificing task-specific information. This method is particularly tailored for regression tasks as it directly addresses the continuous nature of target variables by preserving pairwise relationships across domains.

Yang et al [YRL24] address DAR by focusing on theoretical guarantees and a conditional alignment methodology. They establish a generalization upper bound for continuous output scenarios, demonstrating that cross-domain joint error can be effectively bound by conditional discrepancy. To tackle the challenge of conditional alignment in continuous settings, the authors model conditional distributions as finite statistical moments in an RKHS. They propose the Conditional Operator Discrepancy (COD), which relates to the conditional Wasserstein distance in the RKHS, and develop a COD-based method for learning conditional invariant representations to mitigate conditional shifts in latent spaces. The study shows that cross-domain conditional discrepancy is a key factor in generalization error for continuous label variables. Additionally, the COD metric offers a practical approach to optimizing conditional discrepancies in the DAR settings.

Discrepancy-based domain adaptation aims to reduce the S2R domain gap (P1) by aligning the statistical distributions of synthetic and real PTZ data, which can lead to improved regression performance. However, its effectiveness is often constrained by the inherent complexity of real-world PTZ scenarios and the reliance on detailed metadata (P2), which are challenging to reproduce accurately in synthetic datasets. As a result, some residual domain discrepancies may persist.

Adversarial-based DAR

Adversarial-based approaches to DAR aim to reduce distributional discrepancies between source and target domains by leveraging adversarial learning strategies tailored for regression tasks. For example, Jiang et al. [JJW⁺21] observe that prediction errors in the unlabeled domain relate to spatial patterns rather than random distribution. This motivates limiting the output space to a set of K keypoints to bridge the gap between regression and classification approaches. The authors introduce Regressive Domain Adaptation (RegDA), which uses an adversarial regressor to maximize target domain dissimilarity while training a feature generator to minimize it. To guide the adversarial regressor’s optimization, a spatial probability distribution is characterized, thus applying adversarial training in a regression specific context.

Addressing regression domain adaptation under covariate shift, Mathelin et al. [MRD⁺21] tackled both common challenges, namely the negative transfer in unsupervised feature-based methods and the computational intensity of instance-based approaches. They developed the Weighting Adversarial Neural Network (WANN) by introducing the Y-discrepancy [MMM12] as supervised metric between source and target distributions. This approach leverages adversarial weighting to minimize the Y-discrepancy, effectively combining the computational efficiency of adversarial neural networks with the instance-based methods’ ability to handle regression domain adaptation under covariate shift. Their research demonstrated that conventional feature-based methods are inadequate for regression domain adaptation when covariate shift assumptions are present.

In our context, adversarial-based domain adaptation methods are promising because they directly address the distributional gap between synthetic and real PTZ data, which is central to our domain gap problem (P1). However, their effectiveness may be limited if the spatial or covariate shifts in our PTZ data are more complex or structured than those considered in prior work, potentially requiring further adaptation or domain-specific modifications.

DAR through Multi-Task Learning

A valid strategy to cope with DAR is multi-task learning. Fourure et al. [FEF⁺17] applied their approach to the problem of hand pose regression using depth images from two datasets with differing annotations (14 joints vs. 16 finger centers). They trained a shared network architecture with convolutional and fully connected layers on mixed batches of equally distributed samples from both datasets, minimizing the L2 error.

This enabled the model to generalize across domains despite differences in the semantic meaning of the labels.

Other works like the Domain Adaptation for Semantic and Geometric-aware Image-based Localization (DASGIL) framework introduced by Hu et al. [HQC⁺20] leverage a multi-task architecture to integrate geometric and semantic information into multi-scale global representations for retrieval-based localization. The approach employs multi-scale feature discrimination and adversarial training to enhance domain adaptation, addressing the challenges of S2R domain gaps (details of S2R will be covered specifically in Chapter 3.4). DASGIL combines geometric and semantic knowledge to enhance long-term visual localization, leveraging continuous embeddings for precise regression.

For PTZ regression, the success of multi-task learning depends on whether rotation and overlap estimation are based on sufficiently similar features. If these tasks are too dissimilar, multi-task learning may not only fail to bridge the S2R domain gap (P1), but could also introduce negative transfer that degrades regression performance.

DAR through Multi-Scale Features

Regression tasks, unlike classification, are highly sensitive to feature scaling. Traditional domain adaptation methods that align distributions of deep representations may unintentionally alter feature scales, leading to degraded regression performance. To address this, Chen et al. [CWWL21] proposed the Representation Subspace Distance (RSD) method, which avoids scaling issues by representing data in orthogonal bases within representation spaces. These orthogonal bases are not affected by feature scaling. RSD leverages the geometry of the Grassmann manifold to align subspaces between source and target domains. Additionally, they introduced a novel regularizer called Bases Mismatch Penalization (BMP), which ensures similarly ranked bases in each subspace are aligned when calculating distances across subspaces.

Another work is proposed by Xia et al. [XWKA⁺22]. They developed an adversarial bi-regressor architecture to achieve direct alignment between source and target domains for DAR. Their approach maximizes the disagreement between two distinct regressors to identify target samples outside the source domain's support and subsequently learn domain-invariant features. To address significant cross-domain distribution discrepancies, they constructed two intermediate domains between the source and target domains, gradually reducing mismatches to achieve alignment. Experimental results on multi-modal RF signal and image benchmarks demonstrated the effectiveness of this method in solving challenging DAR problems in real-world applications.

Multi-scale feature hierarchies allow a model to capture both broad scene layout and fine details, which are important for precise PTZ rotation and overlap estimation. Using these features from synthetic data can help reduce the domain gap noted in (P1). However, different feature scales are affected by different types of domain shifts. Coarse features often reflect pose or lighting differences, while fine features can exaggerate noise. Aligning all scales at once can increase the variability already present in real-world data. When

only a small amount of annotated real data is available (P2), the model may rely too heavily on synthetic features at certain scales, leading to unbalanced learning and reduced generalisation.

3.3.2 DAR Challenges for PTZ Rotation and Overlap Regression

Domain adaptation for improving relative rotation and overlap estimation in PTZ cameras presents complex challenges due to the specific characteristics of PTZ systems and the continuous nature of regression tasks. Unlike classification with discrete outputs, regression operates in a continuous output space, complicating the use of traditional domain adaptation techniques such as feature alignment. PTZ cameras add further complexity by introducing multiple differing configurations in pan, tilt, zoom, and resolution settings. These variations result in a large number of keypoint combinations that must be aligned across domains, making feature matching and adaptation particularly challenging (P1).

The domain gap between synthetic and real-world data amplifies these issues, as differences in covariate and conditional distributions between the synthetic source and real-world target domains must be addressed effectively. This requires advanced methods capable of handling both marginal input-level and conditional output-level shifts while preserving regression specific relationships. Unlike classification, regression demands precise alignment of continuous outputs, which makes it crucial to adapt models while keeping the fine-grained relationships between input features and their corresponding outputs. Section 3.4 will therefore focus on S2R domain adaptation strategies relevant for (P1), which is further necessitated by (P2).

3.4 Synthetic-To-Real (S2R) Domain Adaptation

A persistent challenge in developing deep learning models for PTZ camera systems is the high cost and complexity of collecting and annotating real-world training data, particularly when precise camera metadata such as intrinsics and extrinsics are required (P2). The process is labor-intensive and expensive, often demanding specialized equipment and repeated deployments across diverse environments [CSVV20, YLZ21, ISB22, AK23]. This scarcity of labeled PTZ data has motivated recent research to generate synthetic PTZ datasets as a scalable and flexible alternative for training [ZRK⁺20, GFW⁺25].

Synthetic data offers a promising solution by providing virtually unlimited simulated data. However, models trained solely on synthetic data often suffer from significant performance drops when applied to real-world tasks, primarily due to the domain gap in visual characteristics such as lighting, textures, and noise between synthetic and real domains (P1).

Furthermore, synthetic datasets tend to be overly structured and lack the imperfections inherent in real-world data [PUK⁺18]. This section explores recent advancements in S2R domain adaptation and discusses their relevance for PTZ regression tasks.



Figure 3.6: Results of visual alignment via CycleGAN. From left to right: real-world image, Synthetic-To-Real (S2R) adapted image, synthetic image, and Real-To-Synthetic (R2S) adapted image. Images source: [YLZ21] page 10

3.4.1 Techniques

Building on these challenges, the following sections focus on three key research directions:

- Visual Alignment
- Transfer Learning
- Methodical Data Blending

These were selected as practical strategies for reducing the S2R domain gap in visual tasks. Visual alignment matches features across domains, transfer learning reuses knowledge from related tasks, and data blending explores how to mix synthetic and real samples best. Together, they support the development of robust computer vision regression models with limited real-world data.

Visual Alignment

Visual alignment aims to reduce the domain gap by making synthetic data visually resemble real-world data (see Figure 3.6). This is often achieved through image-to-image translation, where the style of synthetic images is adapted while preserving their structure. CycleGAN [ZPIE17] is an established method for this task. It enables unpaired image-to-image translation by transforming source images to match the target domain’s appearance while maintaining structural details via cycle-consistency loss. CyCADA [HTP⁺18] extends this idea by aligning both pixel and feature-level representations and enforcing semantic consistency through adversarial training and task-specific losses. This dual adaptation ensures that translated images are both structurally accurate and semantically meaningful, making CyCADA a promising approach for high-level tasks, demonstrated on semantic segmentation and potentially transferable to pose-related problems

Yang et al. [YLZ21] applied a combination of cycle-consistent style transfer and adversarial training to estimate relative camera poses using mixed synthetic-real image pairs. They demonstrate how domain adaptation techniques can be adapted to address tasks like pose regression. A further existing strategy is reverse domain adaptation. Acharya et

al. [AK23] utilize CycleGAN for both S2R and R2S adaptations in PoseNet [KGC15] based localization tasks. Their findings reveal that such a reverse adaptation approach can achieve higher localization accuracy but might introduce ambiguities in areas with similar structural geometry or discrepancies between 3D models and actual scenes.

Another promising approach is contrastive learning for patch-based adaptation. Imbusch et al. [ISB22] employed CUT [PEZZ20] on patch-based images for segmentation tasks in cluttered tabletop scenes virtually generated using Stilleben [SB20]. By focusing on localized features rather than full-resolution images, they effectively reduced the domain gap.

Also in this context, we want to recite Hu et al. [HQC⁺20] and their proposed DASGIL, which bridges the S2R gap through multi-task learning and adversarial training. By integrating geometric and semantic information via multi-scale feature discrimination, DASGIL facilitates knowledge transfer from synthetic to real-world environments without additional human annotations, demonstrating the effectiveness of multi-modal data fusion for domain adaptation in localization tasks.

Additionally, Jin et al. [JR21] proposed a novel technique for generating training data by independent sampling from object spaces (transformed views) and context spaces (challenging backgrounds). This approach creates scenarios that are difficult to classify and thus enables models to learn context-invariant representations, improving robustness against variations in background or scene composition.

Finally, Wu et al. [WLX⁺20] and Luo et al. [LHC25] highlight the importance of leveraging diverse augmentation methods to simulate realistic conditions. In the context of S2R domain adaptation, such techniques are particularly valuable as they help bridge visual or contextual discrepancies between synthetic and real data. By introducing more realistic variations into synthetic data or tailoring augmentations to mimic real-world complexities, these approaches can enhance a model's ability to generalize effectively across domains.

Visual alignment methods are particularly relevant to (P1), as they directly address the domain gap between synthetic and real PTZ data that limits model accuracy. They also help with (P2) by making synthetic data more useful when real-world data is scarce or difficult to obtain. However, as indicated by Acharya et al. [AK23], these approaches may still introduce artifacts or leave subtle discrepancies, so further refinement is needed to ensure robust performance in PTZ applications.

Transfer Learning

Pre-training on large synthetic datasets, followed by fine-tuning on real-world data, has proven effective. Simulators like CARLA [DRC⁺17] can generate virtually unlimited, pixel-accurate annotations across diverse weather, lighting, and viewpoints, reducing expensive manual annotation. This provides the model with diverse examples of spatial relationships, helping to reduce the amount of expensive real-world data needed for training. For example, Charco et al. [CSVV20] showed the benefits of pre-training on images

generated by the CARLA simulator [DRC⁺17]. Later, Charco et al. [CSVV21] highlighted that matching the spatial layout and camera viewpoints of the synthetic scenes to those in the real target environment is crucial for effective adaptation. This alignment allowed their detector to reach target accuracy using up to ten times fewer real-world samples. Similarly, Sagmeister et al. [SSN⁺23] evaluated the impact of domain-aligned synthetic datasets compared to general-purpose ones such as PeopleSansPeople [EEDV⁺22]. Using their domain-aligned *SimulatedCabin* dataset (generated in Unity¹) for pre-training, they achieved a 30.5% improvement in Average Precision (AP) on the DriPE [GCJT21] dataset using 1% of the annotated real data compared to training on the real-world data alone.

Studies have shown that pretrained backbones like ResNet [HZRS16] are widely used in domain adaptation tasks due to their ability to extract robust features that generalize across domains [KWSS22]. For instance, frameworks like PoseNet [KGC15] and Camera Pose Auto-Encoders (PAEs) [SK22] utilize convolutional backbones to encode latent representations for camera pose regression tasks, demonstrating their effectiveness in both absolute and relative pose estimation scenarios.

Leveraging models pretrained on real-world datasets helps mitigate the S2R domain gap by providing a strong initialization for downstream tasks. These models already encode features that are representative of real-world image characteristics, making them effective for adaptation from synthetic data in camera pose regression tasks. Fine-tuning specific layers of the backbone might further enhance the transferability of these pretrained features to reduce domain discrepancies [CSVV20, CSVV21, YLZ21].

Collectively, these transfer-learning strategies address both Problems (P1) (the synthetic-to-real domain gap) and (P2) (the high cost and complexity of acquiring large real-world PTZ datasets) by leveraging large synthetic datasets for pre-training and fine-tuning models on limited real data, thereby improving generalization and reducing real data requirements. However, this approach requires additional training steps or access to suitable pre-trained models, and its effectiveness depends on the similarity between synthetic and real domains, the quality of the pre-trained model, and careful selection of which layers to transfer or fine-tune.

Methodical Data Blending

Another important but less explored technique to improve S2R domain adaptation during training is the way batches are composed. Lee et al. [LPYL20] and Wu et al. [WLX⁺20] use batches made up of an equal mix of synthetic and real data (50% each), which helps the model generalize better across domains. Although this balanced sampling shows promising results, neither study looks into other mixing strategies or different ratios, leaving room for further investigation and optimization.

Recent advancements suggest alternative perspectives on data mixing strategies that could be adapted to S2R domain adaptation. For instance, Yin et al. [YWQX21] introduced

¹<https://unity.com/en>

BatchMixup, a method that enhances training by interpolating hidden states across the entire batch rather than relying solely on pairwise sample combinations. This approach generates synthetic samples distributed throughout the representation space of the batch, potentially improving model robustness across diverse domains. Although originally applied in Natural Language Processing (NLP) tasks, its principles could be extended to S2R settings by exploring how interpolating synthetic and real data representations at the batch-level might mitigate domain gaps and improve generalization to real-world scenarios.

Furthermore, Furqon et al. [FPL⁺24] proposed a Mixup-based domain adaptation technique for dynamic Remaining Useful Life (RUL) predictions, which estimate the time left before a system or component fails or requires maintenance. This work emphasizes the importance of balancing synthetic and real data during training to address domain shift challenges. While Yin et al.'s BatchMixup [YWQX21] does not explicitly involve dynamically varying mixing ratios, it demonstrates how advanced interpolation techniques can optimize performance by better covering the representation space. These ideas can be translated to S2R domain adaptation by progressively adjusting the ratio of synthetic to real data during training. Such an approach could enable models to increasingly rely on real-world features as training progresses, potentially improving alignment between synthetic and real domains. However, further research is needed to validate this hypothesis and explore its practical implications.

The domain gap introduced by synthetic PTZ data negatively impacts the performance of neural networks in regressing relative rotation and overlap for real-world image pairs (P1). Addressing this gap requires a systematic evaluation of domain adaptation strategies, including the optimal balance and mixing of synthetic and real-world data. Recent theoretical advancements, such as the Golden Ratio Mixing framework [HXC25] (currently in preprint), propose an optimal weighting scheme for integrating real and synthetic data during training. This framework suggests that the reciprocal of the golden ratio may represent an ideal balance for stabilizing model performance by effectively managing the trade-off between leveraging synthetic data and maintaining generalization capabilities. While primarily developed to address challenges in generative model training, its principles could inspire novel approaches to mitigate domain gaps in S2R tasks.

3.4.2 S2R Challenges for PTZ Rotation and Overlap Regression

The domain gap between synthetic and real-world PTZ data introduces several challenges for regressing relative rotation and overlap (P1). Visual discrepancies (see Figure 3.7), such as differences in lighting, textures, and noise, often hinder the transferability of models trained on synthetic data to real-world applications. Furthermore, synthetic datasets tend to simplify scene structures, failing to capture the intricate geometry and contextual variability present in real-world environments. Compounding this issue is the scarcity of annotated real-world PTZ data (P2), as collecting such datasets requires specialized equipment and significant effort to gather metadata like camera intrinsics and extrinsics.



Figure 3.7: Comparison of synthetic data (left) and comparable real-world scenes (right), highlighting simplified structure, geometry, and variability in the synthetic data. Image source of real-world pictures: [Cle12, Wor25]

To address these issues, researchers have explored a variety of promising solutions. Visual alignment techniques, such as CycleGAN and CyCADA, transform synthetic images to better resemble real-world styles while preserving structural integrity. These methods have shown success in reducing domain gaps for tasks like pose regression. Transfer learning offers another effective approach by leveraging pretrained models on large-scale real-world datasets, which provide robust feature representations that can be fine-tuned for specific tasks. Additionally, methodical data blending strategies, such as balanced batch sampling or advanced interpolation methods like BatchMixup, have demonstrated potential in improving model generalization by combining synthetic and real-world data in optimal proportions.

Despite these advancements, opportunities remain for further research. Investigating dynamic mixing strategies during training or exploring the relationship between limited real-world data and its synthetic counterpart could yield valuable insights. By addressing these challenges through innovative domain adaptation techniques, researchers can enhance the accuracy and robustness of PTZ rotation and overlap regression models while minimizing reliance on extensive real-world datasets.

3.5 Summary and Research Gaps

To sum up, there are four main ways to approach the problem of PTZ rotation regression:

- Key-Point Detection and Feature Matching (KDFM)
- Scene Coordinate Regression (SCR)
- Absolute Pose Regression (APR)
- Relative Pose Regression (RPR)

Among these, RPR emerges as the most promising, owing to its strong potential to generalize across unseen scenes and its effectiveness in regressing relative PTZ rotation.

In addition to rotation regression, overlap estimation plays an important role in PTZ camera applications by ensuring sufficiently shared features between image pairs. Geometry-based methods like NSO [RGHS⁺20], transformer-based architectures such as COTR [JTH⁺21], and photogrammetry techniques [DUK22] have shown potential in estimating overlaps efficiently. However, these approaches are not explicitly designed for S2R domain adaptation or PTZ specific challenges.

We further examine several domain adaptation techniques designed to close the domain gap of models trained on simulated synthetic data when used on real-world PTZ images. Discrepancy-based methods are discussed with particular emphasis on their ability to align marginal and conditional distributions, such as DTN [ZYCW15], but these approaches are typically designed for classification tasks and underperform when applied directly to regression problems, which require a fine-grained alignment of continuous outputs. Adversarial-based methods like GANs [GPAM⁺14], CycleGANs [ZPIE17], and conditional approaches such as CDANs [Mir14], are also reviewed for their applicability in aligning global representations. Their focus on overall statistics might be problematic as we unintentionally compromise the preservation of subtle geometric and spatial details which are critical for reliable PTZ regression. Additionally, reconstruction-based techniques, like DRCN [GKZ⁺16], are capable of learning domain-invariant features, which is promising but risks emphasizing irrelevant details or noise.

We further explore how synthetic data can mitigate the scarcity of annotated real-world PTZ data. While synthetic data provides a nearly limitless source of training examples, it introduces significant domain gaps due to differences in visual characteristics such as lighting, texture, and noise. To counteract these challenges, we have investigated the importance of methodical mixing strategies, like Batch-Level Mixing (BLM) and Dataset-Level Mixing (DLM). A review of the current literature suggests that advanced interpolation techniques, such as BatchMixup [YWQX21], seem promising for improving the simultaneous integration of synthetic and real-world data.

Despite significant progress in these areas, several research gaps exist. First, there appears to be a need for domain adaptation techniques specifically tailored to regression tasks in the context of PTZ cameras. Unlike classification problems, regression tasks require methods that can capture and preserve the fine-grained relationships between input features and continuous output variables, such as relative rotation and overlap.

Second, while synthetic data serves as a valuable resource for addressing the scarcity of annotated real-world PTZ data, optimal strategies for seamless integration of synthetic and real-world data remain underexplored. Approaches such as dynamic mixing ratios or progressive adaptation toward real-world features are promising, but further investigation is required.

Addressing these research gaps shows potential for enhancing the accuracy and reliability of PTZ displacement regression models. By designing domain adaptation techniques specifically tailored to regression challenges and refining strategies for effectively leveraging synthetic data, as explored here and further extendable in subsequent research, neural networks can develop robust domain generalization while delivering precise and consistent predictions for continuous output variables.

We present two tables evaluating the recent literature. We highlight strengths and limitations in Table 3.1 and provide a qualitative comparison of how well those approaches address the challenges of the domain gap issue of synthetic PTZ regression (P1) and real-world PTZ data scarcity (P2) in Table 3.1.

3.6 Contributions

Building upon the foundational Siamese architecture of RCPNet [YLZ20] and the synthetic data motivations of DeepPTZ [ZRK⁺20] (which addresses data scarcity via panoramic stitching), this thesis addresses the specific challenges of the S2R domain gap in PTZ urban monitoring and traffic management. Unlike previous approaches that rely on fixed mixing ratios [LPYL20, WLX⁺20] or purely sequential training phases [CSVV20, CSVV21, SSN⁺23], we propose a dynamic integration of domains. Our main contributions are:

- **Analysis of the S2R domain gap in PTZ scenarios:** We provide a detailed analysis of the S2R domain gap through a systematic matrix-based cross-comparison study. By evaluating inference performance across isolated synthetic and real-world datasets against various R2S ratios, we quantify the specific performance impact of domain shifts and identify the best-performing configurations for relative PTZ pose and overlap regression (C1).
- **Optimization of BLM Strategies:** We investigate simultaneous training strategies where the synthetic surplus is included continuously and strategically alongside real-world data. Distinct from sequential transfer learning approaches like those of Charco et al. [CSVV20, CSVV21], which separate training into distinct phases, our work focuses on optimizing the mixing process itself. Related work by Sagmeister et al. [SSN⁺23] similarly highlights the importance of domain-aligned synthetic data for reducing the S2R gap. We extend the fixed-ratio strategies of Lee et al. [LPYL20] and Wu et al. [WLX⁺20], demonstrating that a synthetic-heavy ratio (R1:S3) notably outperforms the balanced R1:S1 baseline, with exceptions limited to overlap estimation tasks at higher real-world sample sizes (C2).

- **Quantitative analysis of synthetic data efficacy under scarcity:** We analyze how synthetic data compensates for the high cost of real-world PTZ annotation, a motivation we share with DeepPTZ [ZRK⁺20]. This analysis confirms efficacy trends observed in the adjacent human pose domain [SSN⁺23]. Through a systematic variation of the R2S data ratio and real-world sample size, we demonstrate that BLM strategies consistently outperform isolated training and DLM baselines, particularly when real-world samples are limited. We further distinguish our analysis by showing that this advantage persists even as real-world data volume increases. This highlights that while returns diminish, optimal mixing ratios evolve toward a balanced R1:S1 configuration for specific tasks like overlap estimation (C3).
- **PTZNet architecture and implementation:** We introduce PTZNet, a Siamese deep-learning model for joint relative PTZ rotation and overlap regression. While adopting the successful end-to-end regression approach of RCPNet [YLZ20], our design utilizes a deeper ResNet-50 [HZRS16] backbone and is specifically optimized for training on methodically balanced, hybrid datasets of synthetic and real-world images (C4).

Category	Strengths	Limitations
Key-Point Detection and Feature Matching	Robust feature extraction under specific conditions	Performance degrades with distortion, compression, or viewpoint changes
Scene Coordinate Regression	High precision with known environments	Poor generalization to unseen environments
Absolute Pose Regression	Directly regresses 6-DOF poses	Not applicable for relative rotation/overlap estimation
Relative Pose Regression	Generalizes well to unseen scenes	Requires large-scale training data
Overlap Estimation	Improves feature matching; enables efficient clustering of camera FoVs	Lacks direct S2R domain adaptation application; limited PTZ-specific solutions
Discrepancy-Based Domain Adaptation	Aligns marginal/conditional distributions	Ineffective for regression tasks requiring fine-grained alignment
Adversarial-Based Domain Adaptation	Generates domain-invariant features	May distort geometric/spatial features critical for PTZ tasks
Reconstruction-Based Domain Adaptation	Learns invariant features while incorporating structural information	Risk of emphasizing irrelevant details
Synthetic-To-Real	Provides virtually unlimited labeled training data	Significant domain gap due to visual discrepancies
Batch-Level Mixing Strategies	Improves generalization by combining synthetic and real data	Optimal mixing ratios remain unexplored

Table 3.1: Related Work Comparison: Strengths and Limitations.

While prior approaches offer valuable insights, they often struggle with the S2R domain gap or require large, annotated real-world datasets. As summarized in Table 3.1, existing methods exhibit several limitations, particularly regarding generalization and the lack of

3. RELATED WORK

PTZ-specific S2R adaptation. In this thesis, we focus on the aspects of these limitations that are most relevant for PTZ regression by analyzing the S2R domain gap in detail and by introducing BLM strategies that combine synthetic and real data during training. These efforts aim to mitigate the identified weaknesses and improve the generalization of PTZ regression models under mixed-domain conditions, as outlined in Contributions (C1) and (C2).

Category	Relevance to (P1)	Relevance to (P2)
Key-Point Detection & Matching	Limited: Not robust enough for dynamic PTZ settings	Limited: Requires diverse real-world datasets
Scene Coordinate Regression	Limited: Not suitable for relative pose estimation	Limited: Relies on annotated real-world data
Absolute Pose Regression	Not relevant: Focuses on absolute poses	Not relevant: Requires extensive labeled datasets
Relative Pose Regression	High: Suitable for relative rotation/overlap tasks	Medium: Needs synthetic-real integration
Overlap Estimation	High: Ensures shared features for pose regression	Medium: Needs integration with synthetic-real data
Discrepancy-Based Domain Adaptation	Medium: Can reduce domain gap partially	Medium: Needs task-specific tuning
Adversarial-Based Domain Adaptation	Medium: Reduces visual domain gap	Medium: Needs careful design to preserve features
Reconstruction-Based Domain Adaptation	Medium: Promising but needs careful implementation	Medium: May help with limited real-world data
Synthetic-To-Real	High: Essential for closing the S2R domain gap	High: Addresses limited availability of real-world data
Batch-Level Mixing Strategies	High: Potentially effective for bridging Synthetic-To-Real gaps	High: Potentially effective to integrate synthetic data

Table 3.2: Related Work Comparison: Relevance to (P1) and (P2).

As shown in Table 3.2, most existing methods only partially address the challenges of domain gap (P1) and real-world data scarcity (P2). Our approach seeks to systematically evaluate the effect of synthetic data on real-world inference as the amount of real data varies, and introduces a deep learning model trained on a balanced mix of synthetic and real-world data. These advances contribute towards addressing both (P1) and (P2), with Contributions (C3) and (C4).

CHAPTER 4

Method

This research is designed to address the challenges posed by the domain gap between synthetic and real-world Pan, Tilt and Zoom (PTZ) data (P1) and the complexity of acquiring extensive real-world datasets with detailed metadata (P2). These challenges are central to the research questions, which aim to determine to what extent neural networks can predict relative PTZ displacement and overlap using image pairs (Q1) and whether synthetic data can be leveraged effectively for training (Q2). To address these questions, we propose our four main contributions (C1), (C2), (C3), and (C4) that structure our approach.

This chapter is organized into four sections, each corresponding to key stages of our method and aligned with these contributions. While the contributions are introduced in conceptual order (C1–C4), the chapter structure follows the logical progression of the method, starting from model design, followed by metrics for model training and evaluation, the varying data mixing techniques, and finally the quantitative domain adaptation analysis:

(C4) **PTZ Rotation and Overlap Model:** Defining the baseline deep learning model used in our domain adaptation study, trained on varying ratios of real and synthetic PTZ samples (Section 4.1).

(C1) **Training and Evaluation Metrics:** Defining the quantitative criteria and metrics used to assess model performance, including displacement error and overlap accuracy (Section 4.2).

(C2) **Data Mixing Strategies:** Detailing the dataset preparation process for dataset-level compared to our proposed Batch-Level Mixing (BLM) strategies for synthetic and real data (Section 4.3).

(C3) **Quantitative Matrix Analysis:** Outlining the experimental framework for systematically comparing results across different data mixes and real-world image amounts using the defined metrics (Section 4.4).

4.1 PTZNet: Siamese Network Design for PTZ Estimation

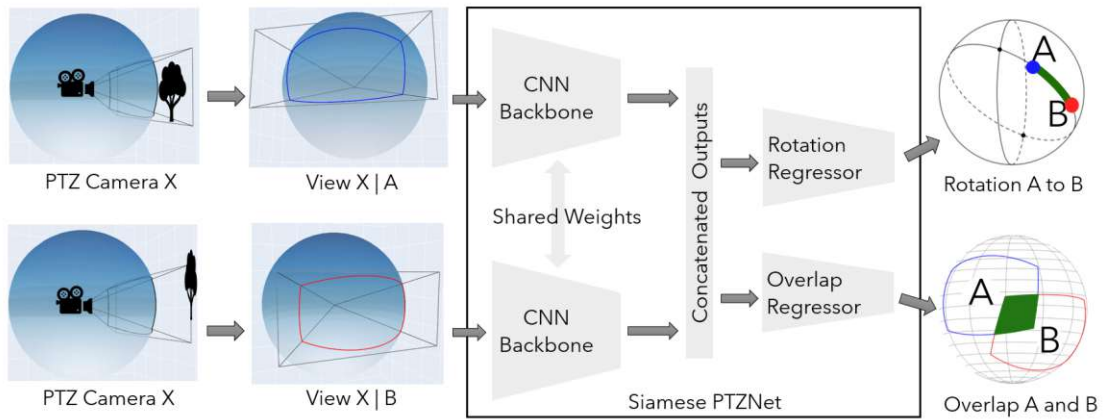


Figure 4.1: Overview of the Siamese PTZNet for comparing relative FoVs of a PTZ camera X. Two different PTZ camera settings (A and B) generate respective views projected on a sphere. Each view is processed by a CNN backbone with shared weights. Their extracted features are concatenated. The network includes a rotation regressor (for estimating the rotation needed to align one view with the other) and an overlap regressor (for predicting the overlap between both FoVs). The rotation from A (blue) to B (red) is depicted as the green great circle arc. The overlap of FoV A (blue) and FoV B (red) is marked as the green Intersection over Union (IoU) on the sphere.

As a foundation for this work, we introduce PTZNet (C4), a model that estimates both relative PTZ displacement and spherical FoV overlap from two input images. This directly addresses research question (Q1), and forms the baseline for exploring the use of synthetic data in training (Q2). The model is reused throughout all subsequent domain adaptation experiments and is central to measuring cross-domain generalization.

PTZNet follows a Siamese Neural Network (SNN) design (Section 2.1.3) commonly used in Relative Pose Regression (RPR) tasks (Section 3.1.4), as visualized in Figure 4.1. Each branch processes one PTZ image using shared weights to ensure consistent feature extraction. As such so-called backbone, we adopt a Convolutional Neural Network (CNN) (Section 2.1.3) pretrained on ImageNet [DDS⁺09], replacing the ResNet-34 [HZRS16] backbone from RCPNet [YLZ20] with ResNet-50 to improve generalization and efficiency [LKB⁺23]. After removing the final classification layer, the global features from both branches are concatenated and passed to two dedicated regressors: one for estimating the relative rotation, the other for the spherical FoV overlap. The shared backbone ensures consistent feature encoding, while the separate heads focus on the

distinct geometric targets of interest. The training objective and evaluation metrics for PTZNet are described in detail in the following section.

4.2 Training and Evaluation Metrics

We continue by defining robust metrics critical for quantifying the performance in regressing PTZ displacement and overlap (Q1) of our PTZNet model 4.1, as illustrated in the composite Figure 4.1. These metrics allow us to cross-compare different evaluation configurations and systematically address the S2R domain gap by optimizing model accuracy under varying data conditions (C1). This gap refers to the performance difference observed when models trained solely on, or with the help of, synthetic data are evaluated on real-world scenarios. This approach enables us to determine which mixing strategy, across synthetic, real-world, and overall mixed datasets (Q2.1), allows the proposed neural network to generalize best to real-world data, and to quantify the impact of BLM versus DLM strategies (Q2.2) as the amount of real-world data varies (Q2.3).

We define a customized training loss function that separately addresses rotation and overlap, and integrates them into a combined loss. In addition, we introduce absolute and relative error metrics to evaluate the accuracy of rotation and overlap predictions across the different experiments.

4.2.1 Training Loss

Artificial Neural Networks (ANNs) learn by actively minimizing a loss function during backpropagation. Thus, we need to define a sound and robust loss function capable of quantifying rotation and overlap estimation performance. To do so, we propose a combined loss function aiming for a balanced training of PTZ rotation and overlap. We first introduce the two distinct loss metrics and how they are combined.

Rotation Loss

A rotation in three dimensions can be expressed as a four-dimensional vector $\mathbf{q} \in \mathbb{R}^4$. Following the definitions reviewed in Section 2.2.3 [Vic01, Ebe02, Tri09], it consists of a scalar part w and a vector part $\mathbf{v} = [x, y, z]^T$. Equivalently, this can be written in the classical Hamilton form $q \in \mathbb{H}$ as shown in Equation 4.1, where i, j, k denote the quaternion imaginary units. We define our rotation loss (Equation 4.2) to minimize the cosine of half the angular difference θ between predicted and target quaternions (Equation 4.3).

$$q = w + xi + yj + zk \quad (4.1)$$

$$L_R(q_{pred}, q_{target}) = 1 - |q_{pred} \cdot q_{target}| \quad (4.2)$$

$$|q_{pred} \cdot q_{target}| = \cos\left(\frac{\theta}{2}\right) \quad (4.3)$$

This loss handles the ambiguity between q and $-q$ and incorporates angular rotation. It grows monotonically from 0 at $\theta = 0^\circ$ to 1 at $\theta = 180^\circ$ (larger angles wrap back into this range because q and $-q$ encode the same rotation).

It does not overly suppress large errors and provides a smooth gradient. Experiments with a squared version did not improve the results. Moreover, this simple but effective loss aligns with proposals of the recent literature in camera pose regression, which often aim for a mix of simplicity and geometric accuracy in loss design [KC17, ZRK⁺20, RMVH22].

Overlap Loss

The overlap o between the relative PTZ FoV areas A and B is computed using their Intersection over Union (IoU) on the surface of the union sphere (Figure 4.2 and Equation 4.4, see Section 2.2.4 for in-depth details). During training, we apply the Mean Squared Error (MSE) loss (Equation 4.5) to this spherical IoU_S, which yielded faster and more stable convergence compared to using the Mean Absolute Error (MAE) loss (Equation 4.6) [TCERP⁺24].

$$o(A, B) = \text{IoU}_S(A, B) = \frac{\text{FoV Intersection}(A, B)}{\text{FoV Union}(A, B)} = \frac{|A_S \cap B_S|}{|A_S \cup B_S|} \quad (4.4)$$

$$L_O(o_{pred}, o_{target}) = \text{MSE}(o_{pred}, o_{target}) = \frac{1}{n} \sum_{i=1}^n (o_{pred} - o_{target})^2 \quad (4.5)$$

$$\text{MAE}(o_{pred}, o_{target}) = \frac{1}{n} \sum_{i=1}^n |o_{pred} - o_{target}| \quad (4.6)$$

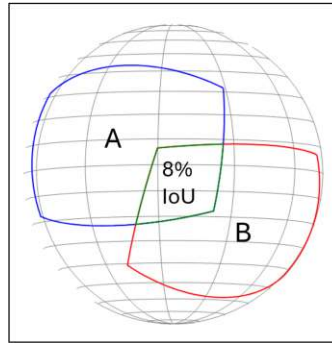


Figure 4.2: Spherical IoU calculation of two overlapping FoVs, A (blue) and B (red), visualized on the unit sphere (Equation 4.4). The overlapping area of 8% is marked in green.

Combined Loss

Our approach to handling multiple loss functions combines the rotation loss and the overlap loss into a single objective function (Equation 4.7). The overall loss L is formulated

as the sum of the rotation loss L_R and the overlap loss L_O , scaled by a balance factor β . This factor was derived to equalize the magnitudes of both loss components during the first training epoch. Throughout the experiments, this β is kept as a fixed value (see Section 6.3, Equation 6.3).

$$L = L_R + \beta \cdot L_O \quad (4.7)$$

This design ensures that both losses contribute to the learning process in a pre-defined proportion. To enhance learning effectiveness, we impose a constraint whereby the rotation loss is suppressed (set to zero) for any relative image pair that overlaps by less than 30%. This threshold was manually tuned, as it yielded the highest rotation accuracy for both small and large rotations in initial trials. This selective gradient removal guarantees that only pairs with a minimum set of shared features influence the rotation regressor, while the overlap loss remains active for every pair. During backpropagation, only the gradients of the active losses are applied. Backpropagation is the phase that propagates the loss backward through the network, computes parameter gradients, and updates the weights and biases to minimize the loss.

4.2.2 Absolute Error Metrics

We report absolute and relative error metrics for rotation and overlap to transparently evaluate predictions on relative PTZ image pairs. In contrast to the training loss, which is mainly intended for optimization and often hard to interpret, these metrics provide a clearer picture of how well the model performs in practice.

Absolute errors reflect the overall deviation between predicted and target values, regardless of scale. This gives a straightforward sense of accuracy. However, absolute errors alone can be misleading, especially for small rotations or overlaps. To address this, we also report relative errors, which normalize the deviation by the size of the target. This makes it easier to assess how precise the model is, particularly when targets are small. By separating the two, we can capture both general performance and accuracy across varying scales.

Absolute Rotation Error $AE_{R,geo}$

To quantify the absolute rotation error $AE_{R,geo}$, we use the geodesic distance (also called great circle arc - Figure 4.3) between two unit quaternions, as proposed by Chen et al. [CSM21] and defined in Equation 4.8. This distance corresponds to the shortest arc on the 4D unit hypersphere between q_1 and q_2 , projected to the rotation space $SO(3)$. Based on the quaternion rotation parameterization, we can derive this distance via the properties of Equation 2.24.

$$AE_{R,geo}(q_{pred}, q_{target}) = 2\arccos(|q_{pred} \cdot q_{target}|) \quad (4.8)$$

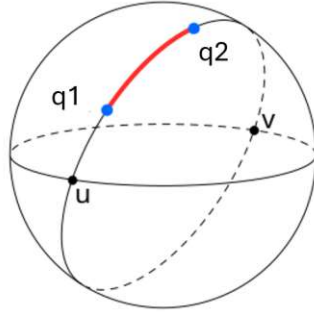


Figure 4.3: Illustration of the geodesic distance (red arc) between two quaternions q_1 and q_2 on the unit sphere, visualizing Equation 4.8. It also shows two antipodal points u and v . Image from [Che16].

Absolute Overlap Error $AE_{O,mae}$

Similarly, to assess the network’s performance in our simulations for overlap estimation, we must define an absolute error metric. For simplicity and due to common usage in deep learning [TCERP⁺24], we calculate and report the Mean Absolute Error (MAE) (Equation 4.6), applying MAE metric to our overlap predictions in Equation 4.9.

$$AE_{O,mae}(o_{pred}, o_{target}) = \text{MAE}(o_{pred}, o_{target}) \quad (4.9)$$

4.2.3 Relative Error Metrics

Next, although relative error metrics are rarely reported in camera pose estimation studies, we introduce a normalized relative error metric RE_{task} . Its use is to capture the magnitude of the absolute error x_{error} between the target x_{target} and predicted x_{pred} values, for both the rotation and overlap regression tasks. This metric, bounded between 0 and 1, is defined in Equation 4.10. The rationale behind reporting this error is to highlight the model’s ability to make precise predictions, with the error normalized relative to the target value. For numerical stability, a small ϵ is added.

$$RE_{task}(x_{pred}, x_{target}) = \frac{|x_{error}|}{|x_{error}| + |x_{target}| + \epsilon} \quad (4.10)$$

Since $|x_{error}|$ and $|x_{target}|$ are non-negative and $\epsilon > 0$, the denominator strictly exceeds the numerator, bounding the metric to the interval $[0, 1[$.

While absolute errors indicate the raw difference between prediction and target, they can be misleading when target values vary significantly in scale. An identical absolute error may be negligible for large targets but critical for small ones. By normalizing errors with the target value, relative error metrics provide a balanced and meaningful metric of model performance across different scenarios. This is relevant for our data, which

contains varying magnitudes of rotations and overlaps. Consequently, the relative error complements absolute metrics by assessing performance consistency independent of the target scale.

Relative Rotation Error $RE_{R,\theta}$

The standard geodesic distance based on the dot product $AE_{R,geo}(q_{pred}, q_{target})$, Equation 4.8, forces a zero target distance, making it unsuitable for measuring relative error as it would yield a zero scaling factor. To address this, we instead compute $RE_{R,\theta}$ (Equation 4.11),

$$RE_{R,\theta} = \frac{|\theta_{error}|}{|\theta_{error}| + |\theta_{target}| + \epsilon} \quad (4.11)$$

using the relative quaternion $q_{rel} = q_{pred}\bar{q}_{target}$, which represents the rotation needed to align q_{target} with q_{pred} . Since q_{rel} is a unit quaternion (see Equation 2.18), we extract its corresponding rotation angle θ using Equation 4.12 (based on Equation 2.19).

$$\theta(q) = 2 \arccos(|\text{scalar}(q)|) = 2 \arcsin(\|\text{vector}(q)\|), \quad \theta \in [0, \pi] \quad (4.12)$$

This angle, denoted as θ_{error} , together with the reference angle θ_{target} (Equation 4.13), serve as inputs to our relative error RE_{task} (Equation 4.10).

$$\begin{aligned} \theta_{error} &= \theta(q_{rel}) = \theta(q_{pred}\bar{q}_{target}) \\ \theta_{target} &= \theta(q_{target}) \end{aligned} \quad (4.13)$$

This additional definition yields a meaningful angular error metric relative to the target rotation (with $\theta_{target} \neq 0$, except in the case of a perfect prediction or identity rotation), and it normalizes the deviation with respect to the target magnitude.

Relative Overlap Error RE_O

To get the relative overlap error RE_O (Equation 4.14),

$$RE_O = \frac{|o_{error}|}{|o_{error}| + |o_{target}| + \epsilon} \quad (4.14)$$

we apply our RE_{task} Equation 4.10 with the overlap target o_{target} and prediction o_{pred} . Since overlaps represent IoU ratios, the values o_{target}, o_{pred} are bounded to the range $[0,1]$. The error term is defined by Equation 4.15.

$$o_{error} = |o_{pred} - o_{target}| \quad (4.15)$$

Combined Relative Error $RE_{R|O,comb}$

To summarize performance across both tasks, we report the combined relative error $RE_{R|O,comb}$ (Equation 4.16). It is defined as the arithmetic mean of the relative rotation $RE_{R,\theta}$ (Equation 4.11) and overlap RE_O (Equation 4.14) errors:

$$RE_{R|O,comb} = \frac{RE_R + RE_O}{2} \quad (4.16)$$

Since both metrics are relative percentages, they share the same scale and can be averaged directly.

Relative Error Visualized

While the relative error metric applies to both rotation ($RE_{R,\theta}$) and overlap (RE_O), we illustrate its normalization behavior using the rotational case. To demonstrate this effect, we assume a fixed absolute deviation of $\theta_{error} = 5^\circ$. Figure 4.4 and Figure 4.5 visualize the resulting $RE_{R,\theta}$ values across a range of target rotations. By holding the absolute error constant, these examples highlight the metric's dependency on the target magnitude: The relative error is large when the target rotation is small, but decreases significantly as the target rotation increases.

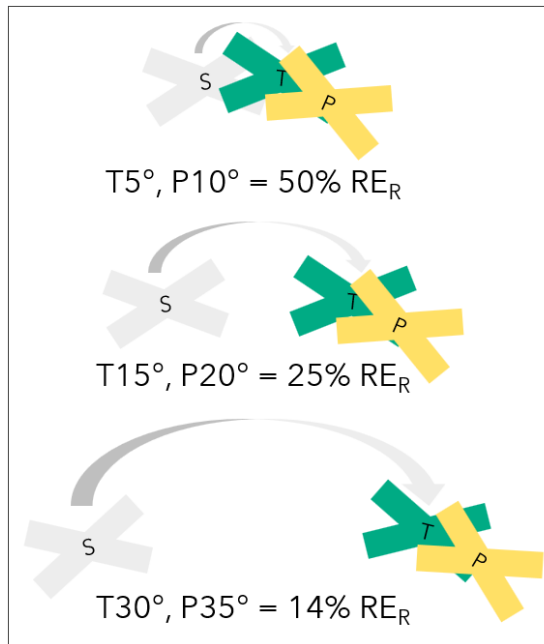


Figure 4.4: Schematic visualization of the relative rotation error $RE_{R,\theta}$ from S to T as symbolic FoV rotation, with $T = \theta_{target}$ and $P = \theta_{pred}$, highlighting how the same fixed 5° absolute deviation results in decreasing relative error magnitudes as the overall target rotation increases.

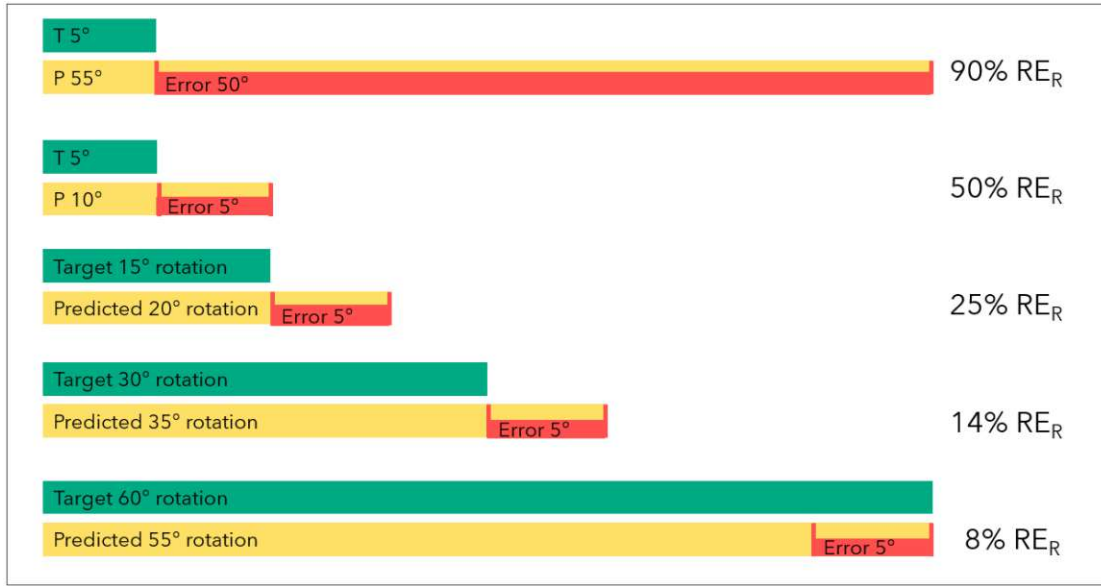


Figure 4.5: Schematic visualization of the relative rotation error $RE_{R,\theta}$ from S to T as error bars, with $T = \theta_{target}$ (green) and $P = \theta_{pred}$ (yellow) as the predicted rotation. The absolute error is shown in red. The top row illustrates a large absolute error of 50° , resulting in a 90% $RE_{R,\theta}$, to show the upper bound of the metric. In contrast, the remaining rows fix the absolute angular error at 5° , while increasing the target rotation, demonstrating how the same absolute deviation yields decreasing relative errors, from 50% down to 8%, as θ_{target} grows.

4.3 Real-World and Synthetic Data Mixing Strategies

With the training and evaluation metrics defined, we now describe how we assess the domain gap introduced by virtually generated synthetic data in PTZ camera systems (P1). A key part of addressing whether we can primarily rely on synthetic data for training (Q2) is how we combine real-world and synthetic data. Given the limitations of real-world data collection (P2), this study explores strategies for integrating synthetic data effectively into training pipelines. Specifically, it investigates Batch-Level Mixing (BLM) approaches as the main method to increase inference performance on real-world datasets (C2). This aligns with the hypothesis that combining synthetic and real-world data enhances model robustness while minimizing reliance on costly real-world data acquisition. Researching the related work in this field (see Section 3.5) revealed a potential gap covering such methodical mixing strategies.

4.3.1 Batch-Level Mixing (BLM)

As outlined in Section 3.4.1, existing literature on BLM strategies remains scarce. In the few instances where such mixing is investigated, such as in the works of Lee et al. [LPYL20] and Wu et al. [WLX⁺20], the distribution is typically fixed at a balanced

50% split between real-world and synthetic data. Consequently, this default configuration is rarely questioned or systematically evaluated for its effectiveness. We also highlighted the potential of BLM to help bridge the S2R domain gap.

To address this, we propose a BLM strategy in which the ratio of real-world to synthetic data can be adjusted within each batch during training. This approach enables a systematic evaluation of various mixing ratios and allows us to investigate whether a particular configuration yields improved performance on real-world evaluation data. Specifically, it lets us test whether the commonly adopted 50% ratio as described above is indeed optimal for the tasks of PTZ pose and overlap estimation, or if this regression scenario benefits from an alternative mixing scheme.

Furthermore, this controlled mixing strategy allows for a direct comparison between models trained with different mixing ratios and those trained solely on either real-world or synthetic data. Given the limited availability of real-world data (P2), this comparison helps quantify how well the model can learn from synthetic data alone. It also supports an investigation into whether leveraging large-scale synthetic datasets improves performance (Q2), or whether the synthetic-to-real domain gap (P1) negatively impacts generalization to real-world scenarios, potentially leading to the model trained purely on real-world data outperforming the others.

The concept of the proposed BLM strategy is illustrated in Figure 4.6. We independently and continuously iterate over both the synthetic and real-world datasets. In this setup, we utilize the full set of available synthetic images, given their low cost of generation, alongside a limited real-world dataset containing n images, constrained by PTZ data scarcity (P2). During each training step, samples from both datasets are mixed at the batch-level according to a predefined R2S ratio, which specifies the desired proportion of R2S data for the current configuration. If either dataset iterator is exhausted during training, it is reshuffled and restarted to maintain an uninterrupted and balanced sampling process.

A direct consequence of this design is that, given a small set of real-world images compared to a large pool of synthetic ones, the network will encounter the same real-world samples significantly more often than unique synthetic ones, depending on the selected ratio. This recurrence is particularly relevant to our study, as it allows us to investigate whether repeated exposure to a small set of real-world data can outperform training on large-scale synthetic datasets, contributing to our broader evaluation of the trade-off between synthetic data volume and real-world generalization (Q2), and the challenges posed by the synthetic-to-real domain gap (P1).

4.3.2 Dataset-Level Mixing (DLM)

As a baseline for comparison, we also evaluate a simpler DLM strategy, visualized in Figure 4.7. In this approach, the synthetic and real-world datasets, each containing n images, are combined into a single, combined dataset before the training. The technique iterates over this merged dataset in a standard fashion, sampling uniformly from the

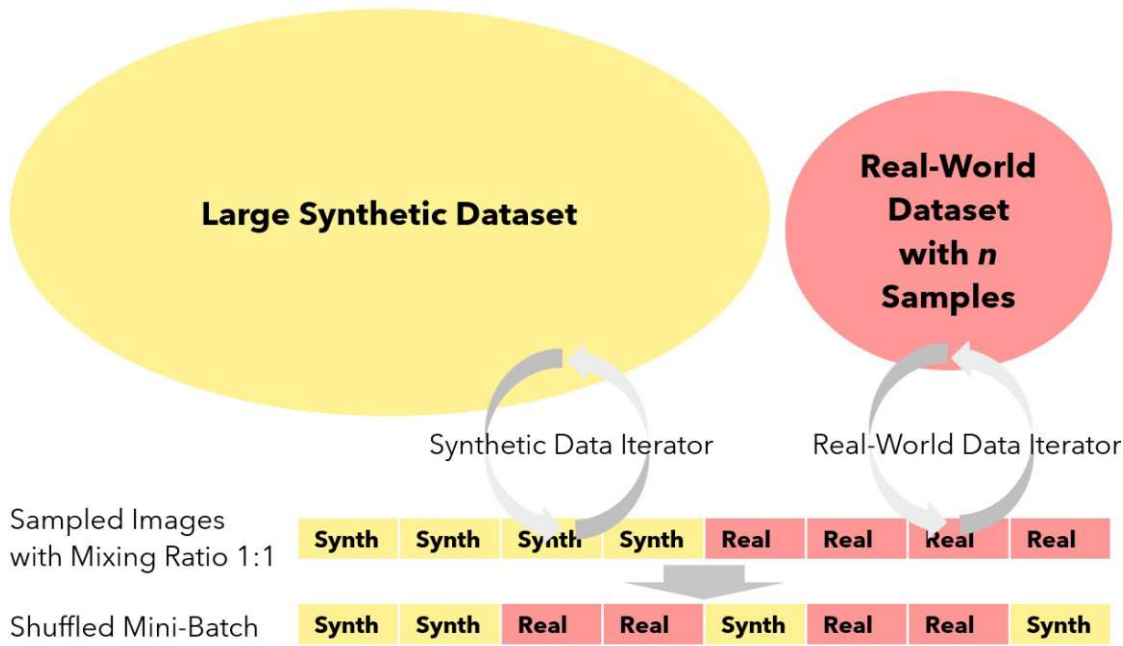


Figure 4.6: Batch mixing process with a large synthetic and a limited real-world dataset with n images. Two iterators separately iterate, sample, and shuffle the batch with the given R2S ratio of 50%.

pool of joint synthetic and real-world images. If the combined dataset is exhausted, the iterator is restarted and reshuffled, continuing this process until the desired number of epochs is reached.

This method represents a straightforward strategy for incorporating multiple data sources. However, it offers no control over the ratio of R2S data at the batch-level, potentially leading to inconsistent exposure of the two domains during training. Due to its simplicity during setup, we include it in the comparison with our proposed BLM approach.

By cross-comparing this dataset-level mixing with the BLM strategy, we aim to evaluate whether finer control over the data composition during training, enabled by BLM, leads to better performance, particularly in the presence of the S2R domain gap (P1). This comparison also allows us to examine whether the added complexity of BLM is justified by improvements in generalization, especially when synthetic data dominates the training process due to the scarcity of available real-world data (P2).

4.4 Quantitative Matrix Analysis

In this section, we outline the methodology for evaluating the impact of different data mixing strategies as described in Section 4.3 on the model performance using a matrix-based approach (C3), as shown in Table 4.1. We perform this evaluation to systematically com-

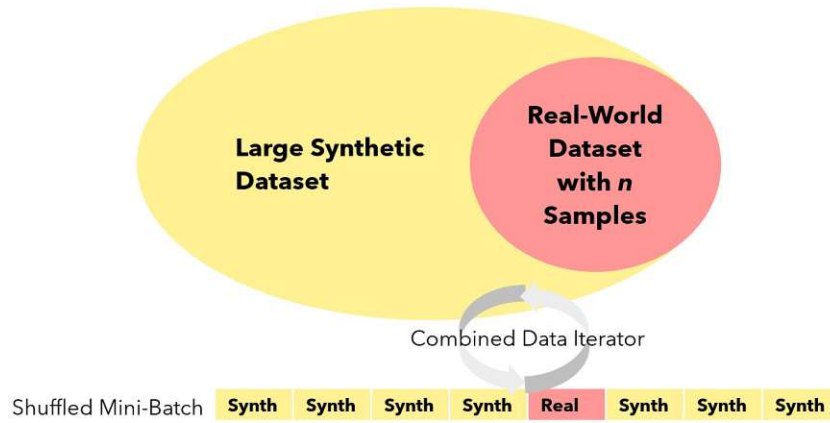


Figure 4.7: DLM of synthetic and a limited number of n real-world images on the dataset level, providing a combined pool of data to sample the batch from.

pare the effects of various synthetic and real-world data ratios on inference performance and their capabilities to close the S2R domain gap (P1).

n\r	0:1	1:3	1:1	3:1	1:0	DLM
2500						
5000						
7500						
...						

Table 4.1: Matrix of training configurations combining varying real-world (R) to synthetic (S) data ratios ($r = R : S$) and absolute numbers of real-world images (n). The rows represent the absolute number of real-world images (2500, 5000, 7500, etc.), and the columns represent DLM, as well as the mixing ratios of real-world to synthetic data (R0:S1, R1:S3, R1:S1, R3:S1, R1:S0). The performance of these configurations is evaluated separately for each metric.

The evaluation will cover the following configurations, containing the absolute and relative evaluation metrics resulting from each trial defined in Section 4.2:

- **Synthetic Data Only (R0:S1):** This configuration represents a model trained exclusively on synthetic data, providing a baseline of the model’s performance when no real-world data is included in the training set.
- **Batch-Level Mixing with Varying Ratios (R:S):** In this setup, the ratio of real-world R to synthetic S data is controlled at the batch-level. This allows for a more nuanced evaluation, as we can experiment with different ratios to understand the effects of varying levels of real-world data exposure during training.

- **Real-World Data Only (R1:S0):** This configuration involves training the model using only real-world data, offering a reference for the upper bound of performance when sufficient real-world data is available.
- **Dataset-Level Mixing (DLM):** We merge the synthetic and real-world datasets into a single combined dataset and iterate over this unified dataset during training. This represents a more traditional and simplistic approach to data mixing.

The experimental design follows the result matrix structure illustrated in Table 4.1. To evaluate the performance comprehensively, we generate a separate result matrix for each of the five metrics listed below. In these result matrices, each cell corresponds to the calculated value of that specific metric for the given configuration (n and r):

- **Absolute rotation error** as geodesic distance: $AE_{R,geo}$ (Equation 4.8)
- **Absolute overlap error** as MAE of the IoU on the sphere: $AE_{O,mae}$ (Equation 4.9)
- **Relative errors**, including:
 - Rotation: $RE_{R,\theta}$ (Equation 4.11)
 - Overlap: RE_O (Equation 4.14)
 - Combined rotation and overlap: $RE_{R|O,comb}$ (Equation 4.16)

This is done for a specific combination of real-world and synthetic data configurations. We will then analyze the following:

- **Overall Results for Each Cell:** The matrix will allow us to present the performance results for each configuration regarding a specific metric, enabling a detailed comparison across all strategies.
- **Relative Change Compared to Purely Real-World Data (R1:S0):** For each configuration, we will calculate the performance difference relative to the purely real-world setup to assess the impact of including synthetic data in the training process.
- **Relative Change Compared to DLM:** Similarly, we will compare the performance of each strategy to the results obtained from the overall dataset-level mixing approach to determine whether BLM offers an advantage.

This methodology provides a comprehensive framework for evaluating the effectiveness of different data mixing strategies, facilitating a deeper understanding of how synthetic and real-world data interact to influence model performance. It also lays the groundwork for comparing the more sophisticated BLM strategy with the simpler overall dataset-level mixing approach, particularly in terms of how well they address the synthetic-to-real domain gap.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Datasets

To answer our research questions, more specifically, if we can use neural networks to derive PTZ displacement and overlap data with competitive performance (Section 3.1.6) (Q1) and subsequently, to determine the viability of relying on virtually created synthetic data for the training (Q2), we need to create our datasets first.

In this chapter, we explain:

- The **Dataset Motivation** (Section 5.1).
- How we gather our large, simulated **Synthetic Dataset** (Section 5.2).
- How we collect **Real-world Data** (Section 5.3), showcasing complexities in this process (P2).

Both datasets are necessary to perform our PTZ regression S2R domain adaptation study (P1), as gathering large-scale real-world data remains a challenge (P2).

5.1 Dataset Motivation

To the best of our knowledge, no publicly available PTZ camera dataset fulfills all our needs regarding annotation metadata for image pose and its FoV, as outlined in (P2). This gap requires the creation of custom datasets tailored to our research objectives. As our research topic focuses on domain adaptation of PTZ camera pose and overlap regression, a review of existing PTZ domain-tailored datasets follows.

Existing PTZ-related datasets only partially address these requirements. For example, DeepPTZ [ZRK⁺20] provides large-scale, synthetically generated PTZ image pairs from SUN360 [XEOT12] panoramas. The views are cropped from panoramas rather than

captured by real PTZ cameras, so there is no per-image intrinsics or extrinsics, and overlap can only be derived from the model parameters rather than measured from actual camera motion. The original SUN360 dataset is no longer reliably available via its official website.

The Ski-Pose PTZ-Camera dataset [Spö16] contains real PTZ footage of alpine skiers, with about 20k images and accurate per-frame PTZ parameters derived from static marker triangulation, along with 2D/3D human pose annotations. Camera motion is constrained, following a single skier along fixed positions on a course, and the dataset covers a very specialized domain with limited scene diversity. Access is by request only.

The ICG Multi-Camera and virtual PTZ dataset [PRS⁺12] includes two 360° panoramic recordings (one indoor, one outdoor) and several static camera videos. A virtual PTZ framework allows cropping controllable PTZ views. Still, the dataset is small and intended primarily for multi-camera calibration and tracking rather than large-scale supervised PTZ pose or overlap regression.

In creating both the synthetic and real-world datasets, we aim to maintain comparable conditions for both. This involves capturing diverse images, as far as possible with PTZ-specific constraints (P2). Additionally, all images are generated with a 16:9 aspect ratio, which aligns with the downsizing method used in the neural network’s preprocessing stage (Section 6.4).

This automated annotation process allows us to compute the relative displacement of PTZ images in the form of a rotation quaternion and measure their FoV overlap on the unit sphere, as detailed in Section 4.2. These features ensure that the dataset provides robust ground truth for training and evaluating models designed for PTZ pose and overlap regression.

Furthermore, this methodology enables the automatic collection of all the necessary data for our domain adaptation study (Section 4.3). In this study, we investigate whether synthetic data can sufficiently train a PTZ pose and overlap regressor when compared to isolated real-world or synthetic-only training (Q2).

5.2 Synthetic Dataset

To create our visual synthetic dataset, we decided against using commonly used frameworks like CARLA [DRC⁺17]. While powerful, these frameworks are optimized for large-scale traffic scene simulation, making them less efficient and overly complex for the more controlled and focused scenarios we required. To simplify this process, we base our simulation data on the video game *Cities: Skylines II*¹, published by Paradox Interactive. This computer game is a city builder and traffic simulator with video quality comparable to CARLA, but with a focus on real-world simulation combined with usability. Its

¹<https://www.paradoxinteractive.com/games/cities-skylines-ii/about>

graphics are designed to appear realistic and of high quality, while remaining accessible to a broad user base without expertise in rendering techniques.

On top of that, Paradox Interactive allows the usage of User-Generated Content (UGC) for non-commercial use [Par23]. A further advantage is that this simulation runs within the Unity² engine, which allows us to adapt and extend its functionality through custom modifications.

We thus have out-of-the-box real-world simulations of urban and rural scenarios to create our PTZ data from. As this simulation is based on Unity³, we can derive the in-game camera parameters and create annotated PTZ simulation data. Once the setup is implemented, we can cheaply create additional training data tailored to our use case.

For our PTZ data collection, we utilize the available Cinematic Camera and Photo Mode⁴ visible in Figure 5.1. This functionality can be used to create pre-defined video scenes that can be loaded, in which the camera moves between defined key-points with configurable settings for camera position, rotation, simulation time, and various camera parameters like sensor type, size, etc. This implementation intends to develop realistic and cinematic sequences of the user’s in-game city. We harness this feature for our data generation with a custom modification, allowing us to capture timed screenshots containing the required camera metadata as part of the screenshot’s filename.

5.2.1 PTZ Movement Sequence Creation

We developed a procedure (see Algorithm 5.1) that generates a JSON file to drive the next sequence. In this algorithm, a new data file is created that directs the camera to move between manually selected key-point coordinates in the scene. For each key point, the algorithm first translates the camera to the target position and applies an initial rotation. It then generates a series of subsequent rotations, mimicking a real-world PTZ camera movement: The vertical tilt is constrained between 0° (forward) and -90° (downward), while the horizontal pan remains unrestricted.

No explicit roll is introduced. Similarly, the zoom parameter is kept constant throughout the sequence generation. Variations in zoom were omitted to isolate rotational effects and due to the additional complexity of varying camera intrinsics, thereby focusing the study on pan and tilt robustness. We strictly simulate mechanical pan-tilt constraints, but the relative rotation between tilted views geometrically inevitably requires an implicit roll component [ZRK⁺20], which is reflected in the generated camera poses of our dataset.

Each movement is scheduled at a fixed interval to allow time for automatic screenshot capture. Furthermore, while the same key-point positions are used for the walk between these points during the nighttime, the rotation parameters are slightly altered. This

²<https://unity.com/en>

³<https://unity.com/en>

⁴<https://www.paradoxinteractive.com/games/cities-skylines-ii/features/cinematic-camera-photo-mode>



Figure 5.1: Cities: Skylines II - Cinematic Camera & Photo Mode UI Example

creates data per key-point as illustrated in Figure 5.2 and enables the simulation of a PTZ camera mounted at key locations, capturing different orientations of the same scene under varying times of day in a controlled manner.



Figure 5.2: In-game movement example of a single PTZ camera at a fixed location. The camera is rotated in a controlled way during the day and at night.

5.2.2 Automatic Annotation Modification

With this script, we can simulate PTZ camera movements, but to create an actual dataset, we need to capture annotated images from this sequence automatically. The base game does not include such functionality, thus we need to add a modification that allows us to first, gather all the required annotation data and link it to the image, in our case as part of the screenshot filename, and second, capture periodic screenshots after starting our custom video sequence every few seconds in between the camera movements. To simplify

Algorithm 5.1: Create a controlled but random PTZ camera movement sequence

```

1 Function calcPTZMovementSequence (
    key_points, numberOfRotations, isDay):
2   currentTime  $\leftarrow$  0;
3   movements  $\leftarrow$  {"translation": [], "rotation": [], "at_time": []};
4   foreach (startPosition, startRotation) in key_points do
5     translate(currentTime, startPosition, movements);
6     rotate(currentTime, startRotation, movements);
7     for i  $\leftarrow$  0 to numberOfRotations do
8       currentTime  $\leftarrow$  currentTime + transformationInterval;
9       randomRotation  $\leftarrow$  getRandomRotation(startRotation);
10      rotate(currentTime, randomRotation, movements);
11      randomTime  $\leftarrow$  getRandomTime(isDay);
12      modifyTimeOfDay(currentTime, randomTime, movements);
13    end
14  end
15  return movements;

16 Procedure translate (currentTime, targetPosition, movements):
17 | Append (currentTime, targetPosition) to movements[translation];

18 Procedure rotate (currentTime, targetRotation, movements):
19 | Append (currentTime, targetRotation) to movements[rotation];

20 Procedure modifyTimeOfDay (currentTime, newTime, movements):
21 | Append (currentTime, newTime) to movements[at_time];

22 Function getRandomRotation (startRotation):
23 | newTilt  $\leftarrow$  startRotation[tilt] + randomUniform(-30, 30);
24 | newTilt  $\leftarrow$  clip(newTilt, (0, -90));
25 | newPan  $\leftarrow$  startRotation[pan] + randomUniform(-30, 30);
26 | return {tilt : newTilt, pan : newPan};

27 Function getRandomTime (isDay):
    // Sample a random hour depending on whether it is day
    // or night
28 if isDay then
29 | return randomUniform(10, 16);
30 else
31 | return randomUniform(0, 3);
32 end

```

this modding process, we implement a Unity⁵ mod with the Bepis Injector Extensible (BepInEx) framework [Bep24].

The final modification can be summarized with Algorithm 5.2 and Algorithm 5.3. In Algorithm 5.2, we use HarmonyX⁶ to patch Cities: Skylines II's `CaptureScreenshot` method of the `ScreenUtility`. We do so by applying a so-called prefix patch. In detail, before the actual `CaptureScreenshot` is invoked, the patch is injected and executed first. We can use this to our advantage to override the `fileName` argument, which would be used by the game as a screenshot filename. We introduce the logic in the described algorithm to persist the image's annotation metadata in the filename string. A resulting annotated image is shown in Figure 5.3.



Figure 5.3: A virtually created synthetic image from Cities: Skylines II with annotated camera parameters for position, rotation, and camera intrinsics, resulting in a filename of `Fabius_d20240823144956_p-634.8_79.9_-259.2_r16.35_238.60_at56.23787_ap87.06240_f14.00000_sh4.28_sw7.60.jpeg`.

The information we are requiring is as follows:

- The game's **Save Name** to distinguish equal positions between different simulations.
- The **Camera Position** of the in game camera in x,y,z world coordinates. With this information, we can group images from the same PTZ camera position.

⁵<https://unity.com/en>

⁶<https://github.com/BepInEx/HarmonyX/wiki>

- The **Camera Rotation**, describing the camera's view point orientation. We only collect tilt and pan orientation in degrees and omit the roll, as the camera is never rolled and thus always 0, as the explicit mechanical roll is fixed at 0°.
- The **Camera FoV**, both horizontal and vertical, allowing it to calculate the overlap between two PTZ images.
- We also collect **Sensor Size** and **Focal Length** for documentation and comparability, even though not directly required in the calculations.

After we described which data we extract automatically during in-game screenshot capture, we now describe how we invoke this periodically for Cities: Skylines II with Algorithm 5.3. This procedure continuously monitors the game's camera state and environmental conditions to determine the best moments for capturing annotated screenshots. The goal is to ensure that each capture occurs only after the camera has completed its position and rotation transitions, and that any significant changes in illumination, such as a transition between day and night, are accounted for by temporarily skipping captures. This careful timing helps maintain consistency in the dataset.

The key steps of the algorithm are as follows:

- **Initialization:**
 - Set a delay slightly smaller than the target two seconds to account for the short freeze when taking an in-game screenshot.
 - Initialize the last recorded camera position with a default value.
 - Record the current sunlight intensity. We will use this information to skip capturing screenshots between day and night cycles to avoid visual artifacts due to extreme and quick illumination changes.
- **Monitoring Camera Movement**
 - In each iteration, get the current camera position.
 - Wait for four seconds if the camera was moving recently between key positions. This should allow the game to load the new scene properly.
- **Monitoring Rotation Movement**
 - Record the time of the last rotation event.
 - Wait in 0.5-second increments until at least 0.5 seconds have passed since the last rotation, ensuring that any rotational movement has finished.
- **Monitoring Illumination Change**
 - Check the current sunlight intensity; if it is 0, we are at night.

- Determine if there is a transition from night to day or vice versa.
 - Skip the screenshot and restart the cycle to prevent captures during periods where the lighting is still adjusting.
- **Invoke Screenshot Capture**
 - If no skip condition is triggered, we invoke the in-game’s `CaptureScreenshot` method, which delivers us annotated image information using our prefix patch annotation of (Algorithm 5.2).
 - After the invocation, wait for the preset delay to let the camera sequence perform the next PTZ movement.
 - **Termination**
 - Continuously check if the cinematic sequence created with the Algorithm 5.1 is still playing.
 - If the sequence has ended, exit the loop and stop further invocations.

By combining the prefix patching strategy (Algorithm 5.2) with periodic invocation (Algorithm 5.3), we establish a robust framework for generating annotated datasets directly from a Unity⁷-based environment using the BepInEx modding framework.

5.3 Real-World Dataset

For our domain adaptation study, we need to collect real-world PTZ data in order to assess whether the training of our PTZ displacement network (Q2) can mainly rely on synthetic data. As explained in Section 5.1, we could not locate any publicly available datasets that cover our required conditions and metadata, namely for each image having annotations of the FoVs of the camera, their PTZ orientation, as well as further parameters like focal length or sensor size. To record our own data, we mounted two separate HIKVISION⁸ PTZ cameras, namely the model DS-2DF8223I-AEL and DS-2DE4220IW-DE.

As we have limited resources for mounting further cameras for collecting more real-world data (P2), we conceptually separate their FoVs for training and evaluation (see Figure 5.4 and Figure 5.5), creating four distinct scenes, two for training and two for evaluation.

To get annotated real-world images from both cameras, we developed a script that controls the HIKVISION cameras via their exposed API (Algorithm 5.4) and can move the camera to a desired location. To collect our dataset, we randomly moved the camera during multiple days and at different times of the day, to collect a diverse dataset with varying lighting, weather, and traffic conditions.

⁷<https://unity.com/en>

⁸<https://www.hikvision.com/en/>

Algorithm 5.2: Prefix patching of the built-in *CaptureScreenshot* method to save annotated screenshots each time a screenshot is invoked

```

1 Function prefixCaptureScreenshot (fileName):
    // Initialize a new fileName, discard original
2   fileName ← getSaveName() + "_d" + getCurrentTime();
    // Append current position
3   pos ← getCameraPosition();
4   fileName ← fileName + "_p" + pos.x + "_" + pos.y + "_" + pos.z;
    // Append current rotation
5   rot ← getCameraRotation();
6   fileName ← fileName + "_r" + rot.tilt + "_" + rot.pan;
    // Append the horizontal FOV
7   horizontalFOV ← getHorizontalFOV();
8   fileName ← fileName + "_at" + horizontalFOV;
    // Calculate and append the vertical FOV
9   aspectRatio ← getCameraAspectRatio();
10  verticalFOV ← convertFOV(horizontalFOV, aspectRatio);
11  fileName ← fileName + "_ap" + verticalFOV;
    // Append the current focal length
12  fileName ← fileName + "_fl" + getFocalLength();
    // Append the current sensor size
13  sensorSize ← getCameraSensorSize();
14  fileName ← fileName + "_sh" + sensorSize.height;
15  fileName ← fileName + "_sw" + sensorSize.width;
    // Return our patched fileName
16  return fileName;
17 end

18 Function convertFOV (inputDegFOV, aspectRatio):
19  inputRadFOV ← toRadians(inputDegFOV);
20  convertedFOV ←  $2 \times \arctan\left(\tan\left(\frac{\textit{inputRadFOV}}{2}\right) \times \textit{aspectRatio}\right)$ ;
21  return toDegrees(convertedFOV);
22 end

```

Algorithm 5.3: Periodic invocation of the patched built-in screenshot method
(see patching Algorithm 5.2)

```

1 Procedure periodicScreenshotInvokation():
2   delayInSeconds  $\leftarrow$  1.7;
3   lastPosition  $\leftarrow$  (0,0,0);
4   lastSunLight  $\leftarrow$  getCurrentSunlightIntensity();
5   while true do
6     currentPosition  $\leftarrow$  getCurrentCameraPosition();
7     while currentPosition  $\neq$  lastPosition do
8       // Let the camera finish position transition
9       waitForSeconds(4.0);
10      lastPosition  $\leftarrow$  currentPosition;
11    end
12    lastRotationTime  $\leftarrow$  getLastRotationTime();
13    while (currentTime - lastRotationTime) < 0.5s do
14      // Let the camera finish any rotation transition
15      waitForSeconds(0.5);
16    end
17    currentSunLight  $\leftarrow$  getCurrentSunlightIntensity();
18    skipScreenshot  $\leftarrow$  false;
19    nightDayChange  $\leftarrow$  (lastSunLight = 0  $\wedge$  currentSunLight > 0);
20    dayNightChange  $\leftarrow$  (lastSunLight > 0  $\wedge$  currentSunLight = 0);
21    if nightDayChange  $\vee$  dayNightChange then
22      // Illumination change requires time to adjust
23      skipScreenshot  $\leftarrow$  true;
24    end
25    lastSunLight  $\leftarrow$  currentSunLight;
26    if  $\neg$ skipScreenshot then
27      // Take screenshots and apply delay
28      invokeCaptureScreenshot();
29      waitForSeconds(delayInSeconds);
30    end
31    if  $\neg$ isCinematicSequencePlaying() then
32      // Sequence over, stop periodic invocation
33      return
34    end
35  end

```

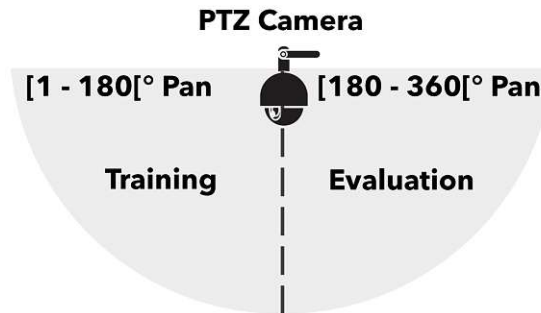


Figure 5.4: Schematic representation of FoV separation for a PTZ camera. The camera covers the lower hemisphere and can tilt around the vertical axis. For training and evaluation data collection, and due to limited hardware availability and mounting constraints, the 360° panning range is split into two half-open intervals: $[1-180[$ for training and $[180-360[$ for evaluation. Some overlap in the lower hemisphere is unavoidable due to tilting, but this separation ensures that the majority of training and evaluation images come from distinct panning regions (P2).

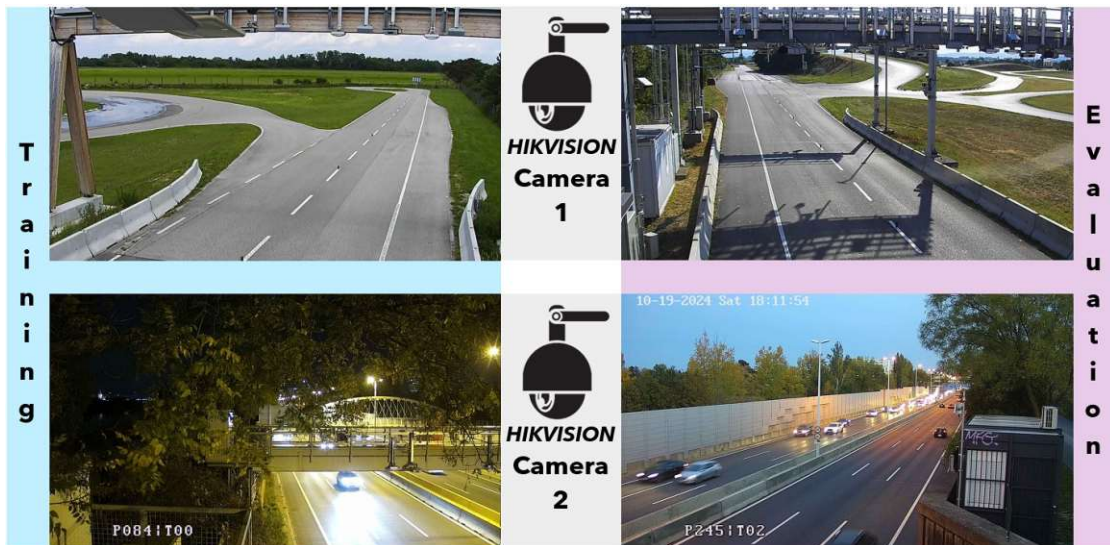


Figure 5.5: This figure shows the installation of our two HIKVISION PTZ cameras and the scenes they capture to gather training and evaluation data. The upper row shows Camera 1 footage at different times of the day, while the lower row shows Camera 2. The figure demonstrates the real-world application of the schematic from Figure 5.4. Each camera pans over 180° of the shared 360° coverage: the left side (blue) indicates the area used for training images, and the right side (magenta) indicates the portion used for evaluation. The separation is defined by the panning angle, while the cameras tilt within the lower hemisphere.

The separation into training and evaluation was performed in postprocessing, where the images were separated regarding their panning angle for one half of the 180° panning FoV into either set, as conceptualized in Figure 5.4.

Algorithm 5.4: Execute PTZ Movements and Capture Annotated Screenshot

```

1 Procedure executePTZMovements (numberOfMovements, cameraId):
2   for  $i \leftarrow 1$  to numberOfMovements do
3     // Create and send PTZ movement payload
4     movementPayload  $\leftarrow$  createFromDesiredMovement();
5     api.requestPTZMovement(movementPayload);
6     // Pause to allow movement to complete
7     sleep(2 seconds);
8     // Request updated PTZ metadata
9     tilt, pan, zoom  $\leftarrow$  api.requestPTZStatus();
10    // Based on camera specification,
11    // calculate FOV for height and width
12    fovHeight  $\leftarrow$  calculateFOV(sensorHeight, focalLength);
13    fovWidth  $\leftarrow$  calculateFOV(sensorWidth, focalLength);
14    // Construct annotated file name
15    filename  $\leftarrow$  cameraId + "_" + tilt + "_" + fovHeight;
16    filename  $\leftarrow$  filename + "_" + pan + "_" + fovWidth;
17    // Capture the screenshot with the annotated filename
18    takePicture(filename);
19  end
20 end

21 Function calculateFOV(sensor_dimension, focalLength):
22   fov  $\leftarrow$   $2 \times \text{toDegrees}\left(\arctan\left(\frac{\text{sensor\_dimension}}{2 \times \text{focalLength}}\right)\right)$ ;
23   return fov;
24 end

```



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Implementation

To address the Synthetic-To-Real (S2R) domain gap in Pan, Tilt and Zoom (PTZ) image-based rotation and overlap estimation (P1) and the limited availability of annotated real-world data (P2), the methodological concepts introduced earlier must be tested in a working system. This chapter provides the actual analysis details by turning the domain adaptation framework (Chapter 4) and datasets (Chapter 5) into an executable study. This implementation is essential for answering the core research questions, whether relative PTZ displacement can be learned with competitive performance (Section 3.1.6) from single-camera images (Q1), and whether synthetic data can be effectively leveraged during training (Q2).

We translate the conceptual domain adaptation methodology (Chapter 4) and the created datasets (Chapter 5) into an executable framework. This includes our PTZNet model, the training and optimization procedures, and the data-handling layer supporting various S2R ratios.

Specifically:

- Detailing the **PTZNet Network Architecture** (Section 6.1).
- Outlining the **Training Pipeline** (Section 6.2).
- Describing the **Training Optimization Strategy** (Section 6.3).
- Covering the **Input Image Preprocessing** (Section 6.4).
- Presenting our **Configurable Dataset Integration**, mixing synthetic and real-world datasets (Section 6.5).

These implementation details form the technical basis for testing our hypotheses. That relative PTZ pose and overlap can be regressed with competitive performance (Section 3.1.6)

from virtually created relative image pairs (H1), and that batch-level integration of synthetic data boosts performance across S2R with varying numbers of real-world images (H2). These hypotheses are addressed in the experiments of Chapter 7.

6.1 PTZNet Network Architecture

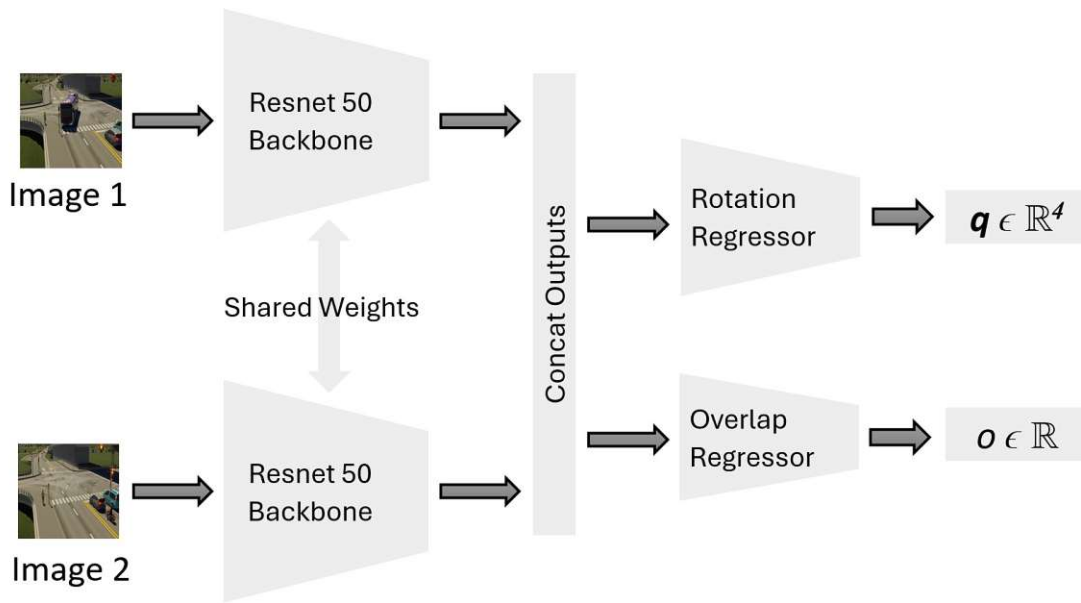


Figure 6.1: Schematic overview of the designed PTZNet architecture. The model follows a Siamese design using weight-shared ResNet50 backbones to extract features from an input image pair. The resulting feature vectors are concatenated and routed to specific regression heads to predict the relative rotation quaternion $\mathbf{q} \in \mathbb{R}^4$ and the scalar view overlap $o \in \mathbb{R}$.

The overall Siamese-based design for relative PTZ pose and overlap estimation, which we call PTZNet, is illustrated in Figure 6.1. As introduced in Section 4.1, the network passes an input image pair through two weight-shared ResNet50 [HZRS16] backbones that are pre-trained on ImageNet [DDS⁺09]. The final classification layers are removed, and the resulting feature vectors are concatenated and forwarded into two separate fully connected regressor heads.

To address the specific complexity of each task, the rotation (Figure 6.2) and overlap (Figure 6.3) regressors employ different sizing using exponential reduction in the subsequent Fully Connected Layers (FCLs). The network outputs a tuple (\mathbf{q}, o) , where $\mathbf{q} \in \mathbb{R}^4$ represents the rotation required to align the camera views, and $o \in \mathbb{R}$ approximates the spherical IoU.

The outputs are constrained to ensure valid geometric representations. The rotation

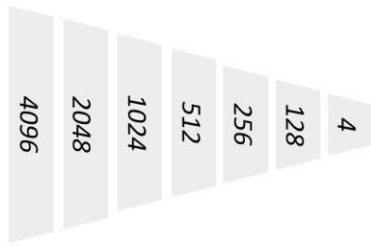


Figure 6.2: Architecture of the rotation regression head. It employs a sequence of fully connected layers with exponential dimensionality reduction to regress the final 4D quaternion vector.

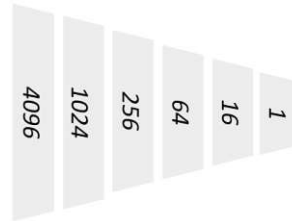


Figure 6.3: Architecture of the overlap regression head. This branch utilizes a steeper exponential reduction scheme to compress features into the final scalar overlap.

vector is normalized to a unit quaternion, as defined in Equation 6.1,

$$\mathbf{q} = \frac{\mathbf{q}_{raw}}{|\mathbf{q}_{raw}|}, \quad \mathbf{q}_{raw} \in \mathbb{R}^4, \quad (6.1)$$

where \mathbf{q}_{raw} denotes the raw output of the rotation head. Simultaneously, the overlap score is bound to the $[0, 1]$ interval via a sigmoid activation, shown in Equation 6.2,

$$o = \sigma(s) = \frac{1}{1 + e^{-s}}, \quad s \in \mathbb{R}, \quad (6.2)$$

where s is the raw logit from the overlap head. For the underlying mathematical definitions of the quaternion representation and spherical intersection-over-union metric, we refer to Sections 2.2.3 and 2.2.4, respectively.

6.2 Training Pipeline

After describing the overall architecture of our PTZNet, we now explain how the whole training pipeline is structured. We outline the overall training procedure, detail the optimization approach, present a combined rotation and overlap loss function, and discuss data pre-processing and augmentation techniques used.

This pipeline addresses the challenge of training a robust model with limited real-world data (P2) by leveraging transfer learning on a ResNet50 backbone, a model pre-trained on real-world images. It further ensures generalizability with a robust optimization approach and incorporates extensive image augmentation methods. Most noticeably, it introduces a configurable framework for systematically integrating synthetic and real-world PTZ camera data, which enables us to perform a quantitative evaluation of diverse BLM and DLM strategies to bridge the S2R domain gap (P1).

As we continuously iterate and combine the synthetic (Section 5.2) and real-world dataset (Section 5.3) as explained in Section 4.3, we do not rely on the standard epoch definition,

where one epoch usually corresponds to one dataset iteration. This fixed structure ensures comparability across experiments and isolates the effects of data mixing strategies. We always train for 700 epochs, during which we observe a general convergence for the whole trial. In this setup, one epoch is represented as 150 batches, each having a batch size of 32. A fixed random seed is introduced to ensure reproducibility, enabling consistent initialization and data shuffling throughout the experiments. We perform an initial evaluation and monitor model performance gradually after each epoch. Each data combination method is tested with the same setup to enable comparability of the results.

By maintaining this uniform experimental design, we ensure that variations in model performance stem solely due to the interaction between synthetic and real-world data and not due to variations in training parameters or randomness. The approach prioritizes reproducibility and comparability over extensive analysis of individual model components.

6.3 Optimization

We optimize the PTZNet architecture to minimize the loss defined in Section 4.2.1. To recapitulate, it is defined in Equation 6.3. Here, the choice of a static balance factor $\beta = 3.092$ is crucial as it regulates the relative influence of the overlap loss L_o compared to the rotation loss L_r . This value was selected to ensure approximately equal initial magnitudes of both losses, supporting stable convergence during training.

To enforce a minimal level of shared features, any image pair below the 30% overlap threshold receives no rotation loss contribution, effectively skipping its gradient update while still accounting for the overlap loss. This mechanism is designed to ensure robustness in feature extraction. The 30% overlap threshold was chosen to ensure a minimal set of shared features between both images while still allowing complex rotations to be trained and evaluated. This value was selected based on preliminary experiments and represents a practical trade-off.

$$L = L_r + \beta \cdot L_o \quad (6.3)$$

with $\beta = 3.092$

Based on this loss function, we adopted the optimization strategies proposed by RCP-Net [YLZ20], but refined them significantly with extensive preliminary experiments, ensuring stability and performance. For our regressor’s FCLs we use ReLU [NH10] activations with its weights being initialized for leaky ReLU non-linearity [HZRS15], mitigating the dying ReLU issue [LSSK19]. To enhance generalization and prevent overfitting, we incorporated dropout regularization, with a probability of $p = 0.4$ for overlap and $p = 0.3$ to $p = 0.1$ for quaternion prediction. These values were manually tuned and selected, providing the best results in preliminary experiments. While dropout is less common in camera pose regression, some studies have demonstrated its potential in similar contexts [VVL19, KK22].

As training optimizer we selected Adaptive Movement Estimation (Adam) [KB14] together with an exponential learning rate scheduler ($1e - 4$ initial learning rate, decay factor $\gamma \approx 0.9977$). We tested alternative configurations such as Stochastic Gradient Descent (SGD) [RM51] or linear learning rate schedules, but they were less effective, converged more slowly, or were more sensitive to hyperparameter choices.

This setup delivered robust results consistently throughout our evaluations in Section 7, making it a reliable choice for the task.

6.4 Data Preprocessing

In this part, we describe our data pre-processing pipeline, starting with downsizing, continuing with our extensive augmentation methods to improve robustness, up to the final normalization step required by ResNet-based models.

6.4.1 Downsizing

Initially, all training and evaluation images are downsized to PTZNet’s input size of 224×224 using bilinear interpolation while maintaining a strict $1 : 1$ aspect ratio. This resolution is chosen to match the input requirements of the ResNet50 backbone. We observed better performance using a $1 : 1$ aspect ratio, even though this introduces distortion, compared to alternative approaches such as black-bar padding (adding empty borders to maintain the original aspect ratio) or center-cropping (removing peripheral regions to fit the target aspect ratio).

Figure 6.4 illustrates those methods. Consider a $16 : 9$ image with a resolution of 1280×720 that is resized to 224×224 pixels. If the image is center-cropped ($1 : 1$ Cropping), the image width transforms to approximately $224 \cdot \frac{16}{9} \approx 398$ pixels, where we only retain the central 224 pixels. Thus, about $1 - \frac{224}{398} \approx 44\%$ of the horizontal content is lost. Similarly, black-bar padding ($1 : 1$ Padding) scales the content to $224 \times (224 \cdot \frac{9}{16}) \approx 224 \times 126$. Also here, about 44% of the original information is discarded.

By reshaping the whole image ($1 : 1$ Distorting), we preserve the entire field of view, but have to consider the cost of the introduced distortion. To verify that this does not hinder our domain adaptation study (Q2), we performed a preliminary ablation (Figure 6.5), showing that for our context, the model can successfully learn such mapping. All the used PTZ datasets, either synthetic or real-world, perform in the same $16 : 9 \rightarrow 1 : 1$ aspect ratio transformation. For our trial context, the network can improve its predictions using the additional $\approx 44\%$ of peripheral content and is thus outperforming the other methods with a minimum increase of 7%.

6.4.2 Data Augmentation

We aim to improve our training dataset diversity through augmenting each image individually with a probabilistic augmentation pipeline, applying distinct visual and geometric-

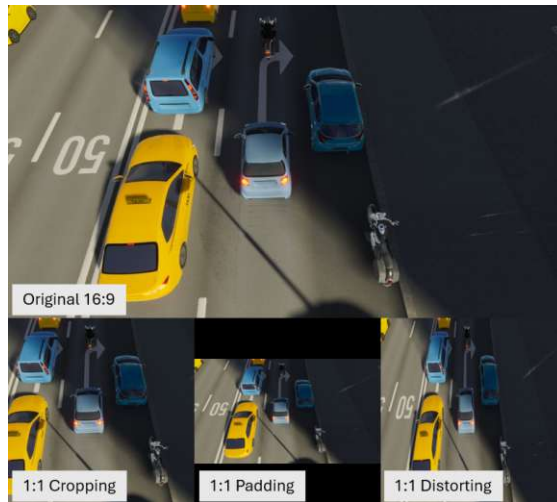


Figure 6.4: Visual comparison of downsizing methods used to perform a 16 : 9 \rightarrow 1 : 1 transformation, including cropping, padding, and distorting.

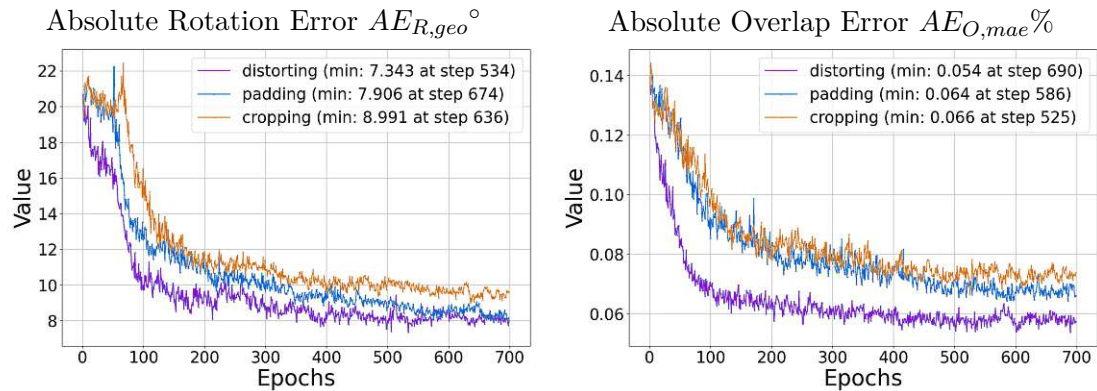


Figure 6.5: Downsizing ablation study for a 25% real-ratio with $n = 5000$, showing absolute rotation ($AE_{R,geo}^\circ$) and overlap ($AE_{O,mae}\%$) errors across epochs.

preserving transformations (see example images of Figure 6.6). The augmentation strategy makes use of the following operations, provided through Albumentations [BIK⁺20]:

- **Noise Injection:** Gaussian noise is applied, with parameters tuned to suit the 224×224 image size.
- **Blur Techniques:** Among motion blur, median blur, or standard blur, one single operation is selected, having its kernel limits optimized for the small image size.
- **Contrast and Detail Enhancement:** Techniques such as Contrast Limited Adaptive Histogram Equalization (CLAHE) [Zui94], sharpening, embossing, or brightness/contrast adjustments are applied to vary levels of image details.

- **Color Adjustments:** We simulate varying lighting conditions with transformations including hue, saturation, and value shifts alongside randomized gamma correction.
- **Advanced Color Transformations:** Color jittering or a conversion to grayscale is performed to diversify the color representations.
- **Compression Artifacts:** Image compression is introduced to mimic quality degradations encountered in real-world scenarios.



Figure 6.6: Example images after applying augmentations, illustrating the resulting diversity. Each image is randomly generated by combining multiple filters such as noise injection, blur, contrast/detail enhancements, color adjustments, advanced color transformations, and compression artifacts.

Additionally, mirrored versions of each train image are generated while recalculating rotation and overlap parameters, thus synthetically doubling the dataset size. This comprehensive preprocessing pipeline not only standardizes the input size for seamless integration with ResNet50 but also promotes robust model generalization by exposing it to a wide range of simulated imaging conditions.

6.4.3 Normalization

The final processing step for the training as well as evaluation data is to normalize the images using the standard ResNet50 parameters [HZRS16]. This involves scaling the image pixel values and applying the normalization to their RGB color channels with $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$, thereby ensuring compatibility with the network’s input requirements. This whole preprocessing approach aims to mitigate overfitting by exposing the model to diverse scenarios, thereby improving generalization performance on unseen data.

6.5 Configurable Dataset Integration

This section describes our framework used to dynamically test different synthetic and real-world data ratios, as well as their isolated impact for varying numbers of real-world images. More specifically, we describe the technical implementation of Section 4.3, which is required to evaluate and answer our question if we can rely mainly on synthetic data for our relative PTZ rotation and overlap regression training (Q2).

Our training framework offers adjustable settings to train with n real-world image pairs taken of a pool of available images. Additionally, we can define the ratio r of real-world images in the combined training batch (Figure 4.6). This allows us to test different real-world ratios mixed with synthetic data. We also offer a distinct setting which allows us to mix the synthetic and real-world data on a global level (Figure 4.7).

We utilize this framework to train and evaluate all scenarios that we will discuss in Section 7. By setting $n = 0$, we establish the synthetic-only isolation baseline, where we train exclusively on the available synthetic data. Additionally, setting $n = 2500$ with a ratio r of 100% configures a real-world isolation baseline, integrating n image pairs from real-world data. Both parameters, r and n , can be adjusted to define mixed batch training configurations. For instance, with $n = 2500$ and a batch ratio r of 50%, batches are evenly shuffled between synthetic and real-world data. Modifying r to 25% or 75% alters the proportion of real-world samples per batch accordingly. Additionally, we can configure the framework to mix synthetic data and n real-world image pairs globally, allowing a direct comparison between batch and full-dataset mixing strategies (Table 4.1). By employing fixed random seeds and carefully controlled data extraction, we ensure a reproducible experimental setup. In each configuration, we evaluate and export the training and evaluation metrics defined in Section 4.2.

Evaluation and Results

Having laid the foundation for our method, we evaluate its performance concerning our research questions and hypotheses. We summarize the tested hypotheses as follows:

- (H1) PTZNet can regress relative rotation displacement with competitive performance (Section 3.1.6) as well as Field of View (FoV) overlap from Pan, Tilt and Zoom (PTZ) image pairs.
- (H2) Mixing synthetic and real-world data, especially with Batch-Level Mixing (BLM), yields better performance than using either dataset in an isolated manner, or Dataset-Level Mixing (DLM).
 - (H2.1) Mixed training outperforms training on only synthetic or only real-world data.
 - (H2.2) BLM achieves better results than DLM.
 - (H2.3) Synthetic data remains beneficial even as the volume of real-world data increases.

First, we assess whether neural networks can predict PTZ displacement and overlap with competitive performance (Section 3.1.6) from a pair of images (Q1), corresponding to Hypothesis (H1). We then investigate whether simulated synthetic data can serve as the main training source when real-world data is limited (Q2), addressing Hypothesis (H2) and its sub-hypotheses. Specifically, we test if mixed training with synthetic and real data improves over using either alone, if BLM is more effective than dataset-level mixing, and if the benefits of synthetic data persist as more real data becomes available (Figure 7.1).

This chapter is structured as follows:

- The **Experimental Setup** (Section 7.1).

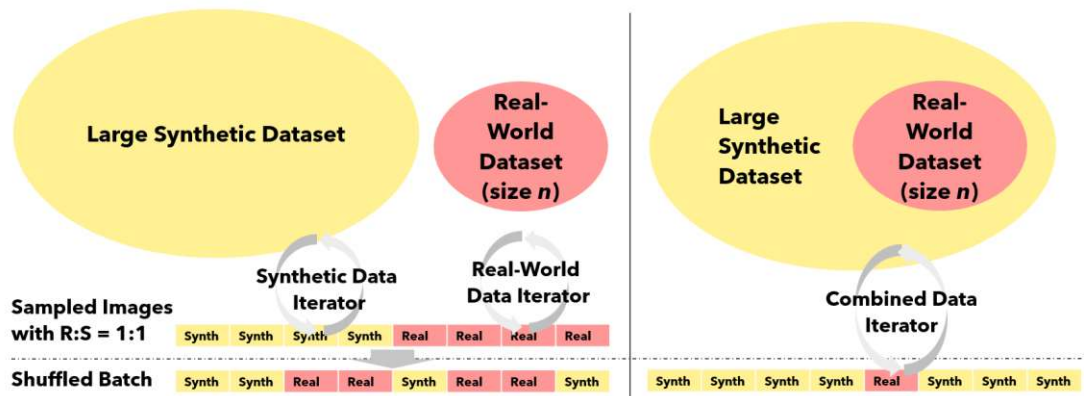


Figure 7.1: Comparison of BLM (Section 4.3.1) on the left and DLM (Section 4.3.2) on the right, when combining a large synthetic dataset with a limited real-world dataset of size n . In Batch-Level Mixing, two iterators separately draw and shuffle samples to enforce a fixed R2S ratio (e.g., R1:S1 with 50% real and 50% synthetic). In Dataset-Level Mixing, both datasets are concatenated and sampled by a single iterator, so the batch composition follows the overall dataset proportions.

- **PTZNet’s Performance** and detailed **Rotation, Overlap and Combined Accuracy Analyses** (Section 7.2).
- Exploring domain adaptation and the impact of different data mixing strategies with our proposed **Matrix Analysis** (Section 7.3).

7.1 Experimental Setup

To systematically evaluate our research questions and hypotheses, we conduct a series of trials based on controlled combinations of real-world and synthetic training data. This section outlines the model training setup, trial configurations, and the evaluation conditions used throughout our experiments.

7.1.1 Model Training Details

All PTZNet models were trained for a total of 700 epochs using a fixed batch size of 32. Each epoch consists of 150 randomly sampled batches. The total runtime per training session was approximately 17 hours. We conducted the training on a machine that is equipped with an NVIDIA RTX 3090 GPU with 25 GB of VRAM¹. This setup provides us with sufficient computing resources, as a single training with the architecture described in Section 6.1 consumes less than 8GB of VRAM.

¹<https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/>

The training pipeline, as described in Section 6.2, remains consistent across all experiments to ensure comparable results. Each training run is performed from scratch with pre-trained ResNet weights, initializing the network with the same random seed to preserve consistency in weight initialization and data shuffling.

7.1.2 Trial Configuration

To analyze how the inclusion and ratio of R2S data affect the model performance, we designed a trial matrix (see Section 4.4) that varies along two key dimensions: the number of real-world image pairs used for training, and the data composition of each training batch.

Specifically, we conducted our experiments with three quantities of real-world training samples n , namely 2500, 5000, and 7500 pairs. For each of those dataset sizes, we evaluated five batch compositions reflecting the ratio r of real-world data within a training batch, which are:

- R0:S1 (synthetic only)
- R1:S3 ($\frac{1}{4}R + \frac{3}{4}S$)
- R1:S1 ($\frac{1}{2}R + \frac{1}{2}S$)
- R3:S1 ($\frac{3}{4}R + \frac{1}{4}S$)
- R1:S0 (real-world only).

In other words, an R0:S1 ratio corresponds to a training performed exclusively with synthetic data S , while an R1:S0 ratio uses only real-world data R . The intermediate ratios represent BLMs where samples are randomly drawn to meet a fixed batch composition. An R1:S3 mix, for example, means that $\frac{1}{4}$ of each training batch consists of randomly selected real-world image pairs and the remaining $\frac{3}{4}$ of synthetic ones. Details about the batch-mix foundation can be found in Section 4.3.1.

In addition to these BLM strategies, we also evaluated Dataset-Level Mixing (DLM) for each real-world data amount n (find theoretical details in Section 4.3.2). In this configuration, the training dataset was formed by simply concatenating synthetic and real-world pairs before performing the random sampling, without enforcing a specific batch composition. This allows us to compare whether structured BLM provides any benefit over naive dataset-level integration.

Throughout all experiments, the synthetic dataset remained fixed, consisting of approximately 345,000 virtually created PTZ image pairs. This ensured that the primary focus was on evaluating the impact of varying the quantity of real-world data and the data-mixing strategies employed for its integration.

7.1.3 Evaluation Metrics

Model performance is evaluated on both rotation and overlap prediction tasks using the metrics introduced in Section 4.2. All metrics are computed on a separate evaluation set consisting of 5120 real-world PTZ image pairs and are reported as median values over the entire set.

For rotation, we assess prediction accuracy using the geodesic distance $AE_{R,geo}$ in degrees (Equation 4.8) and the relative rotation error $RE_{R,\theta}$ in percent (Equation 4.11). The overlap predictions are evaluated using the Mean Absolute Error (MAE) $AE_{O,mae}$ in percent (Equation 4.9) and the relative overlap error RE_O in percent (Equation 4.14). Finally, to provide a unified view of model performance across both tasks, we additionally report the combined relative error $RE_{R|O,comb}$ (Equation 4.16), computed as the arithmetic mean of the respective median relative rotation and overlap errors.

These metrics enable consistent comparison across different training configurations and data compositions, while capturing both absolute and relative performance accuracy for rotation and overlap prediction tasks.

7.2 Results

To validate our hypothesis (H1), that a deep neural network trained on a methodically combined dataset of virtually created synthetic and real-world PTZ image pairs can regress relative rotation and overlap with competitive performance (Section 3.1.6) using images from a single PTZ camera, we conducted experiments across varying configurations of synthetic and real-world data mixing strategies. The results of these trials are presented in Figures 7.2, 7.3, and 7.4.

7.2.1 Performance Analysis

We want to start with an overall evaluation of our trial. An in-depth comparison and matrix-like analysis will be conducted afterwards in the following Section 7.3.

Across all trials, we observe that the model can achieve competitive performance (Section 3.1.6) in regressing both relative rotation and overlap metrics when trained on mixed datasets of synthetic and real-world PTZ image pairs. Specifically, configurations such as DLM, R1:S3, R1:S1, and R3:S1 consistently outperform synthetic-only (R0:S1) and real-world-only (R1:S0) training setups. This indicates that combining synthetic and real-world data provides complementary benefits, enabling the model to generalize better to unseen real-world scenarios (see Sections 4.2.2 and 4.2.3 for the definition of the reported metrics).

7.2.2 Rotation Accuracy

The rotational errors $AE_{R,geo}$ (Equation 4.8) and $RE_{R,\theta}$ (Equation 4.11), seen in Figures 7.2, 7.3, and 7.4, show a clear improvement when using mixed datasets. For

Trial Execution for $n = 2500$ Real-World Images

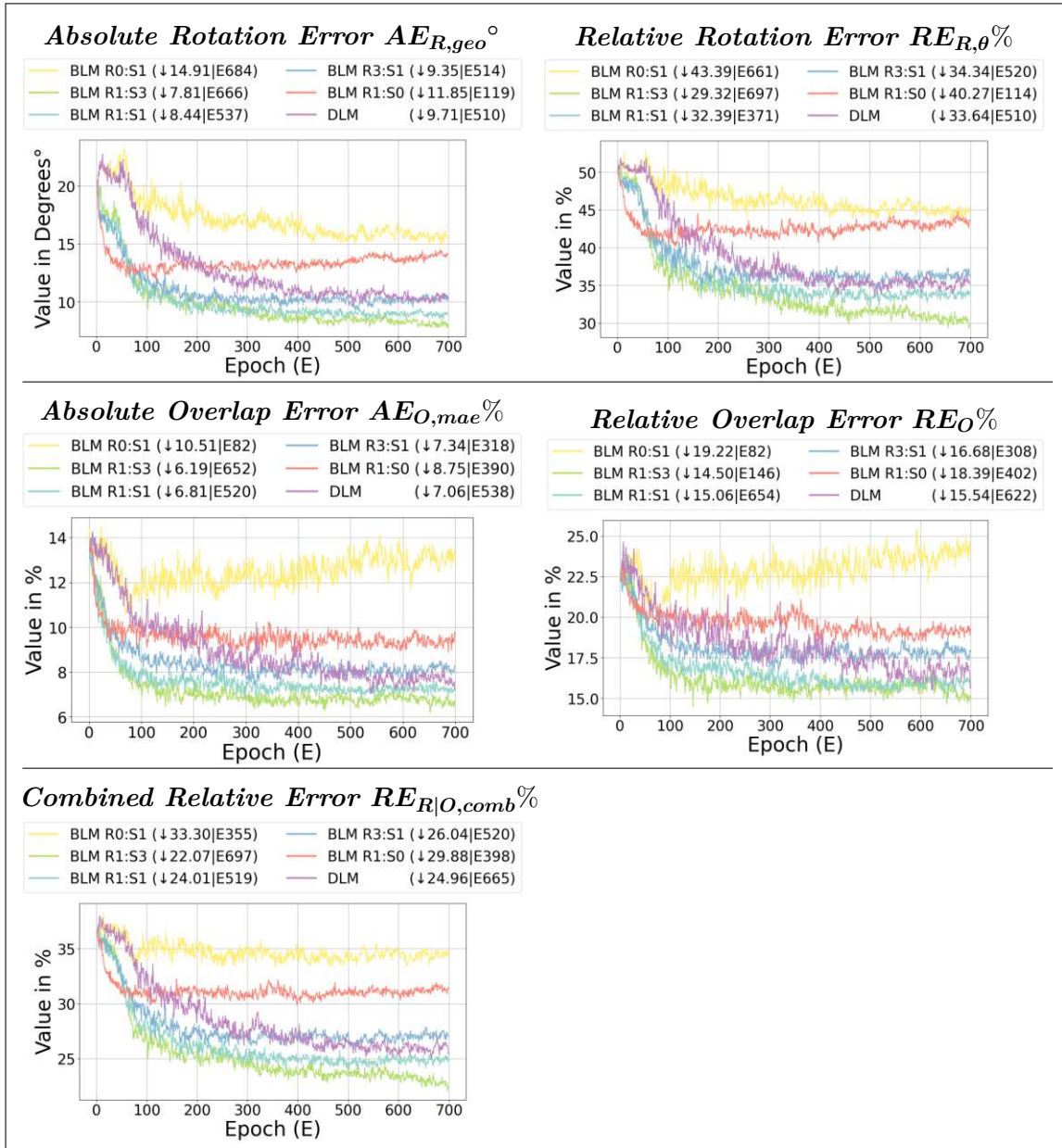


Figure 7.2: Performance evaluation of PTZNet trainings with $n = 2500$ real-world image pairs. Comparison of various Batch-Level Mixing (BLM) ratios: R0:S1 (synthetic-only), R1:S0 (real-world only), R1:S3, R1:S1, and R3:S1 (batch ratio) against Dataset-Level Mixing (DLM). DLM does not refer to a fixed ratio, as the batch elements are randomly drawn from a globally mixed dataset (see Figure 7.1). The metrics are reported in degrees for $AE_{R,geo}$ (Equation 4.8) and in percent for the remainder: $RE_{R,\theta}$ (Equation 4.11), $AE_{O,mae}$ (Equation 4.9), RE_O (Equation 4.14), and $RE_{R|O,comb}$ (Equation 4.16).

Trial Execution for $n = 5000$ Real-World Images

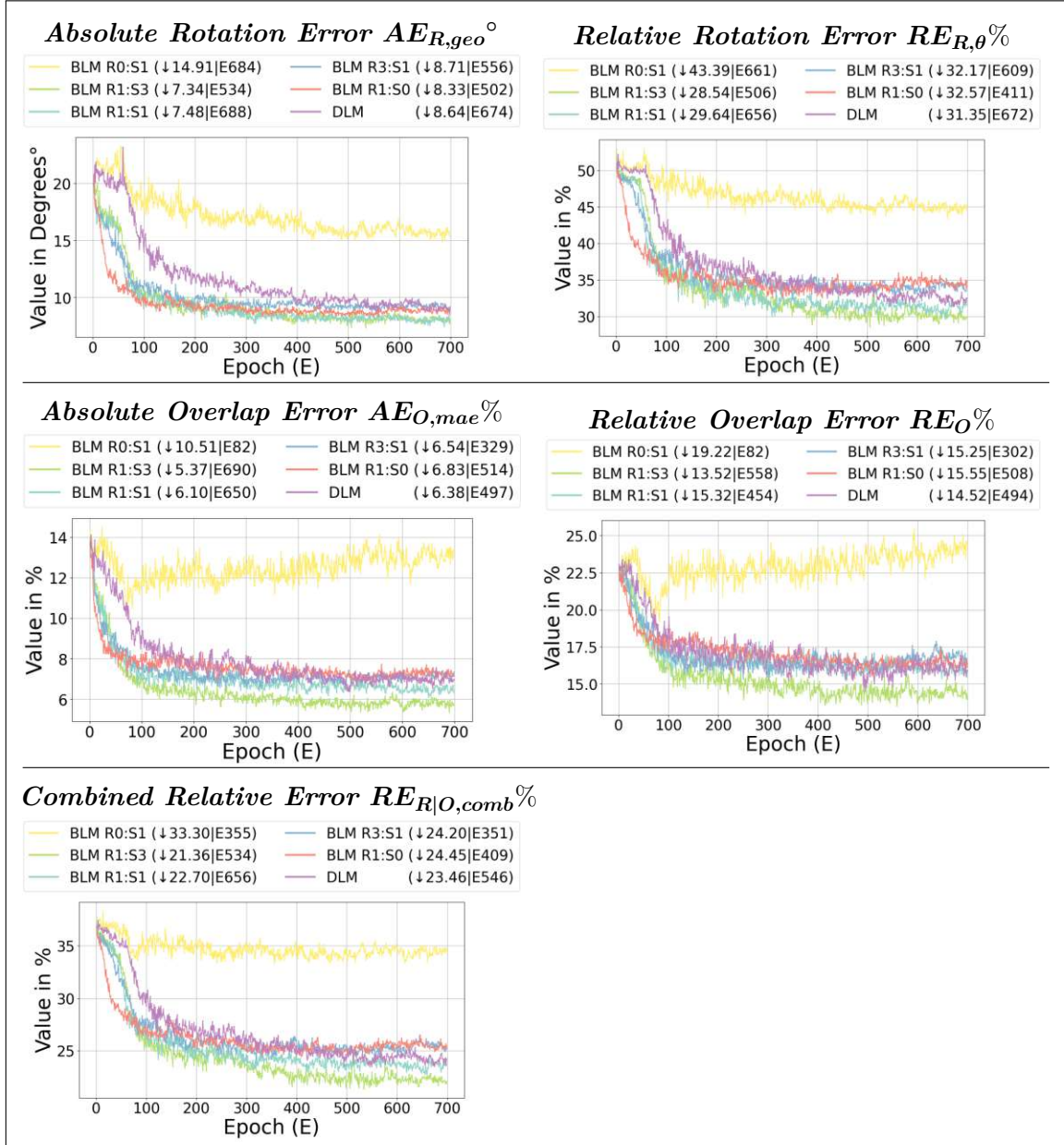


Figure 7.3: Performance evaluation of PTZNet with $n = 5000$ real-world image pairs. The experimental setup and metric definitions correspond to Figure 7.2.

Trial Execution for $n = 7500$ Real-World Images

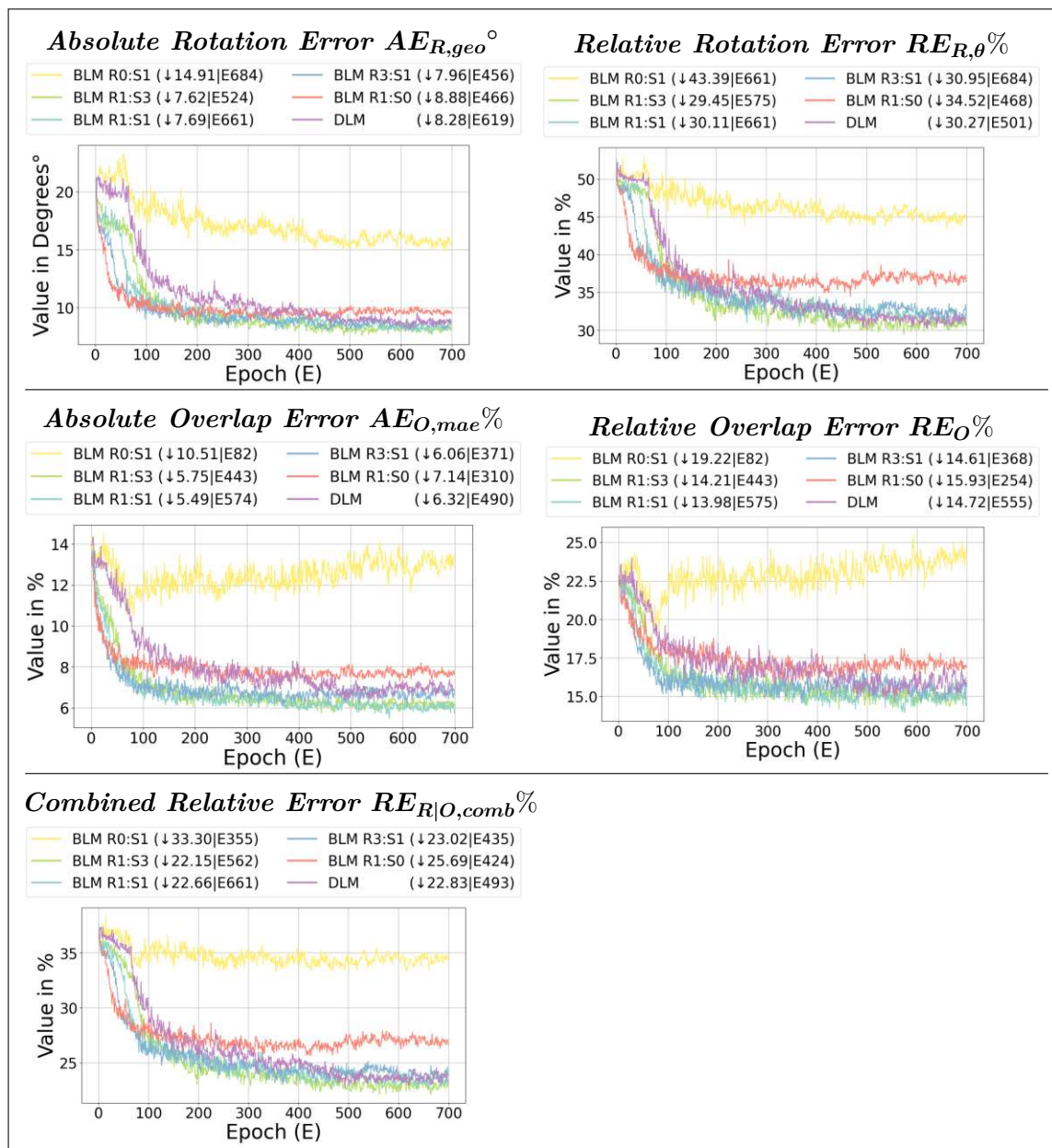


Figure 7.4: Performance evaluation of PTZNet with $n = 7500$ real-world image pairs. The experimental setup and metric definitions correspond to Figure 7.2.

example:

- With $n = 2500$ real-world images, the median absolute rotation error $AE_{R,geo}$ error decreases from
 - 14.9° synthetic only (R0:S1) to
 - 11.9° real-world only (R1:S0) and further to
 - 7.8° BLM (R1:S3).
- A similar pattern holds for the median relative rotation error $RE_{R,\theta}$ metric. Incorporating real-world data steadily improves performance, with the error dropping from
 - 43.5% synthetic only (R0:S1) to
 - 40.3% real-world only (R1:S0) and reaching
 - 29.3% BLM (R1:S3).
- Similar trends are observed for $n = 5000$ and $n = 7500$, where mixed configurations consistently reduce rotation error compared to single-source datasets, although the performance gap to the real-world only baseline (R1:S0) narrows as the amount of available real-world data n increases.

With a median $AE_{R,geo}$ (Equation 4.8) error of 7.8° , the best-performing BLM model (R1:S3) successfully achieves **competitive performance** as defined in Section 3.1.6 with a target range of 5° – 10° . This result supports (H1), particularly given the challenges of training primarily on synthetic data with significant S2R domain gaps (P1) and limited real-world samples (P2).

Then, the relative rotation error $RE_{R,\theta}$ (Equation 4.11) values, which range from 30% to 50% across trials, are influenced by the evaluation framework and the nature of the regression task, as the metric tends to amplify errors when the target rotation (θ_{target}) is small due to its dependence on both θ_{error} and θ_{target} . This sensitivity is particularly noticeable in the synthetic-only training trials (R0:S1), where domain gaps caused by (P1) lead to higher $RE_{R,\theta}$ values. However, mixed configurations, such as DLM or BLM ratios like R1:S3, help mitigate this effect, reducing relative errors to around 30%. At the same time, absolute rotation accuracy ($AE_{R,geo}$) improves significantly in these mixed setups, with geodesic errors consistently below 10° . This highlights that while $RE_{R,\theta}$ provides useful insights into relative performance, it should be considered as an auxiliary metric in addition to the absolute metrics for a more comprehensive evaluation of model accuracy. Nonetheless, these results confirm that the model regresses overlap metrics with competitive performance (Section 3.1.6) when trained on combined datasets, further supporting (H1).

7.2.3 Overlap Accuracy

Also seen in Figures 7.2, 7.3, and 7.4, for overlap regression, evaluated via $AE_{O,mae}$ (Equation 4.9) and RE_O (Equation 4.14), mixed datasets yield substantial improvements, too:

- At $n = 2500$, the median absolute error $AE_{O,mae}$ drops from
 - 10.5% synthetic only (R0:S1) to
 - 8.7% real-world only (R1:S0) and further to
 - 6.1% BLM (R1:S3).
- Similarly, the median relative error RE_O drops from
 - 19.2% synthetic only (R0:S1) to
 - 18.4% real-world only (R1:S0) and further to
 - 14.5% BLM (R1:S3).
- As the number of real-world images increases ($n = 5000$ and $n = 7500$), mixed configurations maintain their advantage.

According to the criteria in Section 3.1.6, the reduction of $AE_{O,mae}$ (Equation 4.9) to 6.1% underlines competitive performance by demonstrating a significant improvement over both single-domain baselines (10.5% and 8.7%). This explicitly validates the effectiveness of the mixed training approach in handling complex outdoor PTZ scenarios (H1).

The relative overlap error RE_O (Equation 4.14) varies between 19.2% and 14.5% across trials. This highlights the network’s sensitivity to variations in overlap magnitude. This effect is especially noticeable in the synthetic-only setting (R0:S1), where the domain gap (P1) amplifies errors. By contrast, mixed configurations such as the BLM ratio R1:S3 reduce relative errors to about 14.5%, showing that the model generalizes better when trained on more diverse data. While RE_O captures relative performance, it should be considered alongside absolute measures like $AE_{O,mae}$ for a nuanced evaluation. These findings confirm that the model can regress overlap metrics with competitive performance (Section 3.1.6) when trained on combined datasets, supporting (H1).

7.2.4 General Observations

Observing the combined median relative error $RE_{R|O,comb}$ (Equation 4.16), shows a similar pattern as observable for the isolated rotational ($AE_{R,geo}$, $RE_{R,\theta}$) and overlap errors ($AE_{O,mae}$ and RE_O), as discussed in Sections 7.2.2 and 7.2.3.

- At $n = 2500$, the combined relative error $RE_{R|O,comb}$ decreases steadily from

- 33.3% synthetic only (R0:S1) to
 - 29.8% real-world only (R1:S0) down to
 - 24.0% BLM (R1:S3).
- This superiority of mixed data trainings stays prevalent with higher numbers of real-world images ($n = 5000$ and $n = 7500$), achieving combined relative errors as low as 21.4%.

The slight performance drop observed when increasing the number of real-world images from $n = 5000$ to $n = 7500$ can possibly be explained with diminishing returns and overrepresentation effects. While a dataset of 5000 image pairs appears to provide sufficient domain realism for learning robust features, expanding this set with additional images from the same cameras introduces redundancy rather than novel information. This limits the model’s ability to benefit from increased data and may even reduce overall generalization due to overfitting to scene-specific details.

Despite the intuition that “more data is better”, our results show that performance slightly decreases beyond this point, with no observable improvement in absolute and relative error metrics. For example, the combined relative error $RE_{R|O,comb}$ (Equation 4.16) remains stable or worsens marginally as we move from 5000 to 7500 image pairs, indicating the network’s learning capacity for domain-specific features has already saturated. The 5000 image configuration thus represents a critical threshold, being large enough to close the domain gap and small enough to preserve the variability introduced by synthetic data.

Still, the consistent performance gains across all metrics, for rotation $AE_{R,geo}$ (Equation 4.8) and $RE_{R,\theta}$ (Equation 4.11), for overlap $AE_{O,mae}$ (Equation 4.9) and RE_O (Equation 4.14), and as the combined relative error $RE_{R|O,comb}$ highlight the importance of leveraging both synthetic and real-world data for training. Synthetic data provides diversity in camera poses and scene configurations, while real-world data ensures domain-specific realism. Even though synthetic data alone does not suffice to achieve desirable performance when evaluating on real-world data, it is shown that incorporating a limited number of real-world images methodologically significantly improves performance while consistently outperforming real-world-only training. Together, they enable the neural network to achieve comparable accuracy in regressing relative rotation and overlap, generally proving (H1).

7.3 Matrix Analysis

In this section, we perform a detailed investigation of this domain adaptation study to evaluate the impact of combining virtually created synthetic and real-world data during training (H2). Specifically, we aim to examine whether leveraging synthetic data alongside real-world data, particularly through Batch-Level Mixing (BLM) strategies, enhances the performance of PTZ rotation and overlap regression models when applied

to real-world scenarios. Furthermore, we explore how varying the quantity of real-world images influences the results.

The previous section (Section 7.2) already hinted at the validity of this statement. To further support this claim, we perform the matrix analysis described in Section 4.4. This provides a cross-comparable picture of the whole domain adaptation study and its distinct trial results.

7.3.1 Data Preparation

The result matrices of our median absolute and relative evaluation errors provide further insights into the performance of rotation and overlap estimation. To quantify improvements, we calculate the relative deviation ΔBL (Equation 7.1) from a chosen baseline $\text{BL} \in \{ \text{R1:S0}, \text{DLM} \}$ for trials with the same number of real-world samples n . Given the error of the current trial E_{current} and the baseline error E_{baseline} , the deviation is computed as:

$$\Delta\text{BL} = \frac{E_{\text{current}} - E_{\text{baseline}}}{E_{\text{baseline}}} \times 100 \quad (7.1)$$

Consequently, the figure for each metric (e.g., Figure 7.5 or 7.6) comprises three matrices: the first row displays the actual errors, while the second row presents two ΔBL matrices derived via Equation 7.1. One compares the trial to the real-only baseline $\Delta\text{R1:S0}$ and the other to the Dataset-Level Mixing (DLM) baseline ΔDLM . Together, these allow for a clear assessment of performance in relation to our sub-hypotheses: (H2.1), (H2.2), and (H2.3).

As a first step, we gather all the data from our trials and fill our error-related matrices:

- ***Absolute Rotation Error $AE_{R,geo}$ in Degrees $^\circ$***
 - Equation 4.8
 - Figure 7.5
- ***Relative Rotation Error $RE_{R,\theta}$ in %***
 - Equation 4.11
 - Figure 7.6
- ***Absolute Overlap Error $AE_{O,mae}$ in %***
 - Equation 4.9
 - Figure 7.7
- ***Relative Overlap Error RE_O in %***
 - Equation 4.14

– Figure 7.8

- **Combined Relative Error $RE_{R|O,comb}$ in %**

– Equation 4.16

– Figure 7.9

From those figures, it is already evident that, across all values of n , the batch-mixing strategy consistently outperforms other mixing strategies as well as isolated training approaches. It achieves the lowest error rates (marked in bold in Figures 7.5, 7.6, 7.7, 7.8 and 7.9) while demonstrating significant improvements over real-world-only training ($\Delta R1:S0$) and Dataset-Level Mixing (DLM) strategies. While the performance advantage of Batch-Level Mixing (BLM) (R1:S3 and R1:S1 ratios) diminishes slightly as n increases, the improvements remain observable and impactful.

These initial observations strongly suggest that our hypothesis (H2) holds validity. Still, we now transition into a detailed discussion.

<i>Absolute Rotation Error $AE_{R,geo}$ in Degrees$^\circ$</i>						
$n \setminus r$	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	14.91	7.81	8.44	9.35	11.85	9.71
5000	14.91	7.34	7.48	8.71	8.33	8.64
7500	14.91	7.62	7.69	7.96	8.88	8.28

<i>Relative Deviation from $\Delta R1:S0$ in %</i>						
$n \setminus r$	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	25.83	-34.14	-28.75	-21.07	0.0	-18.09
5000	79.1	-11.82	-10.22	4.6	0.0	3.8
7500	67.99	-14.16	-13.42	-10.3	0.0	-6.69

<i>Relative Deviation from ΔDLM in %</i>						
$n \setminus r$	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	53.62	-19.59	-13.01	-3.63	22.09	0.0
5000	72.54	-15.04	-13.5	0.77	-3.66	0.0
7500	80.03	-8.0	-7.21	-3.87	7.17	0.0

Figure 7.5: Result matrices for $AE_{R,geo}^\circ$ (Equation 4.8), containing the median absolute geodesic rotation errors for all test configurations. The rows contain the results for the respective real-world image pair amount $n = 2500..7500$, with the lowest error for each n marked in bold. Sampling strategies include: synthetic only (R0:S1), mixed synthetic–real batch ratios (R1:S3, R1:S1, R3:S1), real-world only (R1:S0), and Dataset-Level Mixing (DLM). Values in the first row’s matrix correspond to the actual errors, while the two matrices in the second row report the relative deviations to the real-world only ($\Delta R1:S0$) and DLM (ΔDLM) baselines, calculated according to Equation (7.1).

<i>Relative Rotation Error $RE_{R,\theta}$ in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	43.39	29.32	32.39	34.34	40.27	33.64
5000	43.39	28.54	29.64	32.17	32.57	31.35
7500	43.39	29.45	30.11	30.95	34.52	30.27

<i>Relative Deviation from $\Delta R1:S0$ in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	7.76	-27.19	-19.56	-14.72	0.0	-16.46
5000	33.23	-12.37	-8.99	-1.24	0.0	-3.74
7500	25.7	-14.68	-12.79	-10.33	0.0	-12.32

<i>Relative Deviation from ΔDLM in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	29.0	-12.85	-3.71	2.08	19.71	0.0
5000	38.4	-8.97	-5.46	2.6	3.88	0.0
7500	43.36	-2.69	-0.53	2.27	14.05	0.0

Figure 7.6: Result matrices for $RE_{R,\theta}\%$ (Equation 4.11), containing the relative geodesic rotation errors for all test configurations, similar to Figure 7.5.

<i>Absolute Overlap Error $AE_{O,mae}$ in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	10.51	6.19	6.81	7.34	8.75	7.06
5000	10.51	5.37	6.1	6.54	6.83	6.38
7500	10.51	5.75	5.49	6.06	7.14	6.32

<i>Relative Deviation from $\Delta R1:S0$ in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	20.12	-29.17	-22.18	-16.1	0.0	-19.23
5000	53.8	-21.44	-10.71	-4.33	0.0	-6.59
7500	47.14	-19.51	-23.16	-15.09	0.0	-11.46

<i>Relative Deviation from ΔDLM in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	48.72	-12.31	-3.65	3.87	23.8	0.0
5000	64.64	-15.9	-4.42	2.41	7.05	0.0
7500	66.19	-9.1	-13.21	-4.1	12.95	0.0

Figure 7.7: Result matrices for $AE_{O,mae}\%$ (Equation 4.9), containing the absolute overlap errors for all test configurations, similar to Figure 7.5.

<i>Relative Overlap Error RE_O in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	19.22	14.5	15.06	16.68	18.39	15.54
5000	19.22	13.52	15.32	15.25	15.55	14.52
7500	19.22	14.21	13.98	14.61	15.93	14.72

<i>Relative Deviation from $\Delta R1:S0$ in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	4.52	-21.19	-18.12	-9.3	0.0	-15.51
5000	23.66	-13.04	-1.46	-1.92	0.0	-6.57
7500	20.71	-10.77	-12.21	-8.29	0.0	-7.59

<i>Relative Deviation from ΔDLM in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	23.7	-6.72	-3.09	7.35	18.35	0.0
5000	32.35	-6.93	5.46	4.97	7.03	0.0
7500	30.62	-3.44	-5.0	-0.75	8.21	0.0

Figure 7.8: Result matrices for $RE_O\%$ (Equation 4.14), containing the relative overlap errors for all test configurations, similar to Figure 7.5.

<i>Combined Relative Error $RE_{R O,comb}$ in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	33.3	22.07	24.01	26.04	29.88	24.96
5000	33.3	21.36	22.7	24.2	24.45	23.46
7500	33.3	22.15	22.66	23.02	25.69	22.83

<i>Relative Deviation from $\Delta R1:S0$ in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	11.45	-26.13	-19.64	-12.85	0.0	-16.46
5000	36.18	-12.66	-7.18	-1.05	0.0	-4.05
7500	29.61	-13.79	-11.81	-10.41	0.0	-11.13

<i>Relative Deviation from ΔDLM in %</i>						
n \ r	R0:S1	R1:S3	R1:S1	R3:S1	R1:S0	DLM
2500	33.41	-11.58	-3.81	4.32	19.7	0.0
5000	41.93	-8.97	-3.26	3.13	4.23	0.0
7500	45.84	-3.0	-0.76	0.81	12.52	0.0

Figure 7.9: Result matrices for $RE_{R|O,comb}\%$ (Equation 4.16), containing the combined rotation and overlap errors for all test configurations, similar to Figure 7.5.

7.3.2 Discussion

In this section, we discuss the results presented in Table 7.1 and evaluate how these findings support our hypotheses. This table does not introduce new data but summarizes the trial candidates with the lowest errors with their corresponding relative improvements over the baselines, providing the quantitative basis necessary to evaluate our hypotheses. It allows the reader to focus on the most relevant data points already visible in the error matrices of the first row, and their corresponding relative-deviation matrices in the second row, presented in Section 7.3.1 in Figures 7.5-7.9. These relative deviation matrices are calculated according to Equation 7.1, which shows each trial’s improvement over the real-only ($\Delta R1:S0$) and DLM (ΔDLM) baselines. This highlights how mixed data strategies, especially via BLM, consistently provide measurable improvements, demonstrating the practical value of our approach.

Metric	Column	n	Error	$\Delta R1:S0$ (% \uparrow)	ΔDLM (% \uparrow)
$AE_{R,geo}$ ($^\circ$)	R1:S3	2500	7.81	-34.14	-19.59
$AE_{R,geo}$ ($^\circ$)	R1:S3	5000	7.34	-11.82	-15.04
$AE_{R,geo}$ ($^\circ$)	R1:S3	7500	7.62	-14.16	-8.00
$RE_{R,\theta}$ (%)	R1:S3	2500	29.32	-27.19	-12.85
$RE_{R,\theta}$ (%)	R1:S3	5000	28.54	-12.37	-8.97
$RE_{R,\theta}$ (%)	R1:S3	7500	29.45	-14.68	-2.69
$AE_{O,mae}$ (%)	R1:S3	2500	6.19	-29.17	-12.31
$AE_{O,mae}$ (%)	R1:S3	5000	5.37	-21.44	-15.90
$AE_{O,mae}$ (%)	R1:S1	7500	5.49	-23.16	-13.21
RE_O (%)	R1:S3	2500	14.50	-21.19	-6.72
RE_O (%)	R1:S3	5000	13.52	-13.04	-6.93
RE_O (%)	R1:S1	7500	13.98	-12.21	-5.00
$RE_{R O,comb}$ (%)	R1:S3	2500	22.07	-26.13	-11.58
$RE_{R O,comb}$ (%)	R1:S3	5000	21.36	-12.66	-8.97
$RE_{R O,comb}$ (%)	R1:S3	7500	22.15	-13.79	-3.00

Table 7.1: Effect of mixed dataset training (strategic mixing of synthetic and real-world samples, as visualized in Figure 7.1) relative to both real-world only (R1:S0) and Dataset-Level Mixing (DLM) baselines across all metrics and real-world sample sizes n . The table summarizes the best-performing trial results, primarily mixed-data strategies, for each metric (mainly R1:S3, but also R1:S1) and compares their median error values to the two baselines. The lowest median error (Error column) within each metric group is highlighted in bold, as are the greatest relative improvements (lowest negative value) in the $\Delta R1:S0$ and ΔDLM columns. These relative deviations, computed according to Equation 7.1, quantify the percentage error improvement achieved by the mixed strategy compared to the baselines. All values are directly selected from the previously presented error matrices (Figures 7.5–7.9).

Optimal Mixing Ratio for Evaluated Metrics

Inspecting the primary error matrices (Figures 7.5–7.9) shows that, across all evaluation metrics, the best results are obtained by the batch-level R1:S3 strategy at $n = 5000$ real-world image pairs. The same pattern is reflected in the Error column of Table 7.1, where the R1:S3 configuration at $n = 5000$ achieves the lowest median error for each metric across the three real-world sample sizes $n \in \{2500, 5000, 7500\}$. This directly supports Hypothesis (H2.1), which argues that combining synthetic and real-world data improves model performance over training with either data source alone.

Effect of Mixing Strategies Relative to Real-World Only Training

The Δ R1:S0 column (Equation 7.1) in Table 7.1 summarizes, for all evaluation metrics and all three real-world sample sizes $n \in \{2500, 5000, 7500\}$, how the best performing data mix strategies compare to real-world only training. Across both rotation and overlap metrics, batch-level R1:S3 mixing consistently yields notable improvements at $n = 2500$ real-world image pairs, with the largest relative gains (bold Δ R1:S0 entries) between roughly 21 % and 35 %, and still provides positive improvements at $n = 5000$ and $n = 7500$.

The R1:S1 configuration becomes competitive for the overlap metrics $AE_{O,mae}$ (Equation 4.9) and RE_O (Equation 4.14) at $n = 7500$. For example, it outperforms R1:S3 on the Δ R1:S0 of $AE_{O,mae}$ with 23.16% (over 19.51% of R1:S3, see Figure 7.7) and Δ R1:S0 of RE_O of 12.21% (over 10.77% of R1:S3, see Figure 7.8). While R1:S3 remains the dominant strategy across the majority of metrics and sample sizes, this isolated advantage for R1:S1 suggests that as real-data volume grows, the optimal synthetic ratio may shift slightly for specific sub-tasks, supporting Hypotheses (H2.1) and (H2.3). This shows that strategically mixing synthetic with real-world data at the batch-level (via BLM) consistently outperforms a real-world only baseline, and thus, the potential to mitigate real-world scarcity by incorporating a surplus of synthetic data.

Effect of Batch-Level Mixing Relative to Dataset-Level Mixing

The Δ DLM column (Equation 7.1) in Table 7.1 enables a direct comparison of BLM versus DLM across all metrics. For all three real-world sample sizes $n \in \{2500, 5000, 7500\}$, a BLM of R1:S3 outperforms DLM for rotation and overlap metrics, with relative gains of roughly 6–20 % at $n = 2500$ and $n = 5000$ real-world image pairs, as indicated by the bold Δ DLM entries, and smaller but still positive gains at $n = 7500$.

The R1:S1 ratio becomes competitive at the largest real-world sample size $n = 7500$, outperforming R1:S3 on the overlap metrics: the Δ DLM of $AE_{O,mae}$ (Equation 4.9) achieves 13.21% (compared to 9.10%, see Figure 7.7), and the Δ DLM of RE_O (Equation 4.14) reaches 5.00% (versus 3.44%, see Figure 7.8). This confirms the advantage of BLM over DLM. While the synthetic-heavy R1:S3 ratio provides the strongest general performance, the mixing strategy remains effective even when the synthetic ratio is reduced to R1:S1.

These results directly support Hypothesis (H2.2) and show that the advantage of BLM extends beyond a single combined metric to all individual median error measures. This also shows that the strategic integration of synthetic data in the training pipeline (via BLM) outperforms a naive mixing on the global level (DLM).

Impact of Increasing Real-World Data

Increasing the real-world image pair sample size from $n = 2500$ to $n = 7500$ yields noticeable convergence in performance across the R1:S3 and R1:S1 strategies. While the relative gains over the real-only baseline $\Delta R1:S0$ diminish as the volume of real data grows, the inclusion of synthetic data remains consistently beneficial, confirming Hypothesis (H2.3).

The previously noted shift where the balanced R1:S1 ratio becomes competitive at $n = 7500$, surpassing R1:S3 in overlap improvements (e.g., 23.16% vs. 19.51% for $AE_{O,mae}$, Figure 7.7), indicates a change in training dynamics. As real-world data becomes sufficient to stabilize training, the necessity for strong synthetic regularization decreases. Nevertheless, the continued dominance of R1:S3 in most metrics suggests that even with larger real-world datasets, a synthetic-heavy mix provides the most robust error reduction. We conclude that, also when increasing the amount of real-world image pairs n , the performance benefits achieved by strategically incorporating synthetic data remain, even though the relative advantage over the other methods narrows.

Summary

In conclusion, all sub-hypotheses (H2.1), (H2.2), and (H2.3) are well supported by our results. We therefore confirm the main hypothesis (H2): training with synthetic data significantly improves performance on real-world PTZ rotation and overlap estimation, successfully reaching competitive performance (Section 3.1.6), especially when combined at the batch-level with an R1:S3 R2S ratio. This mixing strategy effectively addresses the domain gap while optimizing model generalization, particularly when only limited amounts of real-world data are available.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Conclusion and Future Work

As a foundation for our concluding discussion, we first recall our research questions addressed in this thesis:

(Q1) Can we use neural networks to achieve competitive performance (Section 3.1.6) for PTZ displacement and overlap estimation by comparing two images from a single PTZ camera?

(Q2) Can we mainly rely on virtually created synthetic data for training of (Q1)? In particular:

(Q2.1) How does purely synthetic training data affect inference performance on real-world images, in terms of rotation and overlap prediction (P1)?

(Q2.2) How do different R2S data batch mixes during training impact prediction accuracy, compared to simply merging the two datasets (P1)?

(Q2.3) When varying the absolute number of real-world samples, how much does adding synthetic data help (P2)?

These questions are closely related to these underlying research problems:

(P1) The domain gap introduced by synthetic PTZ data negatively impacts the performance of neural networks in regressing relative rotation and overlap for real-world image pairs. Addressing this gap requires a systematic evaluation of domain adaptation strategies, including the optimal balance and mixing of synthetic and real-world data.

(P2) For PTZ cameras, gathering huge amounts of diverse real-world images including the required metadata like camera intrinsics and extrinsics is expensive and complex. Specialized equipment is required and cameras have to be remounted in different, varying scenes.

8.1 Hypotheses Validation

Building on the research questions and problems outlined above, we conducted a systematic experimental evaluation (Chapter 7). We evaluated the performance of neural networks for PTZ rotation and overlap regression when trained mainly on virtually created synthetic data, specifically addressing the resulting Synthetic-To-Real (S2R) domain gap. Our findings underline our hypotheses (see Table 8.1) and propose relevant insights regarding the effective combination of synthetic and real-world data during training.

Hypothesis	Description	✓
(H1)	We can regress PTZ rotation and overlap displacement with competitive performance (Section 3.1.6), mainly relying on virtually created synthetic data	✓
(H2)	Batch-Level Mixing of synthetic and real-world data achieves best performance, compared to naively mixing data on global level (Dataset-Level Mixing) or the isolated real-world only baseline (R1:S0), see Figure 7.1	✓
(H2.1)	Integrating both virtually created synthetic and real-world data for training improves performance compared to using only synthetic or only real-world data, with synthetic-only training performing weakest	✓
(H2.2)	Strategic Batch-Level Mixing outperforms Dataset-Level Mixing	✓
(H2.3)	Virtually created synthetic data positively affects performance even with increasing number of real data	✓

Table 8.1: Evaluation of hypotheses with descriptions and confirmation.

Below, we summarize the key results from our analysis:

For (H1), as shown in Section 7.2, our proposed Pan, Tilt and Zoom (PTZ) rotation and overlap estimation model achieved reasonable performance for outdoor scenes. This confirms that a neural network trained on a combination of synthetic and real-world data can effectively handle this task.

For (H2), the matrix analysis in Section 7.3 confirms all sub-hypotheses:

- Regarding (H2.1), as expected, models trained purely on synthetic data performed the worst. However, combining synthetic and real-world data consistently outperformed using only real-world data. This demonstrates that virtually created synthetic data can be a helpful complement.
- Regarding (H2.2), Batch-Level Mixing (BLM) strategies consistently delivered superior results compared to Dataset-Level Mixing (DLM) or pure real-world training (R1:S0), particularly when using the R1:S3 ratio. While Batch-Level Mixing (BLM) allows for defining specific mixing ratios (such as 1 part real to 3

parts synthetic) enforced within every batch, Dataset-Level Mixing (DLM) operates without a fixed per-batch ratio, as samples are drawn randomly from the globally combined dataset (see Figure 7.1).

- Regarding (H2.3), increasing the number of real-world images narrows the performance gap gained from strategically incorporating synthetic data. Nonetheless, even with higher amounts of real-world data, synthetic data continued to provide performance improvements (see Table 7.1).

These combined findings collectively confirm (H2): Synthetic data significantly enhances inference performance when real-world training data is limited, with BLM strategies yielding the highest performance.

8.2 Contributions

Accordingly, the thesis delivers the following key contributions:

8.2.1 S2R Gap Analysis (C1)

In Chapter 7, we present a quantitative investigation of the domain gap introduced by synthetic data when used for PTZ camera model training. By performing controlled experiments with varying batch ratios of synthetic and real-world data, and evaluating across multiple absolute and relative error metrics (as proposed in Chapter 4), we showcase the impact of synthetic-only training on real-world inference (see Table 7.1). For example, switching from a real-world only batch composition to our best performing synthetic and real-world image pair ratio, being one part real to three parts synthetic (R1:S3), reduces error rates by up to 35%. This analysis enables a clear understanding of how strongly domain shifts affect PTZ estimation accuracy in practical settings (P1).

8.2.2 Data Mixing Framework (C2)

A Batch-Level Mixing (BLM) strategy is proposed in Chapter 4 and empirically validated in Chapter 7. Specifically, mixing at an R1:S3 Real-To-Synthetic (R2S) ratio consistently delivers lower error rates compared to both dataset-level mixing and real-only training, as demonstrated in Table 7.1. In particular, this R1:S3 R2S refers to a batch consisting of one part real-world data and three parts virtually created synthetic data, randomly mixed.

For instance, BLM achieves rotation and overlap error rates as low as 7.34° for $AE_{R,geo}$ (Equation 4.8) and 5.37% for $AE_{O,mae}$ (Equation 4.9) at $n = 5000$ real-world image pairs. This corresponds to a 11.82% reduction in rotation errors and 21.44% reduction in overlap errors compared to real-only baselines $\Delta R1:S0$ (Equation 7.1), and a 15.04% reduction in rotation errors and 15.90% reduction in overlap errors compared to the DLM baseline. The biggest improvement over the real-world baseline $\Delta R1:S0$ is achieved

with $n = 2500$, where the $AE_{R,geo}$ of 7.81° improves by 34.14% and $AE_{O,mae}$ of 6.19% improves by 29.17%.

Our results also indicate that simply opting for a balanced R1:S1 batch ratio between synthetic and real-world data, as done by Lee et al. [LPYL20] and Wu et al. [WLX⁺20], might not be optimal if the datasets differ in size (P1). With high numbers of virtually created synthetic data over limited numbers of real-world images, our R1:S3 R2S batch ratio delivered the best results in 13/15 cases shown in Table 7.1. Only the overlap performance, specifically for $AE_{O,mae}$ (Equation 4.9) and RE_O (Equation 4.14), was outperformed in one scenario by the R1:S1 mix, namely when using the highest amount of real-world data ($n = 7500$).

8.2.3 Quantitative Benefit of Synthetic Data (C3)

Our study provides quantitative evidence that supporting training with an abundance of virtually created synthetic data enhances model generalization for PTZ regression, especially where real data is limited or costly to obtain. This is demonstrated in Chapter 5 and Chapter 7. Mixed strategies outperform real-only training at all sample sizes. For example, viewing column $\Delta R1:S0$ (Equation 7.1) in Table 7.1, being the relative deviation to the R1:S0 baseline, the absolute rotation error $AE_{R,geo}$ (Equation 4.8) could be reduced by 14.16% at $n = 7500$ real-world image pairs with R1:S3 mix, and the absolute overlap error $AE_{O,mae}$ (Equation 4.9) by 23.16% at $n = 7500$ in the R1:S1 mixing case. The benefit is most prominent at lower real data quantities, with diminishing, but still present, returns as real data volume increases (P2).

8.2.4 PTZNet (C4)

Lastly, we developed PTZNet, a Siamese neural network designed to estimate relative rotation and overlap from pairs of PTZ camera images. Its implementation is described in Chapter 6.

When trained with our controlled BLM strategy (proposed in Chapter 4), PTZNet achieves a median geodesic error $AE_{R,geo}$ (Equation 4.8) below 8° and a Mean Absolute Error (MAE) $AE_{O,mae}$ (Equation 4.9) below 6.2% on real-world evaluation tasks (Chapter 7). This level of accuracy satisfies the competitive performance requirements (Section 3.1.6) in terms of a rotational error within $5^\circ - 10^\circ$ and significantly reduces the overlap error compared to single-domain baselines. The network’s architecture and mixed-domain training approach make it suitable for practical PTZ estimation when the amount of annotated real-world data is limited (Q1).

8.3 Limitations

Although controlled BLM of synthetic and real-world data improves PTZ estimation, several limitations remain.

8.3.1 Data Constraints

The study primarily relies on limited real-world datasets for training and evaluation. While synthetic data was extensively used to augment training, pre-training PTZNet on large synthetic datasets followed by fine-tuning on smaller real-world sets was not systematically explored. Consequently, performance may be constrained by the availability and diversity of real-world data. Furthermore, variations in the zoom parameter of PTZ cameras were omitted due to their complexity and effect on camera intrinsics, limiting robustness to dynamic zoom levels.

8.3.2 Methodological Scope

This work demonstrates that BLM effectively mitigates the S2R domain gap. However, alternative domain adaptation techniques, including discrepancy-based, adversarial-based, and reconstruction-based methods, were not evaluated, and it remains an open question how they would compare to or complement BLM.

8.3.3 Task Specifics

The evaluation of BLM is restricted to relative PTZ rotation and overlap regression. While results are robust within this domain, the generalizability of the approach to other regression-based domain adaptation tasks or different camera setups is untested. As such, findings should be interpreted within the context of PTZ estimation tasks.

8.3.4 Practical Considerations

PTZNet achieves robust performance in controlled evaluations and was tested with a feasible level of environmental variation. However, practical deployment challenges persist: only two real-world cameras from the same vendor were used, and acquiring images across all seasons, lighting conditions, and weather scenarios is challenging. Consequently, model performance under broader real-world conditions has not been evaluated and may vary.

8.4 Future Work

Building on these limitations, several ideas are proposed to further tighten the S2R domain gap of PTZ rotation and overlap estimation with synthetic data.

Firstly, pre-training PTZNet on large synthetic datasets followed by fine-tuning on smaller real-world sets, could provide additional insight. Controlled comparisons should assess whether such a strategy outperforms BLM, or whether BLM remains superior when real-world data is limited.

Secondly, alternative domain adaptation techniques could complement the current approach. Discrepancy-based, adversarial-based, and reconstruction-based domain adaptation methods offer potential for reducing the S2R domain gap, while coming with their

own limitations such as feature distortion or task-specific tuning. Investigating these methods in combination with BLM may improve generalization for relative PTZ rotation and overlap estimation.

Thirdly, the zoom parameter of PTZ cameras was omitted in this study due to the inherent complexity of varying zoom levels and their impact on camera intrinsics. Systematically incorporating zoom variations remains an interesting topic, since zoom can affect both rotation and overlap estimation. In this context, BLM may mitigate the S2R gap by combining synthetic data generated at multiple zoom levels with real-world samples, enabling the network to learn more robust representations across zoom settings.

Finally, results from previous studies by Lee et al. [LPYL20] and Wu et al. [WLX⁺20] could be re-evaluated using optimized BLM ratios. More generally, controlled BLM may improve performance in other regression-based domain adaptation tasks beyond PTZ estimation.

8.5 Summary

The systematic evaluation presented in this thesis demonstrates that controlled Batch-Level Mixing (BLM) of virtually created synthetic and real-world data is an effective strategy to improve the performance of PTZ rotation and overlap estimation, particularly under conditions of limited real-world data. Specifically, BLM (primarily at an R1:S3 Real-To-Synthetic (R2S) ratio) consistently reduced rotation error by up to 35% and overlap error by up to 30% relative to real-only baselines, outperforming DLM in all 15 evaluated scenarios (see Table 7.1).

PTZNet, trained using this controlled BLM strategy, achieves in our best-performing configuration below 7.5° median geodesic rotation error $AE_{R,geo}$ (Equation 4.8) and 5.5% overlap Mean Absolute Error (MAE) $AE_{O,mae}$ (Equation 4.9) on real-world evaluation sets, confirming that synthetic data can significantly enhance model generalization when real-world data is limited. These results provide strong empirical evidence that the strategic combination of synthetic and real data mitigates the S2R domain gap and supports robust usage of neural networks for PTZ estimation tasks.

Overall, the framework developed in this work delivers practical performance gains and establishes a clear methodology for future work, including integration with alternative domain adaptation techniques and zoom-aware modeling, thereby advancing the SOTA in regression-based domain adaptation for camera pose estimation.

Overview of Generative AI Tools Used

Following the TU Wien's guidelines on the use of generative AI in academic theses of 2025, I am affirming the following use of such tools while writing this work:

Grammarly

- **Source:** <https://www.grammarly.com/>
- **Purpose:** Assisted with grammar, spelling, and style suggestions.
- **Usage:** Grammarly was active as a browser plugin during drafting. It automatically highlights issues and suggested changes. Those were carefully reviewed. The final decision to accept, modify, or reject any suggestion was made by me to ensure correctness and consistency.

DeepL Translator

- **Source:** <https://www.deepl.com/>
- **Purpose:** Assisted with translation from German to English.
- **Usage:** Drafts written in German were translated using DeepL. Multiple translation options are proposed, but the final English phrasing was independently reworded and refined by me to maintain technical accuracy and scientific tone.

ChatGPT 4o (Writing Assistance)

- **Source:** <https://www.chatgpt.com/>
- **Purpose:** Provided suggestions for clarity, conciseness, and flow.

- **Usage:** My own drafted paragraphs were submitted as prompts when requesting for improved wording, grammar checking or reducing redundant formulations. All responses were checked to not add new information. All resulting suggestions were thoroughly reworked and integrated by me, ensuring the integrity of the final text.

ChatGPT 4o (Coding Support)

- **Source:** <https://www.chatgpt.com/>
- **Purpose:** Assisted with code drafting, debugging, and technical clarifications.
- **Usage:** Descriptions of technical issues and code snippets were submitted for bug-fixing or improvement suggestions. Any code solutions offered by ChatGPT were revised and incorporated as needed, with all core logic, structure, and implementation decisions made by me.

Perplexity AI

- **Source:** <https://www.perplexity.ai>
- **Purpose:** Tool for literature discovery.
- **Usage:** Perplexity AI was used to identify additional research papers relevant to the thesis topic. All literature suggested by Perplexity AI was carefully reviewed based on relevance, quality, and validity.

Scope and Attribution of AI Assistance

AI tools were used only as aids for drafting, rewording, translation, or technical troubleshooting. No AI-generated text, tables, code, or figures were included verbatim or with minimal changes. All academic content, formulations, analyses, and arguments were independently written and thoroughly reworked by me.

List of Figures

1.1	Left: a camera observing a foggy highway in northern India. Right: a camera surveying a crosswalk in Shinjuku, Tokyo, Japan. These images illustrate diverse traffic monitoring locations. Images from [New25, Sky25].	1
1.2	Left: front view of a PTZ camera mounted on a pole. Right: back view, also showing parts of the camera's FoV. Images from [Gmb25, Tea23].	2
2.1	Hierarchical relationship of artificial intelligence, machine learning, and deep learning. Image from [VSvLD ⁺ 20] page 27.	8
2.2	Schema of a single ANN neuron, where weighted inputs x_i with weights w_{ij} are linearly summed by the transfer function \sum into net_j , then with the bias θ_j passed through the non-linear activation function φ to produce the output o_j . Image from [i2t19].	10
2.3	Comparison of common ANN activation functions, showing their mathematical definitions and response graphs, including sigmoid, hyperbolic tangent, ReLU, leaky ReLU, maxout function, and ELU. Image from [Jad18] page 1.	10
2.4	Illustration of a deep feedforward ANN architecture, with an input layer of n units, multiple fully connected hidden layers h_1, h_2, \dots, h_L , and an output layer of o units. Image from [Lav25].	11
2.5	Gradient descent trajectory on a two-dimensional cost surface, showing iterative steps from the initial parameter estimate to a local minimum. Image from [Xia19].	13
2.6	CNN architecture (a) where an input image is processed by convolutional layers with an intermediate pooling step to produce feature maps, followed by a fully connected layer for prediction. The convolution applies a sliding kernel (b) over the input to extract local features, max pooling (c) down-samples each feature map, and the flattened feature vector is fed into a multilayer perceptron (d) to generate the final output. Image from [AMS ⁺ 22] page 3721.	14
2.7	SNN architecture with two identical CNN-based backbones sharing weights and parameters. Each branch encodes one input into a common embedding space, and a distance metric compares their embeddings to learn a similarity function. Image from [ZFCB ⁺ 22] page 82.	15
		115

2.8	Illustration of the theoretical mechanical rotation capabilities of a PTZ camera. (1) The camera’s internal coordinate system, with the Y -axis serving as the up vector. (2) Tilt rotation about the X -axis. (3) Pan rotation about the Y -axis. (4) Roll rotation about the optical Z -axis, a degree of freedom not supported by all camera models. Roll should not be confused with zooming, which modifies the camera’s intrinsic parameters.	17
2.9	Schematic representation of unit quaternions $\mathbf{q}_{0..2}$ located on S^3 , with their axes X, Y, Z and fourth dimension W . The antipodal $-\mathbf{q}_2$ of \mathbf{q}_2 shows the double-cover property [Tri09], as both quaternions represent the same rotation in $SO(3)$. Image source: [Lei22].	20
2.10	Step-by-step transformation of a rectangular FoV from its default position on the sphere to its actual orientation using PTZ parameters. (a) shows the initial spherical coordinates, (b) visualizes the polygon in Cartesian space, (c) applies rotation based on PTZ angles, and (d) maps the result back to spherical coordinates.	24
2.11	A solid angle Ω (measured in steradians) formed by a cone on a unit sphere, covering a curved surface area A . Image source: [Met25].	25
2.12	Unit spheres visualising relative PTZ camera FoVs (blue and red), projected radially outward from each sphere’s center. The intersection areas (green) represent the spherical IoU across three distinct scenarios: partial overlap (Sphere 1), high overlap (Sphere 2), and no overlap (Sphere 3).	26
3.1	Multi-view camera pose estimation with camera rotation R and translation T . Image source: [HK18] page 94.	28
3.2	Keypoint detection identifies distinctive features in both images (red dots), and feature matching connects corresponding points across the images (green lines), enabling accurate alignment and comparison. Image source: [RRKB11] page 1	29
3.3	Camera FoV intersection examples with various camera mounting locations. Image source: [AEBM19] page 16	32
3.4	Illustration of domain adaptation, showing the initial distribution gap between the source (blue) and target (green) domains, and their alignment through adaptation to enable knowledge transfer. Image source: [HTF ⁺ 23] page 2.	36
3.5	Example of GAN-based domain adaptation via CyCADA. The model transforms an image from GTA V, a computer game, to the real-world Cityscapes [COR ⁺ 16] dataset (top), and performs the reverse transformation from Cityscapes to GTA V style (bottom). Images source: [HTP ⁺ 18] page 6	38
3.6	Results of visual alignment via CycleGAN. From left to right: real-world image, Synthetic-To-Real (S2R) adapted image, synthetic image, and Real-To-Synthetic (R2S) adapted image. Images source: [YLZ21] page 10	44
3.7	Comparison of synthetic data (left) and comparable real-world scenes (right), highlighting simplified structure, geometry, and variability in the synthetic data. Image source of real-world pictures: [Cle12, Wor25]	48

4.1	Overview of the Siamese PTZNet for comparing relative FoVs of a PTZ camera X. Two different PTZ camera settings (A and B) generate respective views projected on a sphere. Each view is processed by a CNN backbone with shared weights. Their extracted features are concatenated. The network includes a rotation regressor (for estimating the rotation needed to align one view with the other) and an overlap regressor (for predicting the overlap between both FoVs). The rotation from A (blue) to B (red) is depicted as the green great circle arc. The overlap of FoV A (blue) and FoV B (red) is marked as the green Intersection over Union (IoU) on the sphere.	54
4.2	Spherical IoU calculation of two overlapping FoVs, A (blue) and B (red), visualized on the unit sphere (Equation 4.4). The overlapping area of 8% is marked in green.	56
4.3	Illustration of the geodesic distance (red arc) between two quaternions q_1 and q_2 on the unit sphere, visualizing Equation 4.8. It also shows two antipodal points u and v . Image from [Che16].	58
4.4	Schematic visualization of the relative rotation error $RE_{R,\theta}$ from S to T as symbolic FoV rotation, with $T = \theta_{target}$ and $P = \theta_{pred}$, highlighting how the same fixed 5° absolute deviation results in decreasing relative error magnitudes as the overall target rotation increases.	60
4.5	Schematic visualization of the relative rotation error $RE_{R,\theta}$ from S to T as error bars, with $T = \theta_{target}$ (green) and $P = \theta_{pred}$ (yellow) as the predicted rotation. The absolute error is shown in red. The top row illustrates a large absolute error of 50° , resulting in a 90% $RE_{R,\theta}$, to show the upper bound of the metric. In contrast, the remaining rows fix the absolute angular error at 5° , while increasing the target rotation, demonstrating how the same absolute deviation yields decreasing relative errors, from 50% down to 8%, as θ_{target} grows.	61
4.6	Batch mixing process with a large synthetic and a limited real-world dataset with n images. Two iterators separately iterate, sample, and shuffle the batch with the given R2S ratio of 50%.	63
4.7	DLM of synthetic and a limited number of n real-world images on the dataset level, providing a combined pool of data to sample the batch from.	64
5.1	Cities: Skylines II - Cinematic Camera & Photo Mode UI Example	70
5.2	In-game movement example of a single PTZ camera at a fixed location. The camera is rotated in a controlled way during the day and at night.	71
5.3	A virtually created synthetic image from Cities: Skylines II with annotated camera parameters for position, rotation, and camera intrinsics, resulting in a filename of Fabius_d20240823144956_p-634.8_79.9_-259.2_r16.35_238.60_at56.23787_ap87.06240_f14.00000_sh4.28_sw7.60.jpeg.	73

5.4	Schematic representation of FoV separation for a PTZ camera. The camera covers the lower hemisphere and can tilt around the vertical axis. For training and evaluation data collection, and due to limited hardware availability and mounting constraints, the 360° panning range is split into two half-open intervals: [1–180[for training and [180–360[for evaluation. Some overlap in the lower hemisphere is unavoidable due to tilting, but this separation ensures that the majority of training and evaluation images come from distinct panning regions (P2).	78
5.5	This figure shows the installation of our two HIKVISION PTZ cameras and the scenes they capture to gather training and evaluation data. The upper row shows Camera 1 footage at different times of the day, while the lower row shows Camera 2. The figure demonstrates the real-world application of the schematic from Figure 5.4. Each camera pans over 180° of the shared 360° coverage: the left side (blue) indicates the area used for training images, and the right side (magenta) indicates the portion used for evaluation. The separation is defined by the panning angle, while the cameras tilt within the lower hemisphere.	78
6.1	Schematic overview of the designed PTZNet architecture. The model follows a Siamese design using weight-shared ResNet50 backbones to extract features from an input image pair. The resulting feature vectors are concatenated and routed to specific regression heads to predict the relative rotation quaternion $\mathbf{q} \in \mathbb{R}^4$ and the scalar view overlap $o \in \mathbb{R}$	82
6.2	Architecture of the rotation regression head. It employs a sequence of fully connected layers with exponential dimensionality reduction to regress the final 4D quaternion vector.	83
6.3	Architecture of the overlap regression head. This branch utilizes a steeper exponential reduction scheme to compress features into the final scalar overlap.	83
6.4	Visual comparison of downsizing methods used to perform a 16 : 9 → 1 : 1 transformation, including cropping, padding, and distorting.	86
6.5	Downsizing ablation study for a 25% real-ratio with $n = 5000$, showing absolute rotation ($AE_{R,geo}^\circ$) and overlap ($AE_{O,mae}\%$) errors across epochs.	86
6.6	Example images after applying augmentations, illustrating the resulting diversity. Each image is randomly generated by combining multiple filters such as noise injection, blur, contrast/detail enhancements, color adjustments, advanced color transformations, and compression artifacts.	87
7.1	Comparison of BLM (Section 4.3.1) on the left and DLM (Section 4.3.2) on the right, when combining a large synthetic dataset with a limited real-world dataset of size n . In Batch-Level Mixing, two iterators separately draw and shuffle samples to enforce a fixed R2S ratio (e.g., R1:S1 with 50% real and 50% synthetic). In Dataset-Level Mixing, both datasets are concatenated and sampled by a single iterator, so the batch composition follows the overall dataset proportions.	90

7.2	Performance evaluation of PTZNet trainings with $n = 2500$ real-world image pairs. Comparison of various Batch-Level Mixing (BLM) ratios: R0:S1 (synthetic-only), R1:S0 (real-world only), R1:S3, R1:S1, and R3:S1 (batch ratio) against Dataset-Level Mixing (DLM). DLM does not refer to a fixed ratio, as the batch elements are randomly drawn from a globally mixed dataset (see Figure 7.1). The metrics are reported in degrees for $AE_{R,geo}$ (Equation 4.8) and in percent for the remainder: $RE_{R,\theta}$ (Equation 4.11), $AE_{O,mae}$ (Equation 4.9), RE_O (Equation 4.14), and $RE_{R O,comb}$ (Equation 4.16).	93
7.3	Performance evaluation of PTZNet with $n = 5000$ real-world image pairs. The experimental setup and metric definitions correspond to Figure 7.2.	94
7.4	Performance evaluation of PTZNet with $n = 7500$ real-world image pairs. The experimental setup and metric definitions correspond to Figure 7.2.	95
7.5	Result matrices for $AE_{R,geo}^\circ$ (Equation 4.8), containing the median absolute geodesic rotation errors for all test configurations. The rows contain the results for the respective real-world image pair amount $n = 2500..7500$, with the lowest error for each n marked in bold. Sampling strategies include: synthetic only (R0:S1), mixed synthetic–real batch ratios (R1:S3, R1:S1, R3:S1), real-world only (R1:S0), and Dataset-Level Mixing (DLM). Values in the first row’s matrix correspond to the actual errors, while the two matrices in the second row report the relative deviations to the real-world only ($\Delta R1:S0$) and DLM (ΔDLM) baselines, calculated according to Equation (7.1).	100
7.6	Result matrices for $RE_{R,\theta}\%$ (Equation 4.11), containing the relative geodesic rotation errors for all test configurations, similar to Figure 7.5.	101
7.7	Result matrices for $AE_{O,mae}\%$ (Equation 4.9), containing the absolute overlap errors for all test configurations, similar to Figure 7.5.	101
7.8	Result matrices for $RE_O\%$ (Equation 4.14), containing the relative overlap errors for all test configurations, similar to Figure 7.5.	102
7.9	Result matrices for $RE_{R O,comb}\%$ (Equation 4.16), containing the combined rotation and overlap errors for all test configurations, similar to Figure 7.5.	102



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

3.1	Related Work Comparison: Strengths and Limitations.	51
3.2	Related Work Comparison: Relevance to (P1) and (P2).	52
4.1	Matrix of training configurations combining varying real-world (R) to synthetic (S) data ratios ($r = R : S$) and absolute numbers of real-world images (n). The rows represent the absolute number of real-world images (2500, 5000, 7500, etc.), and the columns represent DLM, as well as the mixing ratios of real-world to synthetic data (R0:S1, R1:S3, R1:S1, R3:S1, R1:S0). The performance of these configurations is evaluated separately for each metric.	64
7.1	Effect of mixed dataset training (strategic mixing of synthetic and real-world samples, as visualized in Figure 7.1) relative to both real-world only (R1:S0) and Dataset-Level Mixing (DLM) baselines across all metrics and real-world sample sizes n . The table summarizes the best-performing trial results, primarily mixed-data strategies, for each metric (mainly R1:S3, but also R1:S1) and compares their median error values to the two baselines. The lowest median error (Error column) within each metric group is highlighted in bold, as are the greatest relative improvements (lowest negative value) in the $\Delta R1:S0$ and ΔDLM columns. These relative deviations, computed according to Equation 7.1, quantify the percentage error improvement achieved by the mixed strategy compared to the baselines. All values are directly selected from the previously presented error matrices (Figures 7.5–7.9).	103
8.1	Evaluation of hypotheses with descriptions and confirmation.	108



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

2.1	Union Sphere IoU Calculation for Camera FoV Overlap	23
5.1	Create a controlled but random PTZ camera movement sequence	72
5.2	Prefix patching of the built-in <i>CaptureScreenshot</i> method to save annotated screenshots each time a screenshot is invoked	76
5.3	Periodic invocation of the patched built-in screenshot method (see patching Algorithm 5.2)	77
5.4	Execute PTZ Movements and Capture Annotated Screenshot	79



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

- Adam** Adaptive Movement Estimation. 85
- ANN** Artificial Neural Network. 7, 9–12, 15, 55, 115
- AP** Average Precision. 46
- APR** Absolute Pose Regression. 28, 30, 49, 51, 52
- BepInEx** Bepis Injector Extensible. 73, 75
- BLM** Batch-Level Mixing. ix, xi, 2, 4, 6, 49–53, 55, 61–63, 65, 83, 89–91, 93, 96–98, 100, 103–105, 108–112, 118, 119
- BMP** Bases Mismatch Penalization. 42
- CDAN** Conditional Domain Adversarial Network. 37, 38, 49
- CDMDA** Cross-Domain Minimization with Deep Autoencoder. 39
- CLAHE** Contrast Limited Adaptive Histogram Equalization. 86
- CNN** Convolutional Neural Network. 13–15, 31, 54, 115, 117
- COD** Conditional Operator Discrepancy. 40
- CoGAN** Coupled Generative Adversarial Network. 37
- COTR** Correspondence Transformer. 32, 49
- CUT** Contrastive Unpaired Translation. 37, 38, 45
- CyCADA** Cycle-Consistent Adversarial Domain Adaptation. 37, 38, 44, 48, 116
- CycleGAN** Cycle-Consistent Adversarial Network. 37, 38, 44, 45, 48, 49, 116
- DAN** Deep Adaptation Network. 35
- DAR** Domain Adaptation Regression. xiii, 35, 39–43

DARC Multi-Domain Adaptation for Regression under Conditional shift. 40

DASGIL Domain Adaptation for Semantic and Geometric-aware Image-based Localization. 42, 45

DINO Distribution-informed Neural Networks. 40

DLM Dataset-Level Mixing. ix, xi, 4, 6, 34, 49, 51, 55, 62, 64, 65, 83, 89–93, 96, 99–105, 108, 109, 112, 117–119, 121

DNN Deep Neural Network. xiii, 8–13

DOF Degrees of Freedom. 30, 51

DRCN Deep Reconstruction-Classification Network. 38, 49

DSAC Differentiable RANSAC. 29

DSC Domain-specific Convolution. 39

DTN Deep Transfer Network. 36, 49

ELU Exponential Linear Unit. 10, 115

FCL Fully Connected Layer. 82, 84

FoV Field of View. ix–xi, 1, 2, 6, 16, 22–26, 31, 32, 51, 54, 56, 60, 67, 68, 74, 75, 78, 79, 89, 115–118, 123

GAN Generative Adversarial Network. 36–38, 49, 116

HFR High-frequency Reconstruction. 39

IoU Intersection over Union. ix, xi, 22, 23, 25, 26, 54, 56, 59, 65, 82, 116, 117, 123

JAN Joint Adaptation Network. 35

JMMD Joint Maximum Mean Discrepancy. 35

KDFM Key-Point Detection and Feature Matching. 28, 49, 51

MAE Mean Absolute Error. 12, 56, 58, 65, 92, 110, 112

MMD Maximum Mean Discrepancy. 35, 40

MSE Mean Squared Error. 12, 56

NCE Noise Contrastive Estimation. 37

NLP Natural Language Processing. 47

NSO Normalized Surface Overlap. 31, 49

NTK Neural Tangent Kernel. 40

ORB Oriented FAST and rotated BRIEF. 29

PAE Camera Pose Auto-Encoder. 46

PSP Pairwise Similarity Preserver. 40

PTZ Pan, Tilt and Zoom. ix–xi, xiii, 1–7, 9, 13, 15–19, 21–29, 31–36, 38, 39, 41–43, 45–57, 61, 62, 67–69, 71–75, 78, 81–83, 85, 88, 89, 91, 92, 97, 98, 105, 107–112, 115–118, 123

R2S Real-To-Synthetic. 3, 44, 45, 50, 51, 62, 63, 90, 91, 105, 107, 109, 110, 112, 116–118

RANSAC Random Sample Consensus. 29

RegDA Regressive Domain Adaptation. 41

ReLU Rectified Linear Unit. 9–11, 84, 115

RKHS Reproducing Kernel Hilbert Space. 35, 40

RPR Relative Pose Regression. 15, 28, 30, 31, 33, 49, 51, 52, 54

RSD Representation Subspace Distance. 42

RUL Remaining Useful Life. 47

S2R Synthetic-To-Real. ix, xi, xiii, 2, 4–6, 27, 28, 33, 35, 39, 41–47, 49–52, 55, 62–64, 67, 81–83, 96, 108, 109, 111, 112, 116

SCR Scene Coordinate Regression. 28–30, 49, 51, 52

SfM Structure from Motion. 31

SGD Stochastic Gradient Descent. 85

SIFT Scale-Invariant Feature Transform. 29, 31

SNN Siamese Neural Network. 13, 15, 30, 31, 54, 115

SOTA State of the Art. 5, 33, 34, 39, 112

SURF Speeded Up Robust Features. 29

UGC User-Generated Content. 69

WANN Weighting Adversarial Neural Network. 41

WMSN Wireless Multimedia Sensor Network. 31

Bibliography

- [ABR⁺12] Gretton Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Smola Alexander J. A Kernel Two-Sample Test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.
- [AEBM19] Benrazek Ala-Eddine, Farou Brahim, and Kurulay Muhammet. Efficient Camera Clustering Method Based on Overlapping FoVs for WMSNs. *International Journal of Informatics and Applied Mathematics*, 1(1):10–23, 2019.
- [AK23] D. Acharya and K. Khoshelham. Reverse Domain Adaptation for Indoor Camera Pose Regression. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 10:453–460, 2023.
- [AMS⁺22] Fatimah Alzubaidi, Peyman Mostaghimi, Guangyao Si, Pawel Swietojanski, and Ryan Armstrong. Automated rock quality designation using convolutional neural networks. *Rock Mechanics and Rock Engineering*, 55:3719–3734, 2022.
- [Aro50] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.
- [Bep24] BepInEx Team. Bepis Injector Extensible. <https://github.com/BepInEx/BepInEx>, 2024. Accessed: 2025-03-25.
- [BGL⁺93] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature Verification using a “Siamese” Time Delay Neural Network. In *NIPS*, pages 737–744, 1993.
- [BIK⁺20] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.
- [BK73] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

- [BKK21] Monika Bansal, Munish Kumar, and Manish Kumar. 2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors. *Multimedia Tools and Applications*, 80:18839–18857, 2021.
- [BKN⁺17] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC — Differentiable RANSAC for Camera Localization. In *CVPR*, pages 2492–2500, 2017.
- [BLP18] Vassileios Balntas, Shuda Li, and Victor Prisacariu. RelocNet: Continuous Metric Learning Relocalisation Using Neural Nets. In *ECCV*, pages 782–799, 2018.
- [BR18] Eric Brachmann and Carsten Rother. Learning Less is More - 6D Camera Localization via 3D Surface Regression. In *CVPR*, pages 4654–4662, 2018.
- [BR21] Eric Brachmann and Carsten Rother. Visual Camera Re-Localization from RGB and RGB-D Images Using DSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5847–5865, 2021.
- [BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *ECCV*, pages 404–417, 2006.
- [CBG13] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML Workshop on Challenges in Representation Learning*, volume 2, 2013.
- [Che16] CheCheDaWaff. Illustration of great-circle distance. https://commons.wikimedia.org/wiki/File:Illustration_of_great-circle_distance.svg, 2016. Licensed under CC BY-SA 4.0, Accessed: 2025-04-08.
- [CHX⁺22] Ying Chen, Dihe Huang, Shang Xu, Jianlin Liu, and Yong Liu. Guide Local Feature Matching by Overlap Estimation. In *AAAI*, volume 36, pages 365–373, 2022.
- [Cle12] Evan Cleary. Bad driving habits – part 1. <https://www.manwithavandublin.ie/how-to-indicate-correctly-on-roundabouts/>, 2012. Accessed: 2025-06-19.
- [COR⁺16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016.

- [CSM21] Kefan Chen, Noah Snavely, and Ameesh Makadia. Wide-Baseline Relative Camera Pose Estimation with Directional Learning. In *CVPR*, pages 3257–3267, 2021.
- [CSVV20] Jorge Charco, Angel Sappa, Boris Vintimilla, and Henry Velesaca. Transfer Learning from Synthetic Data in the Camera Pose Estimation Problem. In *VISIGRAPP*, pages 498–505, 2020.
- [CSVV21] Jorge L. Charco, Angel D. Sappa, Boris X. Vintimilla, and Henry O. Velesaca. Camera pose estimation in multi-view environments: From virtual scenarios to the real world. *Image and Vision Computing*, 110(C):104182, 2021.
- [CWWL21] Xinyang Chen, Sinan Wang, Jianmin Wang, and Mingsheng Long. Representation Subspace Distance for Domain Adaptation Regression. In *ICML*, pages 1749–1759, 2021.
- [DCS⁺17] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017.
- [DDS⁺09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.
- [DRC⁺17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *CoRL*, pages 1–16, 2017.
- [DUK22] A. Dashora, C. S. Utlal, and A. V. Kulkarni. A Field-Based Method for Estimation of Overlap for Convergent Images for View Planning of Buildings. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B2-2022:29–35, 2022.
- [DWS⁺19] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. CamNet: Coarse-to-Fine Retrieval for Camera Re-Localization. In *ICCV*, pages 2871–2880, 2019.
- [Ebe02] David Eberly. Quaternions and Rotations. *Magic Software Inc*, 26:1–8, 2002.
- [EEDV⁺22] Salehe Erfanian Ebadi, Saurav Dhakad, Sanjay Vishwakarma, Chunpu Wang, You-Cyuan Jhang, Maciek Chociej, Adam Crespi, Alex Thaman, and Sujoy Ganguly. Psp-hdri+: A synthetic dataset generator for pre-training of human-centric computer vision models. In *ICML Workshop on Pre-training*, 2022.

- [FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [FEF⁺17] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Natalia Neverova, Alain Trémeau, and Christian Wolf. Multi-task, multi-domain learning: Application to semantic segmentation and pose regression. *Neurocomputing*, 251:68–80, 2017.
- [FPL⁺24] Muhammad Tanzil Furqon, Mahardhika Pratama, Lin Liu, Habibullah, and Kutluyil Dogancay. Mixup Domain Adaptations for Dynamic Remaining Useful Life Predictions. *Knowledge-Based Systems*, 295:111783, 2024.
- [FVRA20] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A Brief Review of Domain Adaptation. In *ICDATA-IKE*, pages 877–894, 2020.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.
- [GCJT21] Romain Guesdon, Carlos Crispim-Junior, and Laure Tougne. Dripe: A dataset for human pose estimation in real-world driving settings. In *ICCVW*, pages 2865–2874, 2021.
- [GFW⁺25] Jinhui Guo, Lubin Fan, Bojian Wu, Jiaqi Gu, Shen Cao, and Jieping Ye. PTZ-Calib: Robust Pan-Tilt-Zoom Camera Calibration. In *ICRA*, pages 1907–1913, 2025.
- [GH10] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304, 2010.
- [GKZ⁺16] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation. In *ECCV*, pages 597–613, 2016.
- [Gmb25] Isarsoft GmbH. What is pan-tilt-zoom (ptz)? <https://www.isarsoft.com/knowledge-hub/ptz>, 2025. Accessed: 2025-09-06.
- [GPAM⁺14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *NIPS*, volume 2, pages 2672–2680, 2014.
- [GSS15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. arXiv:1412.6572.

- [GUA⁺17] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2017.
- [HK18] Seyyed Ali Hoseini and Peyman Kabiri. Camera pose estimation in unknown environments using a sequence of wide-baseline monocular images. *Journal of AI and Data Mining*, 6:93–103, 2018.
- [HLX22] Shishuai Hu, Zehui Liao, and Yong Xia. Domain specific convolution and high frequency reconstruction based unsupervised domain adaptation for medical image segmentation. In *MICCAI*, pages 650–659, 2022.
- [HQC⁺20] Hanjiang Hu, Zhijian Qiao, Ming Cheng, Zhe Liu, and Hesheng Wang. DASGIL: Domain Adaptation for Semantic and Geometric-aware Image-based Localization. *IEEE Transactions on Image Processing*, 30:1342–1353, 2020.
- [HSC⁺19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *ICCV*, pages 1314–1324, 2019.
- [HSS12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6e: RMSProp—Divide the gradient by a running average of its recent magnitude. Technical report, University of Toronto, 2012. Coursera: Neural Networks for Machine Learning.
- [HTF⁺23] Chunmei He, Taifeng Tan, Xianjun Fan, Lanqing Zheng, and Zhengchun Ye. Noise-residual mixup for unsupervised adversarial domain adaptation. *Applied Intelligence*, 53(3):3034–3047, 2023.
- [HTP⁺18] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *ICML*, pages 1989–1998, 2018.
- [HXC25] Hengzhi He, Shirong Xu, and Guang Cheng. Golden Ratio Mixing of Real and Synthetic Data for Stabilizing Generative Model Training. arXiv:2502.18049, 2025.
- [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

- [i2t19] i2tutorials.com. Explain Activation Function in Neural Network and its types. <https://www.i2tutorials.com/wp-content/media/2019/09/Deep-learning-20-i2tutorials.png>, 2019. Accessed: 2025-05-02.
- [ISB22] Benedikt T. Imbusch, Max Schwarz, and Sven Behnke. Synthetic-to-Real Domain Adaptation using Contrastive Unpaired Translation. In *CASE*, pages 595–602, 2022.
- [Jad18] Shruti Jadon. Introduction to different activation functions for deep learning. *Medium, Augmenting Humanity*, 16, 2018.
- [JCJ20] Boyuan Jiang, Chao Chen, and Xinyu Jin. Unsupervised domain adaptation with target reconstruction and label confusion in the common subspace. *Neural Computing and Applications*, 32(9):4743–4756, 2020.
- [JJW⁺21] Janguang Jiang, Yifei Ji, Ximei Wang, Yufeng Liu, Jianmin Wang, and Mingsheng Long. Regressive Domain Adaptation for Unsupervised Key-point Detection. In *CVPR*, pages 6776–6785, 2021.
- [JR21] Charles Jin and Martin Rinard. Towards Context-Agnostic Learning Using Synthetic Data. *Advances in Neural Information Processing Systems*, 34:26223–26236, 2021.
- [JTH⁺21] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: Correspondence Transformer for Matching Across Images. In *ICCV*, pages 6187–6197, 2021.
- [KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.
- [KC17] Alex Kendall and Roberto Cipolla. Geometric Loss Functions for Camera Pose Regression with Deep Learning. In *CVPR*, pages 6555–6564, 2017.
- [KGC15] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, pages 2938–2946, 2015.
- [KK22] Daewoon Kim and Kwanghee Ko. Camera localization with Siamese neural networks using iterative relative pose estimation. *Journal of Computational Design and Engineering*, 9(4):1482–1497, 2022.
- [KPS17] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. arXiv:1710.02726, 2017.

- [KWSS22] Donghyun Kim, Kaihong Wang, Stan Sclaroff, and Kate Saenko. A Broad Study of Pre-training for Domain Generalization and Adaptation. In *ECCV*, pages 621–638, 2022.
- [KZS15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-shot Image Recognition. In *ICML Deep Learning Workshop*, 2015.
- [Lav25] Lavanya Shukla. Designing Your Neural Networks. <https://www.kdnuggets.com/2019/11/designing-neural-networks.html>, 2025. Accessed: 2025-05-02.
- [LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
- [LCWJ15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning Transferable Features with Deep Adaptation Networks. In *ICML*, pages 97–105, 2015.
- [LCWJ18] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional Adversarial Domain Adaptation. In *NIPS*, pages 1647–1657, 2018.
- [Lei22] Lei Mao. Unit Quaternion 3D Rotation Representation. <https://leimao.github.io/blog/3D-Rotation-Unit-Quaternion/>, 2022. Accessed: 2025-04-07.
- [LHC25] Lingkun Luo, Shiqiang Hu, and Liming Chen. Beyond Batch Learning: Global Awareness Enhanced Domain Adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):4473–4488, 2025.
- [LKB⁺23] Jieun Lee, Tae-yong Kim, Seunghyo Beak, Yeeun Moon, and Jongpil Jeong. Real-time pose estimation based on ResNet-50 for rapid safety prevention and accident detection for field workers. *Electronics*, 12(16), 2023.
- [Low04] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
- [LPYL20] Sangrok Lee, Eunsoo Park, Hongsuk Yi, and Sang Hun Lee. StR-DAN: Synthetic-to-Real Domain Adaptation Network for Vehicle Re-Identification. In *CVPRW*, pages 608–609, 2020.
- [LSSK19] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. arXiv:1903.06733, 2019.

- [LT16] Ming-Yu Liu and Oncel Tuzel. Coupled Generative Adversarial Networks. In *NIPS*, pages 469–477, 2016.
- [LXW21] Cuiyin Liu, Jishang Xu, and Feng Wang. A Review of Keypoints’ Detection and Feature Description in Image Registration. *Scientific Programming*, 2021:1–25, 2021.
- [LYS⁺23] Kai Leng, Cong Yang, Wei Sui, Jie Liu, and Zhijun Li. Sitpose: A Siamese Convolutional Transformer for Relative Camera Pose Estimation. In *ICME*, pages 1871–1876, 2023.
- [LZWJ17] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep Transfer Learning with Joint Adaptation Networks. In *ICML*, pages 2208–2217, 2017.
- [Met25] Metric System. steradian. <https://metricsystem.net/derived-units/special-names/steradian/>, 2025. Accessed: 2025-04-07.
- [Mir14] Mehdi Mirza. Conditional generative adversarial nets. arXiv:1411.1784, 2014.
- [MMM12] Mehryar Mohri and Andrés Muñoz Medina. New analysis and algorithm for learning with drifting distributions. In *ALT*, pages 124–138, 2012.
- [MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- [MRD⁺21] Antoine de Mathelin, Guillaume Richard, Francois Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Adversarial Weighting for Domain Adaptation in Regression. In *ICTAI*, pages 49–56, 2021.
- [MYKR17] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Image-Based Localization Using Hourglass Networks. In *ICCVW*, pages 870–877, 2017.
- [New25] News9Live. Live | dense layer of fog blankets north india; zero visibility at several airports. <https://www.news9live.com/india/fog-in-north-india-live-updates-travel-weather-2787867>, 2025. Accessed: 2025-09-06.
- [NH10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.
- [Par23] Paradox Interactive. User Agreement - User Generated Content. <https://legal.paradoxplaza.com/eula>, 2023. Accessed: 2025-03-25.

- [PEZZ20] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive Learning for Unpaired Image-to-Image Translation. In *ECCV*, pages 319–345, 2020.
- [PRS⁺12] Horst Possegger, Matthias R  ther, Sabine Sternig, Thomas Mauthner, Manfred Klopschitz, Peter M Roth, and Horst Bischof. Unsupervised calibration of camera networks and virtual ptz cameras. In *CVWW*, volume 2, page 7, 2012.
- [PUK⁺18] Xingchao Peng, Ben Usman, Neela Kaushik, Dequan Wang, Judy Hoffman, and Kate Saenko. VisDA: A Synthetic-to-Real Benchmark for Visual Domain Adaptation. In *CVPRW*, pages 2102–21025, 2018.
- [Qia99] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.
- [QSMG17] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, pages 652–660, 2017.
- [RGHS⁺20] Anita Rau, Guillermo Garcia-Hernando, Danail Stoyanov, Gabriel J. Brostow, and Daniyar Turmukhambetov. Predicting Visual Overlap of Images Through Interpretable Non-Metric Box Embeddings. In *ECCV*, pages 629–646, 2020.
- [RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.
- [RMVH22] Praveen Kumar Rajendran, Sumit Mishra, Luiz Felipe Vecchietti, and Dongsoo Har. RelMobNet: End-to-end relative camera pose estimation using a robust two-stage training. In *ECCV*, pages 238–252, 2022.
- [Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.
- [RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, pages 2564–2571, 2011.
- [RTG⁺19] Hamid Rezatofghi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *CVPR*, pages 658–666, 2019.
- [SB20] Max Schwarz and Sven Behnke. Stilleben: Realistic scene synthesis for deep learning in robotics. In *ICRA*, pages 10502–10508, 2020.

- [SGZ⁺13] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *CVPR*, pages 2930–2937, 2013.
- [SK22] Yoli Shavit and Yosi Keller. Camera Pose Auto-encoders for Improving Pose Regression. In *ECCV*, volume 13670, pages 140–157, 2022.
- [Sky25] SkylineWebcams. Tokyo - shinjuku live cam. <https://www.skylinewebcams.com/en/webcam/japan/kanto/tokyo/tokyo-shinjuku.html>, 2025. Accessed: 2025-09-06.
- [Sla99] Gregory G. Slabaugh. Computing Euler angles from a rotation matrix. Technical report, Queen Mary University of London, 1999.
- [Sph25a] Spherical Geometry Team. Spherical Geometry. https://github.com/spacetelescope/spherical_geometry/, 2025. Accessed: 2025-04-07.
- [Sph25b] Spherical Geometry Team. Spherical Polygon. https://github.com/spacetelescope/spherical_geometry/blob/master/spherical_geometry/polygon.py, 2025. Accessed: 2025-04-07.
- [Spö16] J. Spörri. Research dedicated to sports injury prevention – the ‘sequence of prevention’ on the example of alpine ski racing. Habilitation thesis for Venia Docendi in “Biomechanics”, 2016.
- [SSN⁺23] Daniel Sagmeister, Dominik Schörkhuber, Matej Nezveda, Fabian Stiedl, Maria Schimkowitsch, and Margrit Gelautz. Transfer Learning for Driver Pose Estimation from Synthetic Data. In *IV*, pages 1–7, 2023.
- [SUL⁺21] Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the Feature: Learning Robust Camera Localization From Pixels To Pose. In *CVPR*, pages 3247–3257, 2021.
- [TCERP⁺24] Juan Terven, Diana M. Cordova-Esparza, Alfonso Ramirez-Pedraza, Edgar A. Chavez-Urbiola, and Julio A. Romero-Gonzalez. Loss Functions and Metrics in Deep Learning. arXiv:2307.02694, 2024.
- [Tea23] The Mammoth Security Team. Ptz vs. bullet cameras. <https://mammothsecurity.com/blog/ptz-vs-bullet-cameras>, 2023. Accessed: 2025-09-06.
- [TNPB23] Zahra Taghiyarrenani, Sławomir Nowaczyk, Sepideh Pashami, and Mohamed-Rafik Bouguelia. Multi-domain adaptation for regression under

conditional distribution shift. *Expert Systems with Applications*, 224:119907, 2023.

- [Tri09] Christopher Triola. Special Orthogonal Groups and Rotations. 2009.
- [Vic01] Leandra Vicci. Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation. Technical report, University of North Carolina at Chapel Hill, 2001.
- [Voi21] John Voight. *Quaternion Algebras*. 2021.
- [VSvLD⁺20] Glenn Van Steenkiste, Gunther van Loon, Annelies Decloedt, Guillaume Crevecoeur, and tammo delhaas. *Equine electrocardiography revisited: 12-lead recording, vectorcardiography and the power of machine intelligence*. PhD thesis, 2020.
- [VVL19] Márton Véges, Viktor Varga, and András Lőrincz. 3d human pose estimation with siamese equivariant embedding. *Neurocomputing*, 339:194–201, 2019.
- [WD18] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.
- [WHW⁺22] Jun Wu, Jingrui He, Sheng Wang, Kaiyu Guan, and Elizabeth Ainsworth. Distribution-Informed Neural Networks for Domain Adaptation Regression. In *NIPS*, volume 729, pages 10040–10054, 2022.
- [WLX⁺20] Weijia Wu, Ning Lu, Enze Xie, Yuxing Wang, Wenwen Yu, Cheng Yang, and Hong Zhou. Synthetic-to-Real Unsupervised Domain Adaptation for Scene Text Detection in the Wild. In *ACCV*, pages 289–303, 2020.
- [Wor25] WorldCam. Marshall - downtown. <https://worldcam.eu/webcams/north-america/michigan-usa/19918-marshall-downtown>, 2025. Accessed: 2025-06-19.
- [XEOT12] Jianxiong Xiao, K. Alex Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. In *CVPR*, pages 2695–2702, 2012.
- [Xia19] Xiaoqiang. Gradient descent method-Gradient descent. <https://easyai.tech/en/ai-definition/gradient-descent/>, 2019. Accessed: 2025-05-02.
- [XWKA⁺22] Haifeng Xia, Pu Perry Wang, Toshiaki Koike-Akino, Ye Wang, Philip Orlik, and Zhengming Ding. Adversarial Bi-Regressor Network for Domain Adaptive Regression. In *IJCAI*, pages 3583–3589, 2022.

- [XWX⁺22] Meng Xu, Youchen Wang, Bin Xu, Jun Zhang, Jian Ren, Stefan Poslad, and Pengfei Xu. A Critical Analysis of Image-based Camera Pose Estimation Techniques. *Neurocomputing*, 570:127125, 2022.
- [YBT⁺19] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. SANet: Scene Agnostic Network for Camera Localization. In *ICCV*, pages 42–51, 2019.
- [YLZ20] Chenhao Yang, Yuyi Liu, and Andreas Zell. RCPNet: Deep-Learning based Relative Camera Pose Estimation for UAVs. In *ICUAS*, pages 1085–1092. IEEE, 2020.
- [YLZ21] Chenhao Yang, Yuyi Liu, and Andreas Zell. Relative Camera Pose Estimation using Synthetic Data with Domain Adaptation via Cycle-Consistent Adversarial Networks. *Journal of Intelligent & Robotic Systems*, 102(4):79, 2021.
- [YRL24] Hao-Ran Yang, Chuan-Xian Ren, and You-Wei Luo. COD: Learning Conditional Invariant Representation for Domain Adaptation Regression. In *ECCV*, pages 108–125, 2024.
- [YWQX21] Wenpeng Yin, Huan Wang, Jin Qu, and Caiming Xiong. BatchMixup: Improving Training by Interpolating Hidden States of the Entire Mini-batch. In *ACL-IJCNLP*, pages 4908–4912, 2021.
- [ZF14] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *ECCV*, pages 818–833, 2014.
- [ZFCB⁺22] Francesco Zola, Jose Alvaro Fernandez-Carrasco, Jan Lukas Bruse, Mikel Galar, and Zeno Geradts. Verification system based on long-range iris and graph siamese neural networks. In *ESSE*, pages 80–88, 2022.
- [ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *ICCV*, pages 2242–2251, 2017.
- [ZQD⁺20] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):46–76, 2020.
- [ZRK⁺20] Chaoning Zhang, Francois Rameau, Junsik Kim, Dawit Mureja Argaw, Jean-Charles Bazin, and In So Kweon. DeepPTZ: Deep Self-Calibration for PTZ Cameras. In *WACV*, pages 1030–1038, 2020.
- [Zui94] Karel Zuiderveld. Contrast limited adaptive histogram equalization. In Paul S. Heckbert, editor, *Graphics Gems IV*, pages 474–485. Academic Press, 1994.

- [ZYCW15] Xu Zhang, Felix Xinnan Yu, Shih-Fu Chang, and Shengjin Wang. Deep Transfer Network: Unsupervised Domain Adaptation. arXiv:1503.00591, 2015.